```
plt.close()
```

module 2

1.Using Deep pre-trained CNN model for feature extraction:

- **Extract features from the FC1 of VGG network.**
- **Train any traditional ML model like SVM for classification.**
- **Repeat the above by considering FC2 of VGG for feature extraction.**

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.datasets import mnist
# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# Normalize pixel values to between 0 and 1
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
# Reshape the images to add a channel dimension
X_train = np.expand_dims(X_train, axis=-1)
X_test = np.expand_dims(X_test, axis=-1)
# Load pre-trained VGG16 model without top (fully connected) layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
# Add fully connected layers on top of VGG16 base model
x = Flatten()(base_model.output)
fc1 = Dense(4096, activation='relu', name='fc1')(x)
fc2 = Dense(4096, activation='relu', name='fc2')(fc1)
# Create models for extracting features from FC1 and FC2 layers
feature_extractor_fc1 = Model(inputs=base_model.input, outputs=fc1)
```

```
feature_extractor_fc2 = Model(inputs=base_model.input, outputs=fc2)
# Resize images to fit VGG16 input shape
X_train_resized = tf.image.resize(X_train, (224, 224)).numpy()
X_test_resized = tf.image.resize(X_test, (224, 224)).numpy()
# Extract features from FC1 and FC2 layers
X_train_features_fc1 = feature_extractor_fc1.predict(X_train_resized)
X_test_features_fc1 = feature_extractor_fc1.predict(X_test_resized)
X_train_features_fc2 = feature_extractor_fc2.predict(X_train_resized)
X_test_features_fc2 = feature_extractor_fc2.predict(X_test_resized)
# Flatten the features
X_train_features_fc1_flat = X_train_features_fc1.reshape((X_train_features_fc1.shape[0], -1))
X_test_features_fc1_flat = X_test_features_fc1.reshape((X_test_features_fc1.shape[0], -1))
X_train_features_fc2_flat = X_train_features_fc2.reshape((X_train_features_fc2.shape[0], -1))
X_test_features_fc2_flat = X_test_features_fc2.reshape((X_test_features_fc2.shape[0], -1))
# Train SVM model using features extracted from FC1
svm_fc1 = SVC()
svm_fc1.fit(X_train_features_fc1_flat, y_train)
y_pred_fc1 = svm_fc1.predict(X_test_features_fc1_flat)
accuracy_fc1 = accuracy_score(y_test, y_pred_fc1)
print("Accuracy using features from FC1:", accuracy_fc1)
# Train SVM model using features extracted from FC2
svm_fc2 = SVC()
svm_fc2.fit(X_train_features_fc2_flat, y_train)
y_pred_fc2 = svm_fc2.predict(X_test_features_fc2_flat)
accuracy_fc2 = accuracy_score(y_test, y_pred_fc2)
print("Accuracy using features from FC2:", accuracy_fc2)
```

2. Fine-tuning Deep pre-trained CNN for Classification:

- **Fine-tune VGG network for the task under consideration.**
- **Check the performance by making.**
- **all the layers trainable.**
- **freezing the initial layers.**

- **freezing the entire network except the final layer.**

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import SGD
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from tensorflow.keras.datasets import mnist
# Load MNIST dataset
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# Normalize pixel values to between 0 and 1
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0
# Resize images to fit VGG16 input shape
X_train_resized = tf.image.resize(X_train[..., tf.newaxis], (224, 224))
X_test_resized = tf.image.resize(X_test[..., tf.newaxis], (224, 224))
# Convert labels to one-hot encoding
y_train = tf.keras.utils.to_categorical(y_train, num_classes=10)
y_test = tf.keras.utils.to_categorical(y_test, num_classes=10)
# Load pre-trained VGG16 model without top (fully connected) layers
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
# Fine-tuning with all layers trainable
model_all_trainable = Sequential([
    base_model,
    Flatten(),
    Dense(512, activation='relu'),
    Dense(10, activation='softmax')
])
```