Image Classification with DL by Sujay Vadlakonda

Libraries and their versions

```
import tensorflow as tf
import matplotlib

print("TensorFlow version:", tf.__version__)
print("MatPlotLib version:", matplotlib.__version__)
```

```
TensorFlow version: 2.12.0
MatPlotLib version: 3.7.1
```

Step 1

Getting the data

Dataset Source: https://www.kaggle.com/datasets/puneet6060/intel-image-classification

Unzipping the data

```
from zipfile import ZipFile

file_name = "archive.zip"

with ZipFile(file_name, 'r') as zip:
    zip.extractall()
```

Divide into train/test

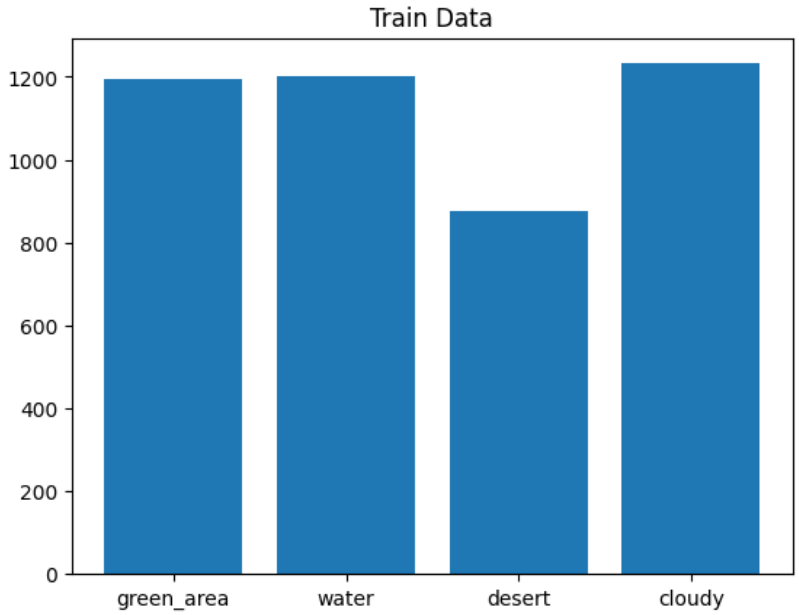
```
data_dir = "data"

data = tf.keras.utils.image_dataset_from_directory(
    data_dir,
)

train = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123
)

test = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123
)
```

```
Found 5631 files belonging to 4 classes.
Found 5631 files belonging to 4 classes.
Using 4505 files for training.
Found 5631 files belonging to 4 classes.
Using 1126 files for validation.
```

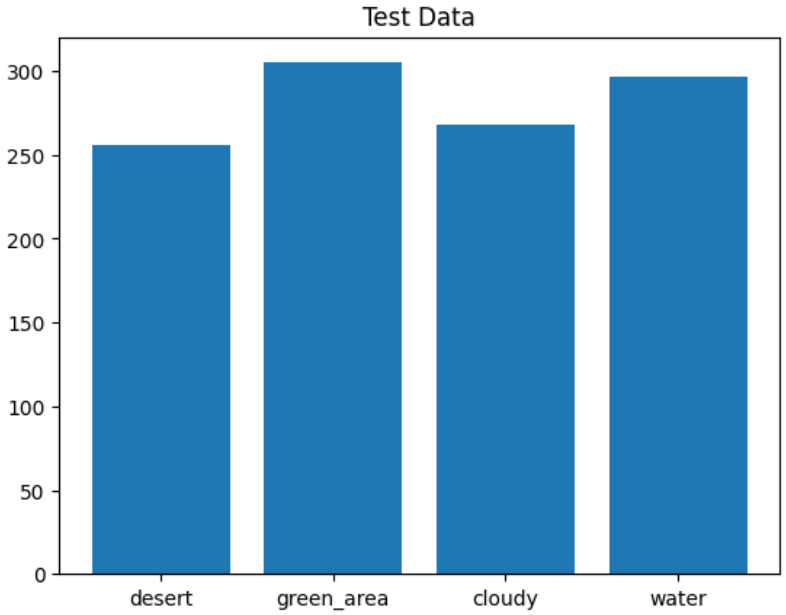
Create a graph showing the distribution of the target classes.

```
def map_increment(map, key):
  if key in map:
    map[key] += 1
  else:
    map[key] = 1

def count_dataset(dataset):
  count = {}
  labels = dataset.class_names

  for batch in dataset.as_numpy_iterator():
    images = batch[0]
    label_indices = batch[1]
    for label_index in label_indices:
      label = labels[label_index]
      map_increment(count, label)

  return count

def print_count(count, dataset_name):
  for label in count:
    c = count[label]
    print(c, "images are", label, "in", dataset_name)
  print()

def plot_count(class_counts, title):
  import matplotlib.pyplot as plt
  plt.title(label=title)
  plt.bar(range(len(class_counts)), list(class_counts.values()), align="center")
  plt.xticks(range(len(class_counts)), list(class_counts.keys()))
  plt.show()

def describe_dataset(dataset, name):
  count = count_dataset(dataset)
  print_count(count, name)
  plot_count(count, name)

describe_dataset(train, "Train Data")
describe_dataset(test, "Test Data")
describe_dataset(data, "All Data")
```

```
1195 images are green_area in Train Data
1203 images are water in Train Data
875 images are desert in Train Data
1232 images are cloudy in Train Data
```




```
256 images are desert in Test Data
305 images are green_area in Test Data
268 images are cloudy in Test Data
297 images are water in Test Data
```




```
1500 images are green_area in All Data
1500 images are water in All Data
1131 images are desert in All Data
1500 images are cloudy in All Data
```

Describe the data set and what the model should be able to predict.

The data set has a bunch of satellite images that are 256x256. The model should be able to predict whether the image shows clouds, water, desert, or a green area.

Step 2

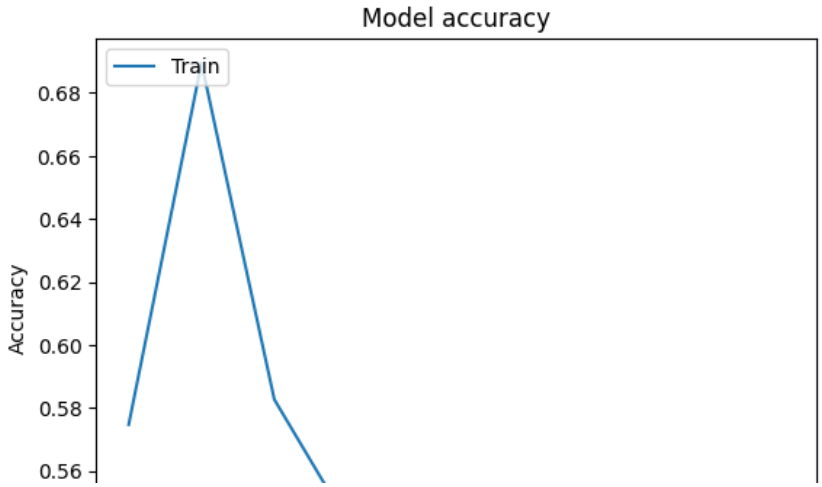
Create a sequential model

```
num_classes = len(train.class_names)

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(256, 256, 3)),
  tf.keras.layers.Dense(512, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(512, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(num_classes, activation='softmax'),
])


model.compile(optimizer='rmsprop',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
epochs=10

history = model.fit(
  train,
  validation_data=test,
  epochs=epochs
)
```

```
Epoch 1/10
/usr/local/lib/python3.9/dist-packages/keras/backend.py:5612: UserWarning: "`sparse_categorical_crossentropy` rec
  output, from_logits = _get_logits(
141/141 [==============================] - 11s 46ms/step - loss: 3561.0654 - accuracy: 0.5747 - val_loss: 877.160
Epoch 2/10
141/141 [==============================] - 8s 53ms/step - loss: 27.3804 - accuracy: 0.6895 - val_loss: 0.4785 - va
Epoch 3/10
141/141 [==============================] - 6s 44ms/step - loss: 33.8158 - accuracy: 0.5827 - val_loss: 0.8822 - va
Epoch 4/10
141/141 [==============================] - 7s 45ms/step - loss: 15.1595 - accuracy: 0.5438 - val_loss: 0.8390 - va
Epoch 5/10
141/141 [==============================] - 8s 54ms/step - loss: 0.7932 - accuracy: 0.5390 - val_loss: 0.8370 - val
Epoch 6/10
141/141 [==============================] - 7s 45ms/step - loss: 0.7924 - accuracy: 0.5425 - val_loss: 0.8363 - val
Epoch 7/10
141/141 [==============================] - 7s 47ms/step - loss: 0.7931 - accuracy: 0.5430 - val_loss: 0.8367 - val
Epoch 8/10
141/141 [==============================] - 9s 62ms/step - loss: 0.7927 - accuracy: 0.5345 - val_loss: 0.8363 - val
Epoch 9/10
141/141 [==============================] - 10s 67ms/step - loss: 0.7935 - accuracy: 0.5430 - val_loss: 0.8358 - va
Epoch 10/10
141/141 [==============================] - 9s 59ms/step - loss: 0.7925 - accuracy: 0.5392 - val_loss: 0.8372 - val
```

Graph

```
import matplotlib.pyplot as plt

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



Evaluate

```
score = model.evaluate(test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.8372419476509094
Test accuracy: 0.5017762184143066
```

Step 3

1st CNN Architecture

Model

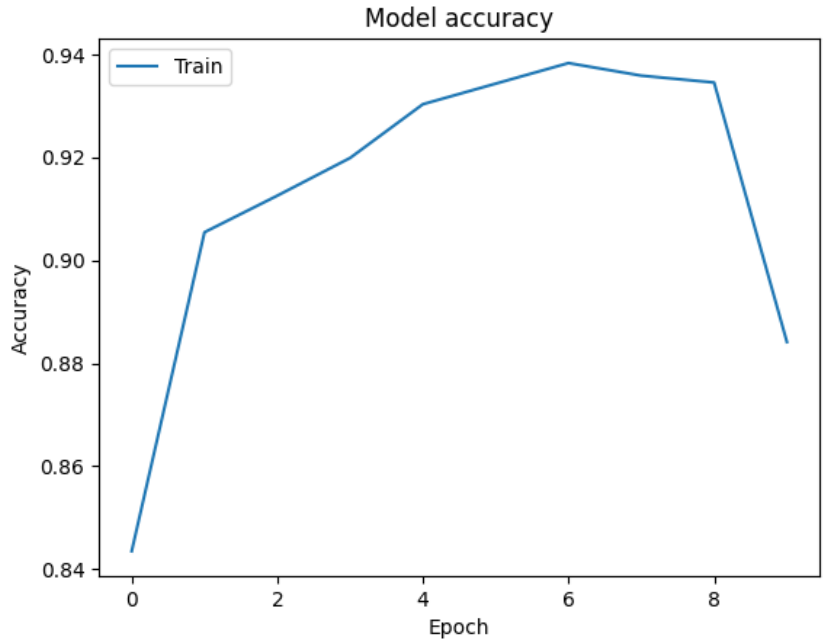
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Rescaling(1./255, input_shape=(256, 256, 3)),
  tf.keras.layers.Conv2D(16, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(num_classes)
])


model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train,
                    epochs=epochs,
                    verbose=1,
                    validation_data=test)
```

```
Epoch 1/10
141/141 [==============================] - 17s 70ms/step - loss: 0.3226 - accuracy: 0.8435 - val_loss: 0.1982 - va
Epoch 2/10
141/141 [==============================] - 9s 61ms/step - loss: 0.2069 - accuracy: 0.9054 - val_loss: 0.1651 - val
Epoch 3/10
141/141 [==============================] - 7s 51ms/step - loss: 0.1966 - accuracy: 0.9125 - val_loss: 0.1829 - val
```

```
Epoch 4/10
141/141 [==============================] - 9s 60ms/step - loss: 0.1836 - accuracy: 0.9199 - val_loss: 0.1616 - va
Epoch 5/10
141/141 [==============================] - 10s 67ms/step - loss: 0.1562 - accuracy: 0.9303 - val_loss: 0.1577 - v
Epoch 6/10
141/141 [==============================] - 7s 51ms/step - loss: 0.1599 - accuracy: 0.9343 - val_loss: 0.1712 - va
Epoch 7/10
141/141 [==============================] - 8s 57ms/step - loss: 0.1395 - accuracy: 0.9383 - val_loss: 0.1696 - va
Epoch 8/10
141/141 [==============================] - 8s 57ms/step - loss: 0.1692 - accuracy: 0.9358 - val_loss: 0.1759 - va
Epoch 9/10
141/141 [==============================] - 7s 50ms/step - loss: 0.1703 - accuracy: 0.9345 - val_loss: 0.5850 - va
Epoch 10/10
141/141 [==============================] - 9s 60ms/step - loss: 0.2438 - accuracy: 0.8841 - val_loss: 0.2300 - va
```

Graph

```
import matplotlib.pyplot as plt

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



Evaluate

```
score = model.evaluate(test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.23002515733242035
Test accuracy: 0.8765541911125183
```

2nd CNN Architecture

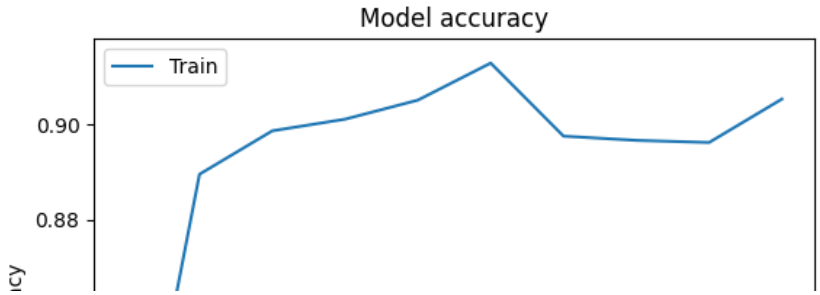
Model

```
model = tf.keras.models.Sequential([
        tf.keras.Input(shape=(256, 256, 3)),
        tf.keras.layers.Conv2D(32, 3, padding="same", activation="relu"),
        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Conv2D(64, 3, padding="same", activation="relu"),
        tf.keras.layers.MaxPooling2D(),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(num_classes, activation="softmax"),
])

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train,
                    epochs=epochs,
                    verbose=1,
                    validation_data=test)
```

```
Epoch 1/10
/usr/local/lib/python3.9/dist-packages/keras/backend.py:5612: UserWarning: "`sparse_categorical_crossentropy` rece
  output, from_logits = _get_logits(
141/141 [==============================] - 13s 78ms/step - loss: 17.6845 - accuracy: 0.8104 - val_loss: 0.2633 - v
Epoch 2/10
141/141 [==============================] - 9s 63ms/step - loss: 0.2509 - accuracy: 0.8895 - val_loss: 0.2294 - val
Epoch 3/10
141/141 [==============================] - 9s 65ms/step - loss: 0.2359 - accuracy: 0.8986 - val_loss: 0.2274 - val
Epoch 4/10
141/141 [==============================] - 9s 66ms/step - loss: 0.2320 - accuracy: 0.9010 - val_loss: 0.2390 - val
Epoch 5/10
141/141 [==============================] - 10s 71ms/step - loss: 0.2220 - accuracy: 0.9050 - val_loss: 0.2157 - va
Epoch 6/10
141/141 [==============================] - 9s 62ms/step - loss: 0.2097 - accuracy: 0.9128 - val_loss: 0.2147 - val
Epoch 7/10
141/141 [==============================] - 10s 72ms/step - loss: 0.2443 - accuracy: 0.8974 - val_loss: 0.3192 - va
Epoch 8/10
141/141 [==============================] - 10s 67ms/step - loss: 0.2255 - accuracy: 0.8966 - val_loss: 0.2332 - va
Epoch 9/10
141/141 [==============================] - 10s 67ms/step - loss: 0.2468 - accuracy: 0.8961 - val_loss: 0.2318 - va
Epoch 10/10
141/141 [==============================] - 9s 62ms/step - loss: 0.2138 - accuracy: 0.9052 - val_loss: 0.2166 - val
```

Graph

```
import matplotlib.pyplot as plt

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



Evaluate

```
score = model.evaluate(test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.21661590039730072
Test accuracy: 0.9085257649421692
```

Step 4

Model

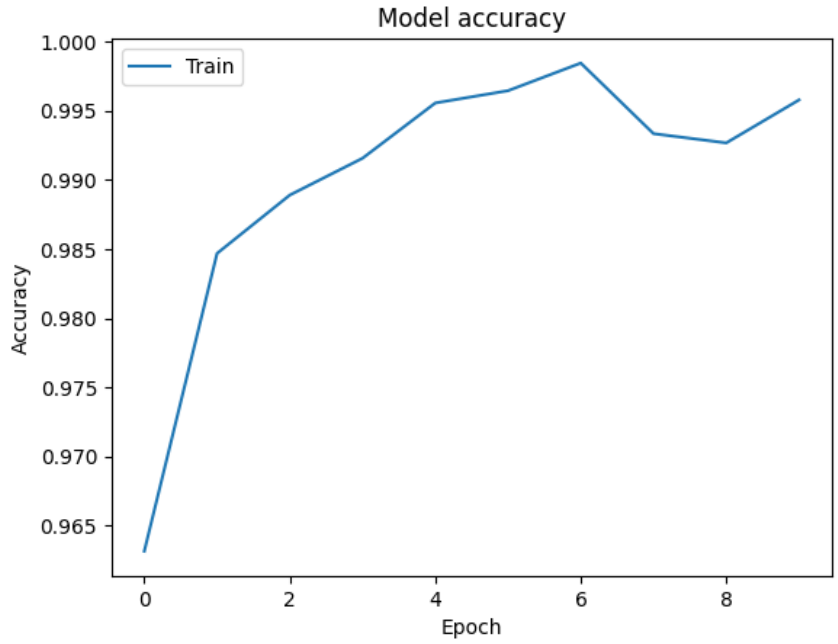
```
base_model = tf.keras.applications.MobileNetV2(input_shape=(256,256,3),
                                               include_top=False,
                                               weights='imagenet')
base_model.trainable = False
inputs = tf.keras.Input(shape=(256, 256, 3))

x = tf.keras.layers.Flatten()(base_model(inputs))
y = tf.keras.layers.Dense(128, activation = "relu")(x)
z = tf.keras.layers.Dense(64, activation = "relu")(y)
outputs = tf.keras.layers.Dense(6, activation='softmax')(z)
model = tf.keras.Model(inputs, outputs)

model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train,
                    epochs=epochs,
                    verbose=1,
                    validation_data=test)
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights
Epoch 1/10
/usr/local/lib/python3.9/dist-packages/keras/backend.py:5612: UserWarning: "`sparse_categorical_crossentropy` rec
  output, from_logits = _get_logits(
141/141 [==============================] - 20s 106ms/step - loss: 0.4713 - accuracy: 0.9632 - val_loss: 0.0495 - v
Epoch 2/10
141/141 [==============================] - 12s 80ms/step - loss: 0.1415 - accuracy: 0.9847 - val_loss: 0.4200 - va
Epoch 3/10
141/141 [==============================] - 11s 75ms/step - loss: 0.1014 - accuracy: 0.9889 - val_loss: 0.0474 - va
Epoch 4/10
141/141 [==============================] - 11s 75ms/step - loss: 0.0749 - accuracy: 0.9916 - val_loss: 0.1280 - va
Epoch 5/10
141/141 [==============================] - 11s 76ms/step - loss: 0.0284 - accuracy: 0.9956 - val_loss: 0.0380 - va
Epoch 6/10
141/141 [==============================] - 11s 74ms/step - loss: 0.0196 - accuracy: 0.9964 - val_loss: 0.0422 - va
Epoch 7/10
141/141 [==============================] - 11s 73ms/step - loss: 0.0085 - accuracy: 0.9984 - val_loss: 0.0867 - va
Epoch 8/10
141/141 [==============================] - 11s 75ms/step - loss: 0.0419 - accuracy: 0.9933 - val_loss: 0.1551 - va
Epoch 9/10
141/141 [==============================] - 11s 74ms/step - loss: 0.0787 - accuracy: 0.9927 - val_loss: 0.0301 - va
```

```
Epoch 10/10
141/141 [==============================] - 11s 75ms/step - loss: 0.0252 - accuracy: 0.9958 - val_loss: 0.0458 - va
```

Graph

```
import matplotlib.pyplot as plt

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



Evaluate

```
score = model.evaluate(test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.04580029472708702
Test accuracy: 0.9946714043617249
```

Step 5

Analyze the performance of various approaches

The pretrained model had an accuracy of 0.99. The 2nd CNN had an accuracy of 0.91. The 1st CNN had an accuracy of 0.88. The sequential model had an accuracy of 0.50. Clearly the CNNs are more capable than the sequential neural network. This is probably because of CNNs' ability to learn features and apply them in multiple places. The second CNN had a slighly higher accuracy, probably because the dropout layer prevented overfitting to the training data. Finally the pretrained model had the highest accuracy because the pretrained model is designed to have a super high base line of image feature detection and was crafted by teams of experts.