Image Classification

Reading in the dataset[1]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
import cv2
from tqdm import tqdm
import os
import random

num_classes = 6
epochs = 10

IMAGE_SIZE = (150,150)

class_names = ['mountain', 'street', 'glacier', 'buildings', 'sea', 'forest']
class_labels = {class_name:i for i, class_name in enumerate(class_names)}

datasets = ['/kaggle/input/intel-image-classification/seg_train/seg_train','/kaggle/input/i

output = []

for dataset in datasets:

    images =[]
    labels = []

    print("Loading {}".format(dataset))

    for folder in os.listdir(dataset):
        label = class_labels[folder]

        for file in tqdm(os.listdir(os.path.join(dataset, folder))):

            img_path = os.path.join(os.path.join(dataset, folder), file)

            image = cv2.imread(img_path)
            image = cv2.resize(image, IMAGE_SIZE)

            images.append(image)
            labels.append(label)
    images = np.array(images, dtype = 'float32')
    labels = np.array(labels, dtype = 'int32')

    output.append((images, labels))
```

```
Loading /kaggle/input/intel-image-classification/seg_train/seg_train
100%|██████████| 2512/2512 [00:11<00:00, 225.54it/s]
100%|██████████| 2382/2382 [00:11<00:00, 215.57it/s]
100%|██████████| 2191/2191 [00:13<00:00, 160.64it/s]
100%|██████████| 2274/2274 [00:11<00:00, 199.82it/s]
100%|██████████| 2271/2271 [00:13<00:00, 164.46it/s]
100%|██████████| 2404/2404 [00:10<00:00, 222.93it/s]
Loading /kaggle/input/intel-image-classification/seg_test/seg_test
100%|██████████| 525/525 [00:02<00:00, 235.33it/s]
100%|██████████| 501/501 [00:02<00:00, 239.16it/s]
100%|██████████| 437/437 [00:01<00:00, 238.41it/s]
100%|██████████| 510/510 [00:02<00:00, 249.46it/s]
100%|██████████| 474/474 [00:02<00:00, 219.13it/s]
100%|██████████| 553/553 [00:02<00:00, 201.24it/s]
```

Seperate the dataset into test and train. Then randomize the order of the train set.

```
(train_images, train_labels), (test_images, test_labels) = output
from sklearn.utils import shuffle
train_images, train_labels = shuffle(train_images, train_labels, random_state=1234)
```

```
print(train_images.shape)
print(train_labels.shape)
```

```
(14034, 150, 150, 3)
(14034,)
```

Sequential Model:

```
train_images = train_images/255.0
test_images = test_images/255.0
```

Scale the images.

```
seqModel = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(150,150,3)),
    tf.keras.layers.Dense(1000, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1000, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1000, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(6, activation='softmax'),
])
```

```
2022-12-02 15:45:05.318742: I tensorflow/core/common_runtime/process_util.cc:146] Cre
```

```
2022-12-02 15:45:09.518742: I tensorflow/core/common_runtime/process_util.cc:146] Cre
```

```
seqModel.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accura
```

```
history = seqModel.fit(train_images, train_labels, batch_size=128, epochs=20, validation_da
```

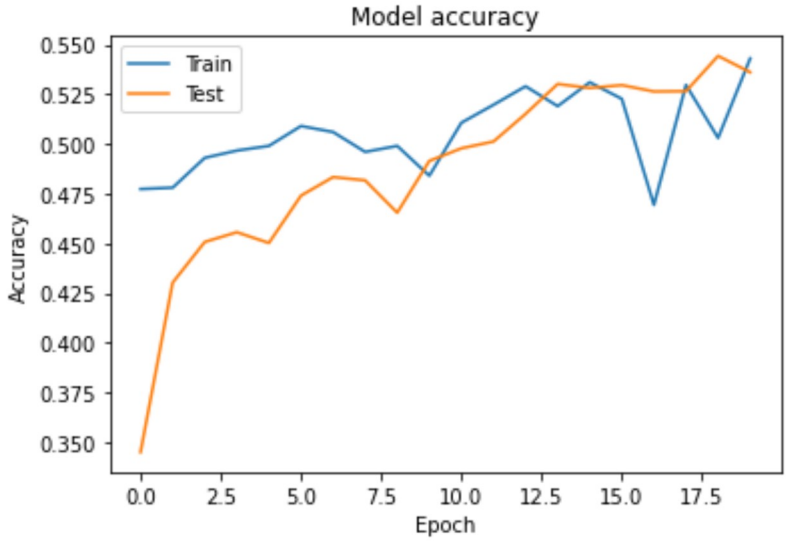
```
2022-12-02 15:45:09.668384: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.c
Epoch 1/20
110/110 [==============================] - 50s 445ms/step - loss: 5.6387 - accuracy:
Epoch 2/20
110/110 [==============================] - 48s 439ms/step - loss: 1.4159 - accuracy:
Epoch 3/20
110/110 [==============================] - 48s 433ms/step - loss: 1.3756 - accuracy:
Epoch 4/20
110/110 [==============================] - 48s 440ms/step - loss: 1.3843 - accuracy:
Epoch 5/20
110/110 [==============================] - 48s 437ms/step - loss: 1.3697 - accuracy:
Epoch 6/20
110/110 [==============================] - 48s 436ms/step - loss: 1.3113 - accuracy:
Epoch 7/20
110/110 [==============================] - 47s 427ms/step - loss: 1.3098 - accuracy:
Epoch 8/20
110/110 [==============================] - 47s 430ms/step - loss: 1.3138 - accuracy:
Epoch 9/20
110/110 [==============================] - 48s 436ms/step - loss: 1.3426 - accuracy:
Epoch 10/20
110/110 [==============================] - 47s 432ms/step - loss: 1.2790 - accuracy:
Epoch 11/20
110/110 [==============================] - 48s 436ms/step - loss: 1.2751 - accuracy:
Epoch 12/20
110/110 [==============================] - 47s 432ms/step - loss: 1.2711 - accuracy:
Epoch 13/20
110/110 [==============================] - 48s 436ms/step - loss: 1.2449 - accuracy:
Epoch 14/20
110/110 [==============================] - 47s 432ms/step - loss: 1.2148 - accuracy:
Epoch 15/20
110/110 [==============================] - 48s 439ms/step - loss: 1.1969 - accuracy:
Epoch 16/20
110/110 [==============================] - 47s 430ms/step - loss: 1.2083 - accuracy:
Epoch 17/20
110/110 [==============================] - 48s 436ms/step - loss: 1.2087 - accuracy:
Epoch 18/20
110/110 [==============================] - 47s 431ms/step - loss: 1.2158 - accuracy:
Epoch 19/20
110/110 [==============================] - 48s 436ms/step - loss: 1.1651 - accuracy:
Epoch 20/20
110/110 [==============================] - 47s 432ms/step - loss: 1.1809 - accuracy:
```

```
history.history.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
import matplotlib.pyplot as plt
```

```
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```




```
score = seqModel.evaluate(test_images, test_labels, verbose=0)
print('Test Loss: ', score[0])
print('Test Accuracy: ', score[1])
```

```
Test Loss:  1.2220392227172852
Test Accuracy:  0.5429999828338623
```

Accuray for the sequential newtwork isn't great, but considering there are six classes in this dataset it is better than guessing randomly.

CNN:

```
cnnModel = tf.keras.models.Sequential([
    tf.keras.Input(shape=(150,150,3)),
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2,2)),
    tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size=(2,2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(6, activation='softmax'),
])
```

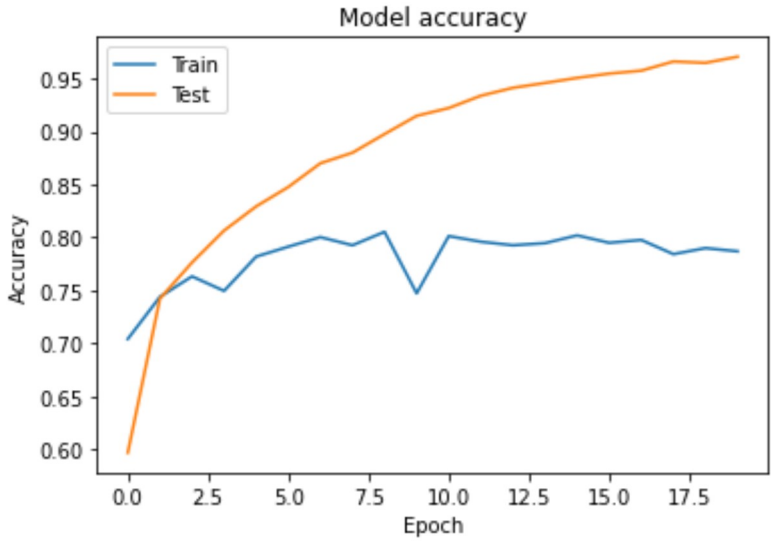
```
cnnModel.compile(loss='sparse_categorical_crossentropy',
                 optimizer='adam',
                 metrics=['accuracy'])

cnnHistory = cnnModel.fit(train_images, train_labels, batch_size=128, epochs=20, validatior
```

```
Epoch 1/20
110/110 [==============================] - 84s 754ms/step - loss: 1.0582 - accuracy:
Epoch 2/20
110/110 [==============================] - 81s 737ms/step - loss: 0.7264 - accuracy:
Epoch 3/20
110/110 [==============================] - 81s 738ms/step - loss: 0.6313 - accuracy:
Epoch 4/20
110/110 [==============================] - 81s 736ms/step - loss: 0.5542 - accuracy:
Epoch 5/20
110/110 [==============================] - 81s 737ms/step - loss: 0.5016 - accuracy:
Epoch 6/20
110/110 [==============================] - 80s 728ms/step - loss: 0.4460 - accuracy:
Epoch 7/20
110/110 [==============================] - 81s 738ms/step - loss: 0.3859 - accuracy:
Epoch 8/20
110/110 [==============================] - 81s 732ms/step - loss: 0.3455 - accuracy:
Epoch 9/20
110/110 [==============================] - 81s 738ms/step - loss: 0.3070 - accuracy:
Epoch 10/20
110/110 [==============================] - 80s 730ms/step - loss: 0.2605 - accuracy:
Epoch 11/20
110/110 [==============================] - 81s 732ms/step - loss: 0.2378 - accuracy:
Epoch 12/20
110/110 [==============================] - 80s 732ms/step - loss: 0.2002 - accuracy:
Epoch 13/20
110/110 [==============================] - 80s 728ms/step - loss: 0.1817 - accuracy:
Epoch 14/20
110/110 [==============================] - 80s 724ms/step - loss: 0.1622 - accuracy:
Epoch 15/20
110/110 [==============================] - 80s 729ms/step - loss: 0.1549 - accuracy:
Epoch 16/20
110/110 [==============================] - 80s 727ms/step - loss: 0.1398 - accuracy:
Epoch 17/20
110/110 [==============================] - 81s 733ms/step - loss: 0.1277 - accuracy:
Epoch 18/20
110/110 [==============================] - 79s 721ms/step - loss: 0.1086 - accuracy:
Epoch 19/20
110/110 [==============================] - 80s 724ms/step - loss: 0.1085 - accuracy:
Epoch 20/20
110/110 [==============================] - 80s 731ms/step - loss: 0.0971 - accuracy:
```

```
plt.plot(cnnHistory.history['val_accuracy'])
plt.plot(cnnHistory.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
```

```
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```




```
cnnScore = cnnModel.evaluate(test_images, test_labels, verbose=0)
print('Test Loss: ', cnnScore[0])
print('Test Accuracy: ', cnnScore[1])
```

```
Test Loss:  0.9063270092010498
Test Accuracy:  0.7870000004768372
```

Overall accuracy of the CNN is substantially better than the simple sequential network.

References:

[1] Liu, Vincent. *Intel Image Classification (CNN - Keras)* Link: https://www.kaggle.com/code/vincee/intel-image-classification-cnn-keras

I referenced Mr. Liu in reading the image dataset.

Colab paid products - Cancel contracts here