```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("kabil007/lungcancer4types-
imagedataset")

print("Path to dataset files:", path)

import os
import glob
import cv2
import numpy as np

from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, confusion_matrix

IMG_SIZE = 64

DATA_ROOT = "Data"  # <- folder from your screenshot
TRAIN_DIR = os.path.join(DATA_ROOT, "train")
VALID_DIR = os.path.join(DATA_ROOT, "valid")
TEST_DIR  = os.path.join(DATA_ROOT, "test")

RANDOM_STATE = 42

def canonical_label_from_folder(folder_name: str) -> str:
    """
    Maps a folder name to the class label.
    For 'adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib' ->
'adenocarcinoma'
    For 'large.cell.carcinoma' -> 'large.cell.carcinoma'
    For 'normal' -> 'normal'
    """
    return folder_name.split("_")[0]

def build_label_map(train_dir):
    labels = set()
    for folder in os.listdir(train_dir):
        full_path = os.path.join(train_dir, folder)
        if not os.path.isdir(full_path):
            continue
        label = canonical_label_from_folder(folder)
        labels.add(label)

    class_names = sorted(labels)
    label_to_idx = {label: idx for idx, label in
enumerate(class_names)}
    return label_to_idx, class_names
```

```
label_to_idx, class_names = build_label_map(TRAIN_DIR)
print("Classes:", class_names)
```

```
Classes: ['adenocarcinoma', 'large.cell.carcinoma', 'normal',
'squamous.cell.carcinoma']
```

```
def load_split(split_dir, label_to_idx, img_size=64):
    X, y = [], []

    # support multiple image extensions
    exts = ("*.png", "*.jpg", "*.jpeg", "*.bmp")

    for folder in os.listdir(split_dir):
        folder_path = os.path.join(split_dir, folder)
        if not os.path.isdir(folder_path):
            continue

        label_name = canonical_label_from_folder(folder)
        if label_name not in label_to_idx:
            # unknown label in this split, skip or raise an error
            print(f"Warning: skipped folder '{folder}' (unknown label
'{label_name}')")
            continue

        label_idx = label_to_idx[label_name]

        # iterate over all supported extensions
        img_paths = []
        for ext in exts:
            img_paths.extend(glob.glob(os.path.join(folder_path,
ext)))

        for img_path in img_paths:
            img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
            if img is None:
                continue

            img = cv2.resize(img, (img_size, img_size))
            img = img.astype(np.float32) / 255.0
            X.append(img.flatten())
            y.append(label_idx)

    X = np.array(X, dtype=np.float32)
    y = np.array(y, dtype=np.int64)
    return X, y

X_train, y_train = load_split(TRAIN_DIR, label_to_idx, IMG_SIZE)
X_valid, y_valid = load_split(VALID_DIR, label_to_idx, IMG_SIZE)
X_test,  y_test  = load_split(TEST_DIR,  label_to_idx, IMG_SIZE)

print("Train:", X_train.shape, y_train.shape)
print("Valid:", X_valid.shape, y_valid.shape)
print("Test :", X_test.shape,  y_test.shape)
```

```
Train: (613, 4096) (613,)
Valid: (72, 4096) (72,)
Test : (315, 4096) (315,)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_valid_scaled = scaler.transform(X_valid)
X_test_scaled  = scaler.transform(X_test)

mlp = MLPClassifier(
    hidden_layer_sizes=(512, 256),
    activation='relu',
    solver='adam',
    alpha=1e-4,
    batch_size=64,
    learning_rate_init=1e-3,
    max_iter=50,
    shuffle=True,
    random_state=RANDOM_STATE,
    early_stopping=False,  # we already have an explicit valid set
    verbose=True
)

mlp.fit(X_train_scaled, y_train)
```

```
Iteration 1, loss = 1.58690908
Iteration 2, loss = 0.49536652
Iteration 3, loss = 0.19321894
Iteration 4, loss = 0.16056618
Iteration 5, loss = 0.16514373
Iteration 6, loss = 0.07103906
Iteration 7, loss = 0.02708962
Iteration 8, loss = 0.03215769
Iteration 9, loss = 0.03401996
Iteration 10, loss = 0.03868072
Iteration 11, loss = 0.02253753
Iteration 12, loss = 0.05003109
Iteration 13, loss = 0.01930694
Iteration 14, loss = 0.02118543
Iteration 15, loss = 0.01095505
Iteration 16, loss = 0.02655013
Iteration 17, loss = 0.01002205
Iteration 18, loss = 0.00766663
Iteration 19, loss = 0.00587946
Iteration 20, loss = 0.00712248
Iteration 21, loss = 0.00843560
Iteration 22, loss = 0.01206040
Iteration 23, loss = 0.00414331
Iteration 24, loss = 0.00977836
Iteration 25, loss = 0.00332852
Iteration 26, loss = 0.01649402
Iteration 27, loss = 0.00479262
Iteration 28, loss = 0.00731907
Iteration 29, loss = 0.00467991
Iteration 30, loss = 0.00682212
Iteration 31, loss = 0.00908221
Iteration 32, loss = 0.00466225
Iteration 33, loss = 0.00387615
Iteration 34, loss = 0.00553482
Iteration 35, loss = 0.00484127
Iteration 36, loss = 0.00643599
Training loss did not improve more than tol=0.000100 for 10
consecutive epochs. Stopping.

MLPClassifier(batch_size=64, hidden_layer_sizes=(512, 256),
max_iter=50,
              random_state=42, verbose=True)
```

```
print("=== Validation set ===")
y_valid_pred = mlp.predict(X_valid_scaled)
print(classification_report(y_valid, y_valid_pred,
target_names=class_names))
print("Confusion matrix (valid):")
print(confusion_matrix(y_valid, y_valid_pred))

print("\n=== Test set ===")
y_test_pred = mlp.predict(X_test_scaled)
print(classification_report(y_test, y_test_pred,
target_names=class_names))
print("Confusion matrix (test):")
print(confusion_matrix(y_test, y_test_pred))
```

```
=== Validation set ===
                         precision    recall  f1-score   support

         adenocarcinoma       0.75      0.78      0.77        23
   large.cell.carcinoma       0.70      0.67      0.68        21
                 normal       1.00      0.92      0.96        13
squamous.cell.carcinoma       0.69      0.73      0.71        15

               accuracy                           0.76        72
              macro avg       0.78      0.78      0.78        72
           weighted avg       0.77      0.76      0.77        72

Confusion matrix (valid):
[[18  4  0  1]
 [ 4 14  0  3]
 [ 0  0 12  1]
 [ 2  2  0 11]]

=== Test set ===
                         precision    recall  f1-score   support

         adenocarcinoma       0.53      0.30      0.38       120
   large.cell.carcinoma       0.38      0.59      0.47        51
                 normal       0.83      0.91      0.87        54
squamous.cell.carcinoma       0.53      0.64      0.58        90

               accuracy                           0.55       315
              macro avg       0.57      0.61      0.57       315
           weighted avg       0.56      0.55      0.54       315

Confusion matrix (test):
[[36 34  9 41]
 [ 9 30  1 11]
 [ 4  1 49  0]
 [19 13  0 58]]
```

```python
def predict_ct_section(img_path, model, scaler, img_size,
class_names):
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    if img is None:
        raise ValueError(f"Could not read image: {img_path}")

    img = cv2.resize(img, (img_size, img_size))
    img = img.astype(np.float32) / 255.0
    x = img.flatten().reshape(1, -1)

    x_scaled = scaler.transform(x)
    probs = model.predict_proba(x_scaled)[0]
    idx = np.argmax(probs)
    return class_names[idx], probs


test_img =
"/content/Data/valid/adenocarcinoma_left.lower.lobe_T2_N0_M0_Ib/000112
(2).png"
pred_class, probs = predict_ct_section(test_img, mlp, scaler,
IMG_SIZE, class_names)
print("Predicted:", pred_class)
if pred_class == "normal":
    print("→ interpreted as negative (no malignant cancer).")
else:
    print("→ interpreted as cancer type:", pred_class)
```

```
Predicted: adenocarcinoma
→ interpreted as cancer type: adenocarcinoma
```