exp-cnn-63130500042

November 28, 2023

[1]:
```
from tensorflow.keras.utils import image_dataset_from_directory

import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A
NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy
(detected version 1.24.3
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

[2]:
```
RAND_STATE_VALUE = 42
DATA_TEST_PATH = "/kaggle/input/landuse-scene-classification/
 ↪images_train_test_val/test"
DATA_TRAIN_PATH = "/kaggle/input/landuse-scene-classification/
 ↪images_train_test_val/train"
DATA_VAL_PATH = "/kaggle/input/landuse-scene-classification/
 ↪images_train_test_val/validation"
IMG_SIZE = (256, 256)
IMG_SHAPE = (256, 256, 3)
LABEL_MODE = "categorical"
```

1 Load Data

[3]:
```
train_ds = image_dataset_from_directory(directory = DATA_TRAIN_PATH, label_mode␣
 ↪= LABEL_MODE, image_size = IMG_SIZE)
test_ds = image_dataset_from_directory(directory = DATA_TEST_PATH, label_mode =␣
 ↪LABEL_MODE, image_size = IMG_SIZE)
val_ds = image_dataset_from_directory(directory = DATA_VAL_PATH, label_mode =␣
 ↪LABEL_MODE, image_size = IMG_SIZE)
```

```
Found 7350 files belonging to 21 classes.
Found 1050 files belonging to 21 classes.
Found 2100 files belonging to 21 classes.
```

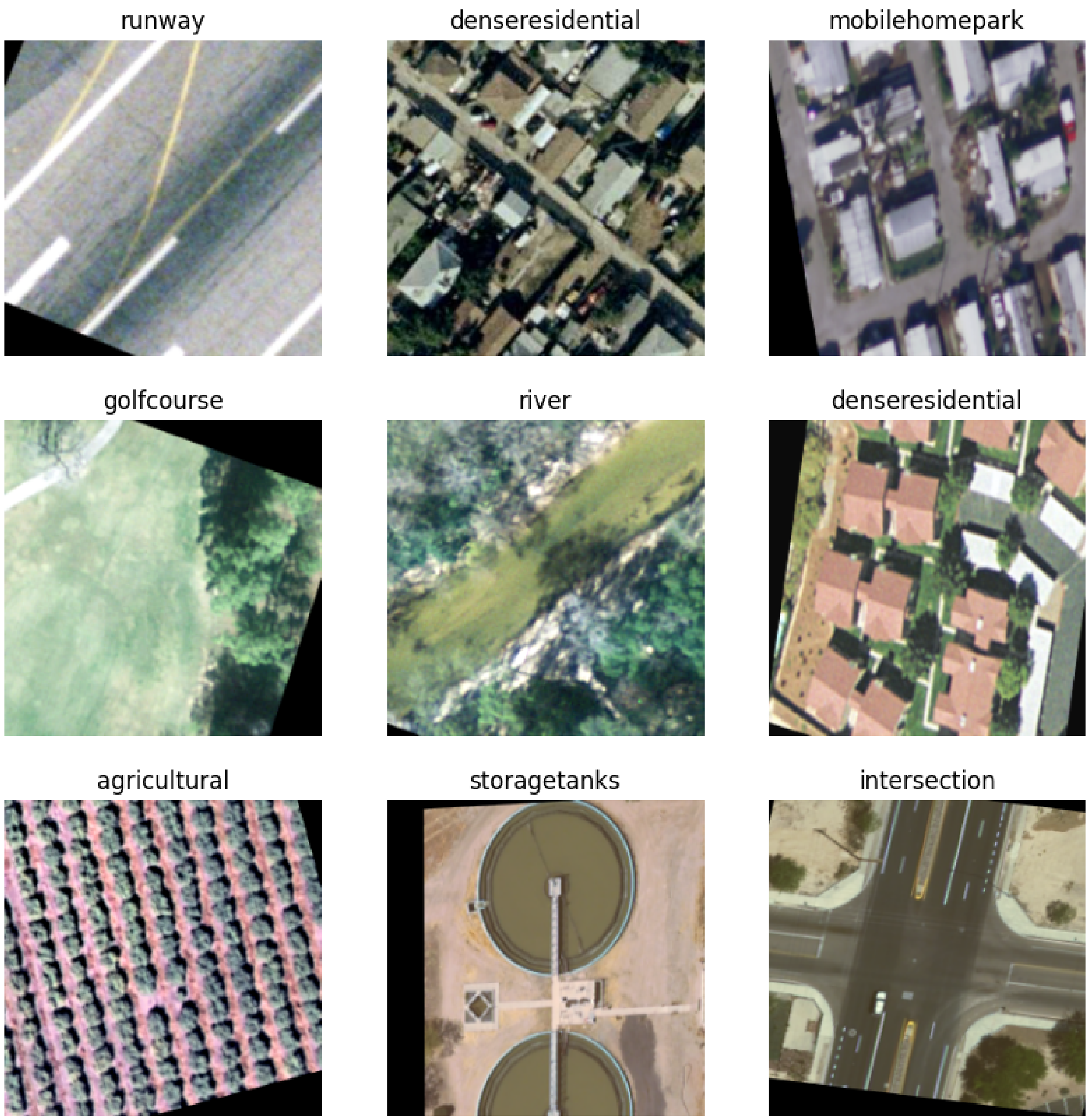
[4]:
```
for image_batch, labels_batch in train_ds:
    print(image_batch.shape)
    print(labels_batch.shape)
    break
```

```
(32, 256, 256, 3)
(32, 21)
```

[5]:
```
class_names = train_ds.class_names
class_names
```

[5]:
```
['agricultural',
 'airplane',
 'baseballdiamond',
 'beach',
 'buildings',
 'chaparral',
 'denseresidential',
 'forest',
 'freeway',
 'golfcourse',
 'harbor',
 'intersection',
 'mediumresidential',
 'mobilehomepark',
 'overpass',
 'parkinglot',
 'river',
 'runway',
 'sparseresidential',
 'storagetanks',
 'tenniscourt']
```

[6]:
```
plt.figure(figsize = (10, 10))
for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[np.argmax(labels[i])])
        plt.axis("off")
    plt.show()
```



2 Model Creation

[7]:
```
from tensorflow.keras import Sequential
from tensorflow.keras import layers
from tensorflow.keras.regularizers import l2
```

[8]:
```
data_augmentation = Sequential([
    layers.RandomFlip("horizontal", input_shape = IMG_SHAPE),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1)
])
```

[9]:
```
model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255),
    layers.Conv2D(32, (3, 3), activation = "relu", kernel_regularizer = l2(0.
↪0001)),
    layers.MaxPooling2D(),
    layers.BatchNormalization(),
    layers.Conv2D(64, (3, 3), activation = "relu", kernel_regularizer = l2(0.
↪0001)),
    layers.MaxPooling2D(),
    layers.BatchNormalization(),
    layers.Conv2D(128, (3, 3), activation = "relu", kernel_regularizer = l2(0.
↪0001)),
    layers.MaxPooling2D(),
    layers.BatchNormalization(),
    layers.Dropout(0.25),
    layers.Flatten(),
    layers.Dense(256, activation = "relu"),
    layers.Dense(len(train_ds.class_names), activation = "softmax")
])
```

[10]:
```
model.compile(optimizer = "adam", loss = "categorical_crossentropy", metrics =␣
↪["accuracy"])
```

[11]:
```
model.summary()
```

```
Model: "sequential_1"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential (Sequential)     (None, 256, 256, 3)       0

 rescaling (Rescaling)       (None, 256, 256, 3)       0

 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2  (None, 127, 127, 32)      0
 D)

 batch_normalization (Batch  (None, 127, 127, 32)      128
 Normalization)

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 64)        0
 g2D)
```

```
 batch_normalization_1 (Bat  (None, 62, 62, 64)        256
 chNormalization)

 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 30, 30, 128)       0
 g2D)

 batch_normalization_2 (Bat  (None, 30, 30, 128)       512
 chNormalization)

 dropout (Dropout)           (None, 30, 30, 128)       0

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 256)               29491456

 dense_1 (Dense)             (None, 21)                5397

=================================================================
Total params: 29590997 (112.88 MB)
Trainable params: 29590549 (112.88 MB)
Non-trainable params: 448 (1.75 KB)
_________________________________________________________________
```

[12]:
```python
history = model.fit(train_ds, epochs = 100, validation_data = val_ds, verbose =
 ↪1)
```

```
Epoch 1/100

2023-11-27 22:17:29.409536: E
tensorflow/core/grappler/optimizers/meta_optimizer.cc:954] layout failed:
INVALID_ARGUMENT: Size of values 0 does not match size of permutation 4 @ fanin
shape insequential_1/dropout/dropout/SelectV2-2-TransposeNHWCToNCHW-
LayoutOptimizer

230/230 [==============================] - 26s 86ms/step - loss: 16.5802 -
accuracy: 0.1419 - val_loss: 13.3509 - val_accuracy: 0.0571
Epoch 2/100
230/230 [==============================] - 19s 80ms/step - loss: 4.9771 -
accuracy: 0.1444 - val_loss: 3.3783 - val_accuracy: 0.1286
Epoch 3/100
230/230 [==============================] - 19s 80ms/step - loss: 3.1955 -
accuracy: 0.1565 - val_loss: 2.7758 - val_accuracy: 0.1471
Epoch 4/100
230/230 [==============================] - 19s 81ms/step - loss: 2.7289 -
accuracy: 0.2033 - val_loss: 3.1069 - val_accuracy: 0.2033
Epoch 5/100
230/230 [==============================] - 19s 80ms/step - loss: 2.5577 -
```

```
accuracy: 0.2135 - val_loss: 2.8194 - val_accuracy: 0.2167
Epoch 6/100
230/230 [==============================] - 19s 81ms/step - loss: 2.5004 -
accuracy: 0.2297 - val_loss: 4.0873 - val_accuracy: 0.1810
Epoch 7/100
230/230 [==============================] - 19s 81ms/step - loss: 2.4138 -
accuracy: 0.2427 - val_loss: 2.6495 - val_accuracy: 0.2748
Epoch 8/100
230/230 [==============================] - 19s 81ms/step - loss: 2.3582 -
accuracy: 0.2620 - val_loss: 2.4668 - val_accuracy: 0.2067
Epoch 9/100
230/230 [==============================] - 19s 82ms/step - loss: 2.3044 -
accuracy: 0.2683 - val_loss: 2.5779 - val_accuracy: 0.2243
Epoch 10/100
230/230 [==============================] - 19s 80ms/step - loss: 2.3033 -
accuracy: 0.2770 - val_loss: 3.0955 - val_accuracy: 0.1943
Epoch 11/100
230/230 [==============================] - 19s 81ms/step - loss: 2.2629 -
accuracy: 0.2882 - val_loss: 2.6829 - val_accuracy: 0.2052
Epoch 12/100
230/230 [==============================] - 19s 80ms/step - loss: 2.2153 -
accuracy: 0.3014 - val_loss: 2.4648 - val_accuracy: 0.2657
Epoch 13/100
230/230 [==============================] - 19s 80ms/step - loss: 2.1489 -
accuracy: 0.3190 - val_loss: 2.2258 - val_accuracy: 0.3071
Epoch 14/100
230/230 [==============================] - 19s 81ms/step - loss: 2.1002 -
accuracy: 0.3328 - val_loss: 2.4760 - val_accuracy: 0.3052
Epoch 15/100
230/230 [==============================] - 19s 80ms/step - loss: 2.1073 -
accuracy: 0.3407 - val_loss: 3.7626 - val_accuracy: 0.2619
Epoch 16/100
230/230 [==============================] - 19s 80ms/step - loss: 2.0555 -
accuracy: 0.3457 - val_loss: 2.2917 - val_accuracy: 0.3367
Epoch 17/100
230/230 [==============================] - 19s 80ms/step - loss: 2.0457 -
accuracy: 0.3584 - val_loss: 2.0553 - val_accuracy: 0.3476
Epoch 18/100
230/230 [==============================] - 19s 80ms/step - loss: 1.9973 -
accuracy: 0.3686 - val_loss: 2.0705 - val_accuracy: 0.3667
Epoch 19/100
230/230 [==============================] - 19s 81ms/step - loss: 1.9695 -
accuracy: 0.3737 - val_loss: 2.2393 - val_accuracy: 0.3619
Epoch 20/100
230/230 [==============================] - 19s 80ms/step - loss: 1.9480 -
accuracy: 0.3795 - val_loss: 1.9986 - val_accuracy: 0.3652
Epoch 21/100
230/230 [==============================] - 19s 81ms/step - loss: 1.9281 -
```

```
accuracy: 0.3893 - val_loss: 3.4114 - val_accuracy: 0.2814
Epoch 22/100
230/230 [==============================] - 19s 80ms/step - loss: 1.9177 -
accuracy: 0.4018 - val_loss: 1.9023 - val_accuracy: 0.4081
Epoch 23/100
230/230 [==============================] - 19s 80ms/step - loss: 1.8712 -
accuracy: 0.4177 - val_loss: 2.1609 - val_accuracy: 0.3719
Epoch 24/100
230/230 [==============================] - 19s 81ms/step - loss: 1.8374 -
accuracy: 0.4186 - val_loss: 1.8434 - val_accuracy: 0.4295
Epoch 25/100
230/230 [==============================] - 19s 80ms/step - loss: 1.8069 -
accuracy: 0.4246 - val_loss: 2.5232 - val_accuracy: 0.3067
Epoch 26/100
230/230 [==============================] - 19s 80ms/step - loss: 1.8157 -
accuracy: 0.4244 - val_loss: 1.9525 - val_accuracy: 0.4229
Epoch 27/100
230/230 [==============================] - 19s 81ms/step - loss: 1.7743 -
accuracy: 0.4371 - val_loss: 1.9757 - val_accuracy: 0.4390
Epoch 28/100
230/230 [==============================] - 19s 80ms/step - loss: 1.7348 -
accuracy: 0.4537 - val_loss: 2.4870 - val_accuracy: 0.3729
Epoch 29/100
230/230 [==============================] - 19s 80ms/step - loss: 1.7664 -
accuracy: 0.4437 - val_loss: 1.9859 - val_accuracy: 0.4090
Epoch 30/100
230/230 [==============================] - 19s 80ms/step - loss: 1.7457 -
accuracy: 0.4521 - val_loss: 1.8777 - val_accuracy: 0.4386
Epoch 31/100
230/230 [==============================] - 19s 81ms/step - loss: 1.6833 -
accuracy: 0.4735 - val_loss: 1.8890 - val_accuracy: 0.4605
Epoch 32/100
230/230 [==============================] - 19s 80ms/step - loss: 1.6808 -
accuracy: 0.4720 - val_loss: 1.9725 - val_accuracy: 0.4286
Epoch 33/100
230/230 [==============================] - 19s 80ms/step - loss: 1.6810 -
accuracy: 0.4799 - val_loss: 1.8703 - val_accuracy: 0.4405
Epoch 34/100
230/230 [==============================] - 19s 80ms/step - loss: 1.6508 -
accuracy: 0.4838 - val_loss: 1.8363 - val_accuracy: 0.4657
Epoch 35/100
230/230 [==============================] - 19s 80ms/step - loss: 1.6338 -
accuracy: 0.4936 - val_loss: 1.7103 - val_accuracy: 0.4900
Epoch 36/100
230/230 [==============================] - 19s 81ms/step - loss: 1.5897 -
accuracy: 0.5026 - val_loss: 2.1301 - val_accuracy: 0.4129
Epoch 37/100
230/230 [==============================] - 19s 81ms/step - loss: 1.5871 -
```

```
accuracy: 0.5050 - val_loss: 2.0472 - val_accuracy: 0.4219
Epoch 38/100
230/230 [==============================] - 19s 80ms/step - loss: 1.5517 -
accuracy: 0.5261 - val_loss: 1.7969 - val_accuracy: 0.4838
Epoch 39/100
230/230 [==============================] - 19s 80ms/step - loss: 1.5688 -
accuracy: 0.5180 - val_loss: 1.7578 - val_accuracy: 0.5005
Epoch 40/100
230/230 [==============================] - 19s 81ms/step - loss: 1.5079 -
accuracy: 0.5378 - val_loss: 1.8278 - val_accuracy: 0.4790
Epoch 41/100
230/230 [==============================] - 19s 81ms/step - loss: 1.5087 -
accuracy: 0.5322 - val_loss: 1.9791 - val_accuracy: 0.4576
Epoch 42/100
230/230 [==============================] - 19s 80ms/step - loss: 1.5097 -
accuracy: 0.5405 - val_loss: 1.8369 - val_accuracy: 0.4576
Epoch 43/100
230/230 [==============================] - 19s 80ms/step - loss: 1.5324 -
accuracy: 0.5332 - val_loss: 2.0184 - val_accuracy: 0.4733
Epoch 44/100
230/230 [==============================] - 19s 81ms/step - loss: 1.4948 -
accuracy: 0.5423 - val_loss: 2.3227 - val_accuracy: 0.3995
Epoch 45/100
230/230 [==============================] - 19s 80ms/step - loss: 1.4702 -
accuracy: 0.5491 - val_loss: 2.0370 - val_accuracy: 0.4724
Epoch 46/100
230/230 [==============================] - 19s 81ms/step - loss: 1.4725 -
accuracy: 0.5548 - val_loss: 1.6260 - val_accuracy: 0.5424
Epoch 47/100
230/230 [==============================] - 19s 80ms/step - loss: 1.4461 -
accuracy: 0.5604 - val_loss: 1.8315 - val_accuracy: 0.4867
Epoch 48/100
230/230 [==============================] - 19s 81ms/step - loss: 1.4204 -
accuracy: 0.5676 - val_loss: 1.6233 - val_accuracy: 0.5281
Epoch 49/100
230/230 [==============================] - 19s 80ms/step - loss: 1.4462 -
accuracy: 0.5599 - val_loss: 1.7576 - val_accuracy: 0.5152
Epoch 50/100
230/230 [==============================] - 19s 81ms/step - loss: 1.4149 -
accuracy: 0.5737 - val_loss: 1.6091 - val_accuracy: 0.5290
Epoch 51/100
230/230 [==============================] - 19s 81ms/step - loss: 1.3780 -
accuracy: 0.5810 - val_loss: 1.6019 - val_accuracy: 0.5590
Epoch 52/100
230/230 [==============================] - 19s 82ms/step - loss: 1.3716 -
accuracy: 0.5837 - val_loss: 1.4176 - val_accuracy: 0.5814
Epoch 53/100
230/230 [==============================] - 19s 81ms/step - loss: 1.3631 -
```

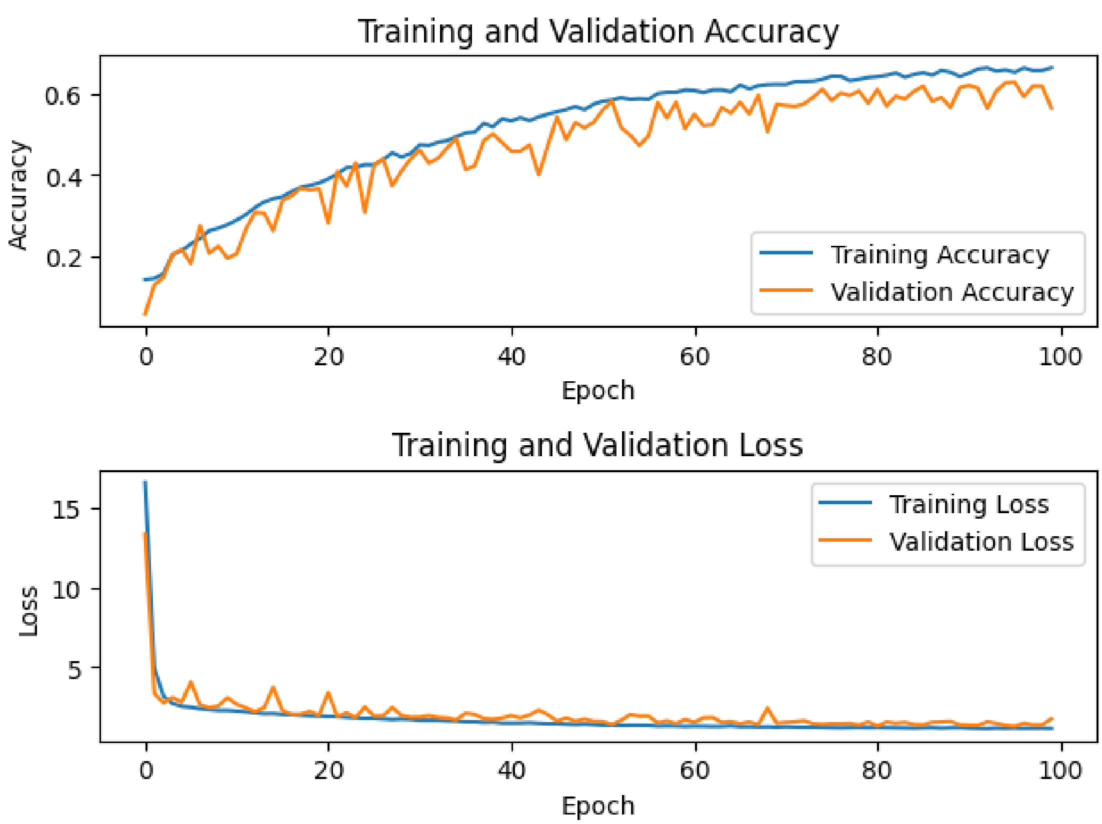
```
accuracy: 0.5895 - val_loss: 1.6844 - val_accuracy: 0.5162
Epoch 54/100
230/230 [==============================] - 19s 80ms/step - loss: 1.3702 -
accuracy: 0.5853 - val_loss: 2.0403 - val_accuracy: 0.4971
Epoch 55/100
230/230 [==============================] - 19s 80ms/step - loss: 1.3677 -
accuracy: 0.5867 - val_loss: 1.9503 - val_accuracy: 0.4719
Epoch 56/100
230/230 [==============================] - 19s 81ms/step - loss: 1.3628 -
accuracy: 0.5856 - val_loss: 1.9504 - val_accuracy: 0.4962
Epoch 57/100
230/230 [==============================] - 19s 81ms/step - loss: 1.3249 -
accuracy: 0.5999 - val_loss: 1.5258 - val_accuracy: 0.5771
Epoch 58/100
230/230 [==============================] - 19s 80ms/step - loss: 1.3354 -
accuracy: 0.6027 - val_loss: 1.6401 - val_accuracy: 0.5395
Epoch 59/100
230/230 [==============================] - 19s 81ms/step - loss: 1.3301 -
accuracy: 0.6029 - val_loss: 1.4793 - val_accuracy: 0.5786
Epoch 60/100
230/230 [==============================] - 19s 81ms/step - loss: 1.3041 -
accuracy: 0.6084 - val_loss: 1.7342 - val_accuracy: 0.5133
Epoch 61/100
230/230 [==============================] - 19s 80ms/step - loss: 1.3197 -
accuracy: 0.6073 - val_loss: 1.5512 - val_accuracy: 0.5490
Epoch 62/100
230/230 [==============================] - 19s 81ms/step - loss: 1.3130 -
accuracy: 0.6020 - val_loss: 1.8412 - val_accuracy: 0.5205
Epoch 63/100
230/230 [==============================] - 19s 81ms/step - loss: 1.2990 -
accuracy: 0.6086 - val_loss: 1.8606 - val_accuracy: 0.5233
Epoch 64/100
230/230 [==============================] - 19s 80ms/step - loss: 1.3048 -
accuracy: 0.6087 - val_loss: 1.5594 - val_accuracy: 0.5652
Epoch 65/100
230/230 [==============================] - 19s 80ms/step - loss: 1.3411 -
accuracy: 0.6042 - val_loss: 1.6015 - val_accuracy: 0.5514
Epoch 66/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2824 -
accuracy: 0.6205 - val_loss: 1.4710 - val_accuracy: 0.5786
Epoch 67/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2872 -
accuracy: 0.6110 - val_loss: 1.5829 - val_accuracy: 0.5495
Epoch 68/100
230/230 [==============================] - 19s 82ms/step - loss: 1.2675 -
accuracy: 0.6190 - val_loss: 1.4142 - val_accuracy: 0.5957
Epoch 69/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2748 -
```

```
accuracy: 0.6216 - val_loss: 2.4479 - val_accuracy: 0.5057
Epoch 70/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2555 -
accuracy: 0.6223 - val_loss: 1.5017 - val_accuracy: 0.5733
Epoch 71/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2776 -
accuracy: 0.6219 - val_loss: 1.5575 - val_accuracy: 0.5705
Epoch 72/100
230/230 [==============================] - 19s 81ms/step - loss: 1.2611 -
accuracy: 0.6291 - val_loss: 1.5925 - val_accuracy: 0.5681
Epoch 73/100
230/230 [==============================] - 19s 82ms/step - loss: 1.2529 -
accuracy: 0.6294 - val_loss: 1.6489 - val_accuracy: 0.5743
Epoch 74/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2491 -
accuracy: 0.6303 - val_loss: 1.4689 - val_accuracy: 0.5910
Epoch 75/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2379 -
accuracy: 0.6348 - val_loss: 1.4342 - val_accuracy: 0.6105
Epoch 76/100
230/230 [==============================] - 19s 81ms/step - loss: 1.2254 -
accuracy: 0.6427 - val_loss: 1.4764 - val_accuracy: 0.5824
Epoch 77/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2148 -
accuracy: 0.6422 - val_loss: 1.4682 - val_accuracy: 0.6010
Epoch 78/100
230/230 [==============================] - 19s 82ms/step - loss: 1.2293 -
accuracy: 0.6314 - val_loss: 1.4897 - val_accuracy: 0.5952
Epoch 79/100
230/230 [==============================] - 19s 81ms/step - loss: 1.2380 -
accuracy: 0.6348 - val_loss: 1.3853 - val_accuracy: 0.6062
Epoch 80/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2240 -
accuracy: 0.6393 - val_loss: 1.5829 - val_accuracy: 0.5757
Epoch 81/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2363 -
accuracy: 0.6415 - val_loss: 1.3228 - val_accuracy: 0.6100
Epoch 82/100
230/230 [==============================] - 19s 81ms/step - loss: 1.2218 -
accuracy: 0.6446 - val_loss: 1.5766 - val_accuracy: 0.5690
Epoch 83/100
230/230 [==============================] - 19s 81ms/step - loss: 1.2140 -
accuracy: 0.6502 - val_loss: 1.4977 - val_accuracy: 0.5933
Epoch 84/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2050 -
accuracy: 0.6403 - val_loss: 1.5543 - val_accuracy: 0.5867
Epoch 85/100
230/230 [==============================] - 19s 80ms/step - loss: 1.1916 -
```

```
accuracy: 0.6465 - val_loss: 1.4223 - val_accuracy: 0.6052
Epoch 86/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2145 -
accuracy: 0.6510 - val_loss: 1.3968 - val_accuracy: 0.6176
Epoch 87/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2229 -
accuracy: 0.6449 - val_loss: 1.5621 - val_accuracy: 0.5810
Epoch 88/100
230/230 [==============================] - 19s 80ms/step - loss: 1.1907 -
accuracy: 0.6567 - val_loss: 1.5821 - val_accuracy: 0.5900
Epoch 89/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2149 -
accuracy: 0.6521 - val_loss: 1.6131 - val_accuracy: 0.5652
Epoch 90/100
230/230 [==============================] - 19s 80ms/step - loss: 1.2228 -
accuracy: 0.6416 - val_loss: 1.3954 - val_accuracy: 0.6143
Epoch 91/100
230/230 [==============================] - 19s 80ms/step - loss: 1.1908 -
accuracy: 0.6498 - val_loss: 1.3755 - val_accuracy: 0.6195
Epoch 92/100
230/230 [==============================] - 19s 80ms/step - loss: 1.1762 -
accuracy: 0.6604 - val_loss: 1.3655 - val_accuracy: 0.6129
Epoch 93/100
230/230 [==============================] - 19s 80ms/step - loss: 1.1612 -
accuracy: 0.6631 - val_loss: 1.5934 - val_accuracy: 0.5633
Epoch 94/100
230/230 [==============================] - 19s 81ms/step - loss: 1.1908 -
accuracy: 0.6547 - val_loss: 1.4892 - val_accuracy: 0.6062
Epoch 95/100
230/230 [==============================] - 19s 81ms/step - loss: 1.1781 -
accuracy: 0.6578 - val_loss: 1.3628 - val_accuracy: 0.6262
Epoch 96/100
230/230 [==============================] - 19s 81ms/step - loss: 1.1916 -
accuracy: 0.6520 - val_loss: 1.3125 - val_accuracy: 0.6276
Epoch 97/100
230/230 [==============================] - 19s 81ms/step - loss: 1.1753 -
accuracy: 0.6630 - val_loss: 1.5007 - val_accuracy: 0.5924
Epoch 98/100
230/230 [==============================] - 19s 80ms/step - loss: 1.1821 -
accuracy: 0.6562 - val_loss: 1.3931 - val_accuracy: 0.6181
Epoch 99/100
230/230 [==============================] - 19s 80ms/step - loss: 1.1852 -
accuracy: 0.6565 - val_loss: 1.3974 - val_accuracy: 0.6176
Epoch 100/100
230/230 [==============================] - 19s 81ms/step - loss: 1.1713 -
accuracy: 0.6634 - val_loss: 1.7742 - val_accuracy: 0.5643
```

[13]:

```
# Plot training history for accuracy
plt.subplot(2, 1, 1)
plt.plot(history.history["accuracy"], label = "Training Accuracy")
plt.plot(history.history["val_accuracy"], label = "Validation Accuracy")
plt.title("Training and Validation Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()

# Plot training history for loss
plt.subplot(2, 1, 2)
plt.plot(history.history["loss"], label = "Training Loss")
plt.plot(history.history["val_loss"], label = "Validation Loss")
plt.title("Training and Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.tight_layout()
plt.show()
```



3 Model Testing and Evaluation

[14]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

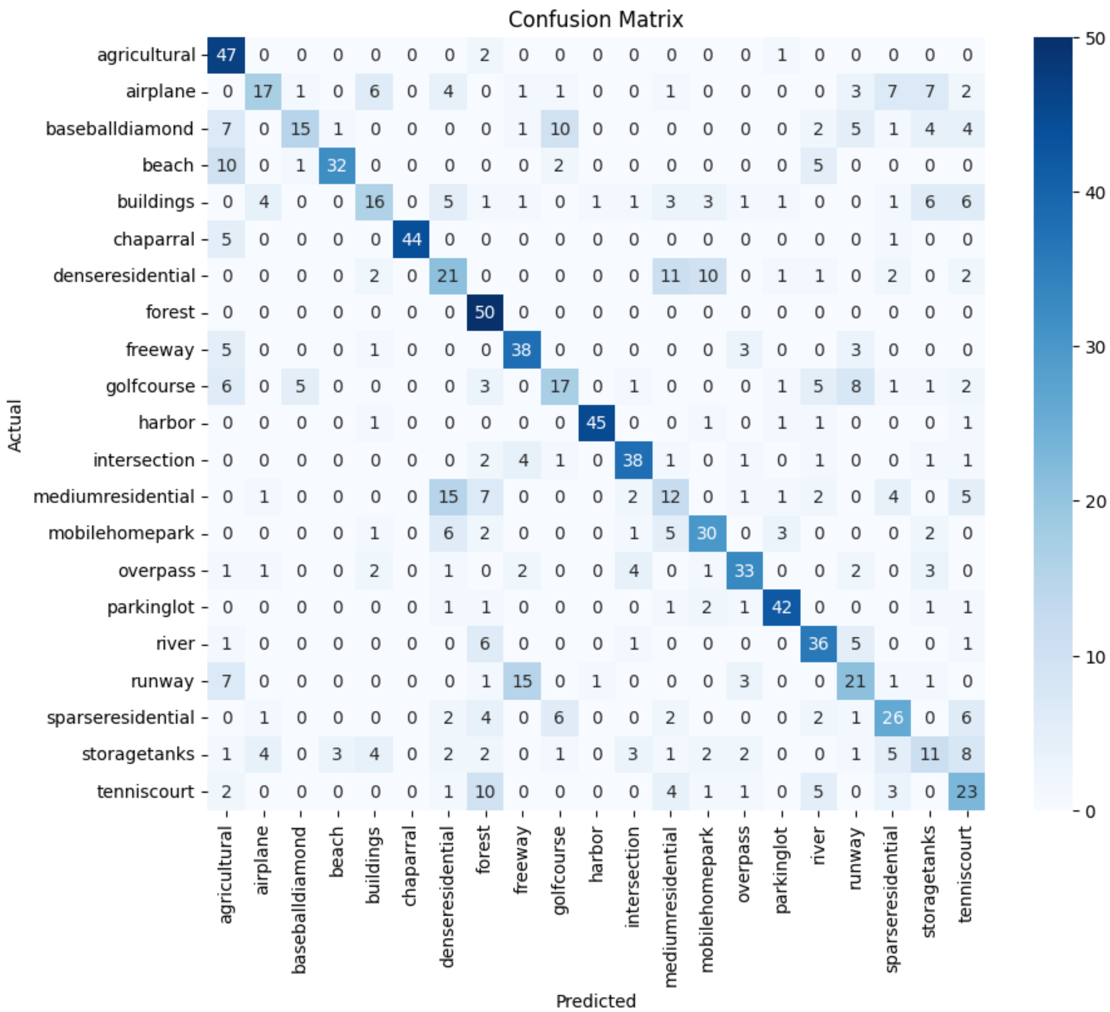
[ ]:
```
def decode_labels(labels):
    return np.argmax(labels, axis = 1)
```

[ ]:
```
true_labels = []
all_pred_classes = []

for images, labels in test_ds:
    predictions = model.predict(images)
    pred_classes = decode_labels(predictions)

    true_labels.extend(decode_labels(labels))
    all_pred_classes.extend(pred_classes)
```

```
1/1 [==============================] - 0s 169ms/step
1/1 [==============================] - 0s 52ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 48ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 45ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 47ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
```

```
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 342ms/step
```

[ ]:
```python
cmt = confusion_matrix(true_labels, all_pred_classes)

plt.figure(figsize = (10, 8))
sns.heatmap(cmt, annot = True, fmt = "d", cmap = "Blues", xticklabels =␣
 ↪class_names, yticklabels = class_names)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



[ ]:
```
report = classification_report(true_labels, all_pred_classes, target_names =
    class_names)
print("Classification Report:")
print(report)
```

```
Classification Report:
                     precision    recall  f1-score   support

      agricultural       0.51      0.94      0.66        50
          airplane       0.61      0.34      0.44        50
   baseballdiamond       0.68      0.30      0.42        50
             beach       0.89      0.64      0.74        50
         buildings       0.48      0.32      0.39        50
         chaparral       1.00      0.88      0.94        50
  denseresidential       0.36      0.42      0.39        50
            forest       0.55      1.00      0.71        50
           freeway       0.61      0.76      0.68        50
        golfcourse       0.45      0.34      0.39        50
            harbor       0.96      0.90      0.93        50
      intersection       0.75      0.76      0.75        50
 mediumresidential       0.29      0.24      0.26        50
    mobilehomepark       0.60      0.60      0.60        50
          overpass       0.72      0.66      0.69        50
        parkinglot       0.82      0.84      0.83        50
             river       0.60      0.72      0.65        50
            runway       0.43      0.42      0.42        50
 sparseresidential       0.50      0.52      0.51        50
      storagetanks       0.30      0.22      0.25        50
       tenniscourt       0.37      0.46      0.41        50

          accuracy                           0.58      1050
         macro avg       0.59      0.58      0.57      1050
      weighted avg       0.59      0.58      0.57      1050
```

[21]:
```
for images, labels in test_ds.take(1):
    predictions = model.predict(images)
    pred_classes = decode_labels(predictions)

    plt.figure(figsize = (10, 10))
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        pred_class = class_names[pred_classes[i]]
        true_class = class_names[np.argmax(labels[i])]
        prob = np.max(predictions[i])
        plt.title(f"Predict: {pred_class} ({prob:.2f})\n Actual: {true_class}")
```

```
        plt.axis("off")
    plt.show()
```

```
1/1 [==============================] - 0s 45ms/step
```