Name: Soumya Ranjan Nayak

PRN: 23070243063

Assignment: Convolution and Pooling Operations on Input Image

In [ ]:
```
import kagglehub
 puneet6060_intel_image_classification_path = kagglehub.dataset_download('puneet6060/intel-image-classification')
 print('Data source import complete.')
```

In [ ]:
```
# import library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import os
import glob as gb
import cv2
import keras
from tensorflow.keras.models import Sequential, Model
```

In [ ]:
```
trainpath = '/kaggle/input/intel-image-classification/seg_train/seg_train'
testpath = '/kaggle/input/intel-image-classification/seg_test/seg_test'
predpath = '/kaggle/input/intel-image-classification/seg_pred/seg_pred'
```

In [ ]:
```
IMAGE_SIZE = (228, 228)

BATCH_SIZE = 32
```

In [ ]:
```
train_ds = tf.keras.utils.image_dataset_from_directory(
  trainpath,
  seed=123,
  image_size=IMAGE_SIZE,
  batch_size=BATCH_SIZE)
```

```
Found 14034 files belonging to 6 classes.
```

In [ ]:
```
test_ds = tf.keras.utils.image_dataset_from_directory(
  testpath,
  seed=123,
  image_size=IMAGE_SIZE,
  batch_size=BATCH_SIZE)
```

```
Found 3000 files belonging to 6 classes.
```

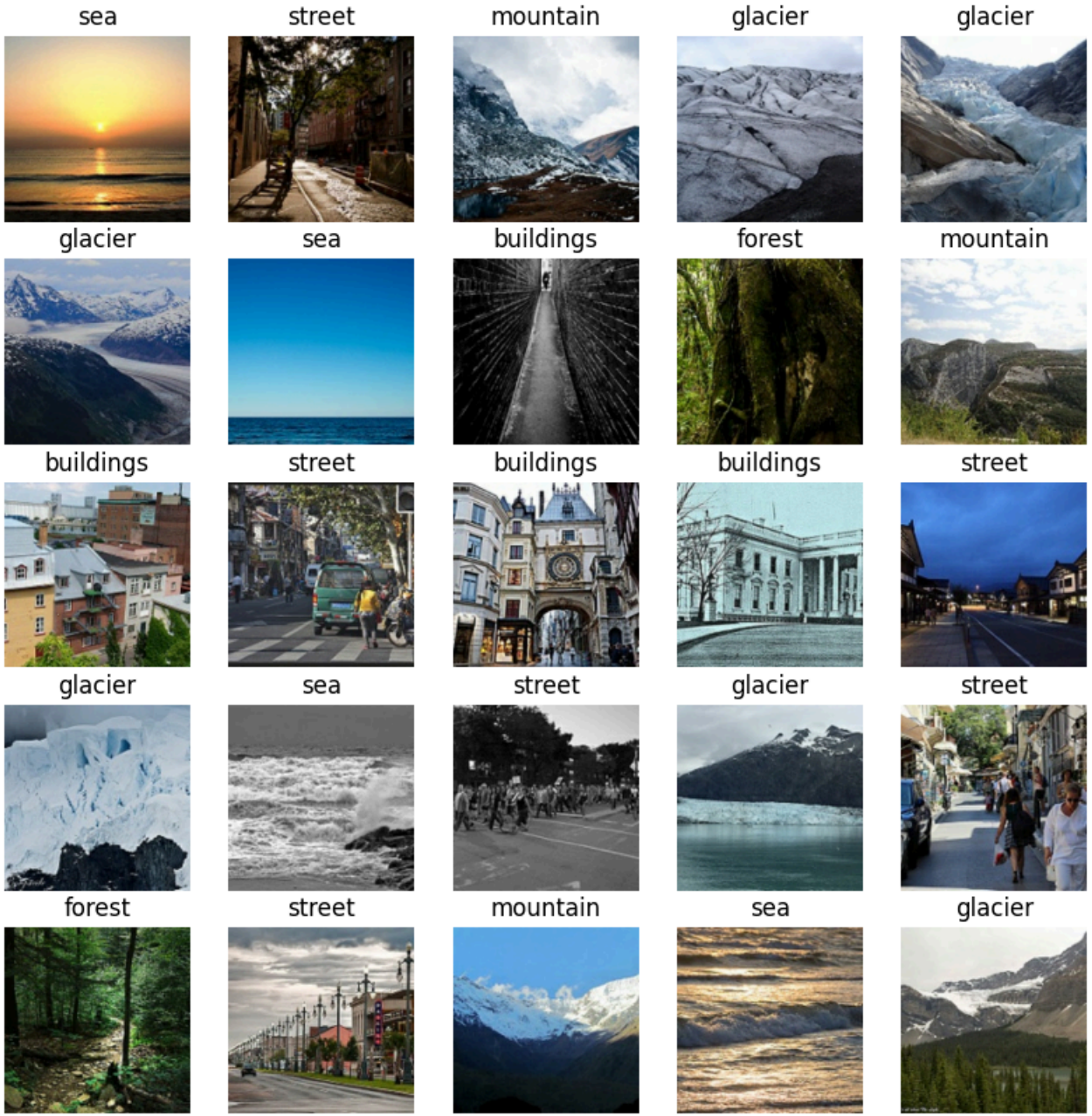
In [ ]:
```
class_names = train_ds.class_names
print(class_names)
```

```
['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']
```

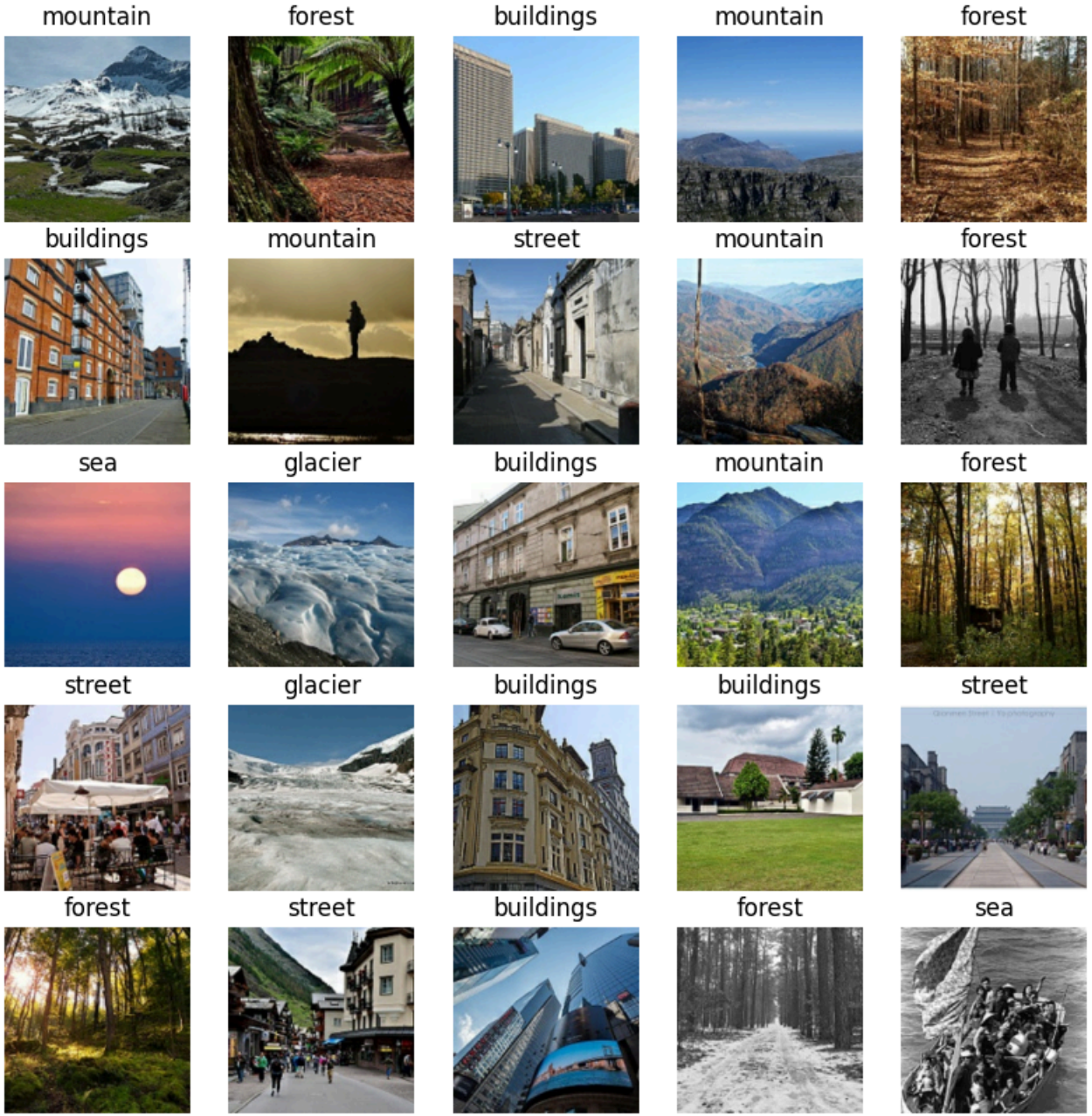
In [ ]:
```
def getImagePaths(path):
    image_names = []
    for dirname, _, filenames in os.walk(path):
        for filename in filenames:
            fullpath = os.path.join(dirname, filename)
            image_names.append(fullpath)
    return image_names
images_paths = getImagePaths(predpath)
len(images_paths)
```

Out[13]:
```
7301
```

In [ ]:
```
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
    for i in range(25):
        ax = plt.subplot(5, 5, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```


sea
street
mountain
glacier
glacier
glacier
sea
buildings
forest
mountain
buildings
street
buildings
buildings
street
glacier
sea
street
glacier
street
forest
street
mountain
sea
glacier

In [ ]:
```
plt.figure(figsize=(10, 10))
for images, labels in test_ds.take(1):
    for i in range(25):
        ax = plt.subplot(5, 5, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```


mountain
forest
buildings
mountain
forest
buildings
mountain
street
mountain
forest
sea
glacier
buildings
mountain
forest
street
glacier
buildings
buildings
street
forest
street
buildings
forest
sea

In [ ]:
```
import tensorflow.keras.models as Models
```

In [ ]:
```
model = Models.Sequential()
model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(228,228,3)))
model.add(tf.keras.layers.MaxPooling2D(2,2))
model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(2,2))
model.add(tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(2,2))
model.add(tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(2,2))
model.add(tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(2,2))
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(1024, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dropout(0.2))
model.add(tf.keras.layers.Dense(len(class_names), activation='softmax'))
```

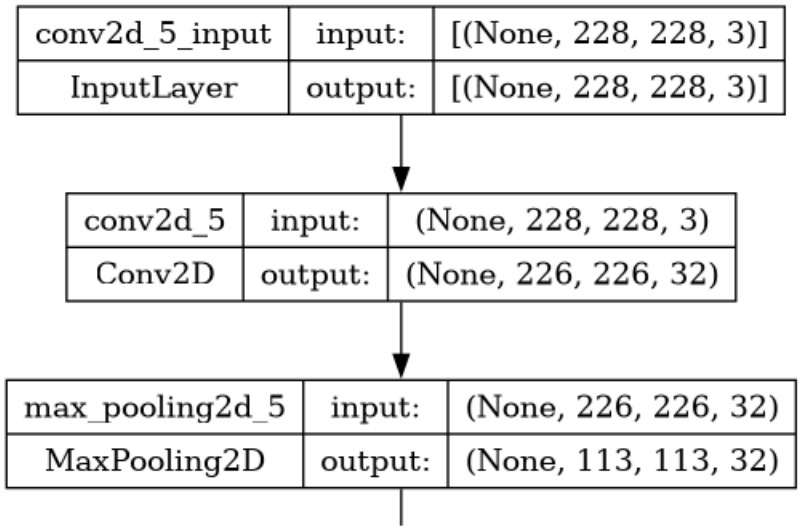
In [ ]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_5 (Conv2D)           (None, 226, 226, 32)      896

 max_pooling2d_5 (MaxPoolin  (None, 113, 113, 32)      0
 g2D)

 conv2d_6 (Conv2D)           (None, 111, 111, 32)      9248

 max_pooling2d_6 (MaxPoolin  (None, 55, 55, 32)        0
 g2D)

 conv2d_7 (Conv2D)           (None, 53, 53, 64)        18496

 max_pooling2d_7 (MaxPoolin  (None, 26, 26, 64)        0
 g2D)

 conv2d_8 (Conv2D)           (None, 24, 24, 64)        36928

 max_pooling2d_8 (MaxPoolin  (None, 12, 12, 64)        0
 g2D)

 conv2d_9 (Conv2D)           (None, 10, 10, 64)        36928

 max_pooling2d_9 (MaxPoolin  (None, 5, 5, 64)          0
 g2D)

 flatten_1 (Flatten)         (None, 1600)              0

 dense_2 (Dense)             (None, 1024)              1639424

 dropout_2 (Dropout)         (None, 1024)              0

 dense_3 (Dense)             (None, 128)               131200

 dropout_3 (Dropout)         (None, 128)               0

 dense_4 (Dense)             (None, 6)                 774

=================================================================
Total params: 1873894 (7.15 MB)
Trainable params: 1873894 (7.15 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Visualize Model

In [ ]:
```
from tensorflow.keras.utils import plot_model
import matplotlib.pyplot as plt

# Visualize the ANN Model
plot_model(model, to_file="ann_model.png", show_shapes=True, show_layer_names=True)
```

Out[23]:



In [ ]:
```
from tensorflow.keras.optimizers import Adam
model.compile(
    optimizer = Adam(learning_rate = 0.001),
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    #loss = "categorical_crossentropy",
    metrics = ["accuracy"])
```

In [ ]:
```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

earlystopping = EarlyStopping(monitor='val_loss',
                              patience=5,
                              verbose=1,
                              mode='min'
                             )

checkpointer = ModelCheckpoint(filepath='bestvalue.keras', verbose=0, save_best_only=True)
callback_list = [checkpointer, earlystopping]
```

In [ ]:
```
history = model.fit(train_ds,
    validation_data=test_ds,
    epochs=40,
    callbacks=callback_list
)
```

```
Epoch 1/40

/opt/conda/lib/python3.10/site-packages/keras/src/backend.py:5727: UserWarning: "`sparse_categorical_crossentropy` received `from_logits=True`, but th
e `output` argument was produced by a Softmax activation and thus does not represent logits. Was this intended?
  output, from_logits = _get_logits(
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1731841900.673195     101 device_compiler.h:186] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the pro
cess.

439/439 [==============================] - 42s 79ms/step - loss: 1.2959 - accuracy: 0.5594 - val_loss: 0.9318 - val_accuracy: 0.6540
Epoch 2/40
439/439 [==============================] - 19s 42ms/step - loss: 0.7888 - accuracy: 0.7098 - val_loss: 0.7498 - val_accuracy: 0.7017
Epoch 3/40
439/439 [==============================] - 19s 43ms/step - loss: 0.6503 - accuracy: 0.7698 - val_loss: 0.6610 - val_accuracy: 0.7700
Epoch 4/40
439/439 [==============================] - 19s 42ms/step - loss: 0.5669 - accuracy: 0.7990 - val_loss: 0.5911 - val_accuracy: 0.8030
Epoch 5/40
439/439 [==============================] - 19s 42ms/step - loss: 0.4990 - accuracy: 0.8244 - val_loss: 0.5630 - val_accuracy: 0.8070
Epoch 6/40
439/439 [==============================] - 18s 41ms/step - loss: 0.4485 - accuracy: 0.8364 - val_loss: 0.8302 - val_accuracy: 0.7403
Epoch 7/40
439/439 [==============================] - 18s 41ms/step - loss: 0.4163 - accuracy: 0.8495 - val_loss: 0.6385 - val_accuracy: 0.7960
Epoch 8/40
439/439 [==============================] - 18s 41ms/step - loss: 0.3804 - accuracy: 0.8617 - val_loss: 0.6208 - val_accuracy: 0.8087
Epoch 9/40
439/439 [==============================] - 18s 41ms/step - loss: 0.3420 - accuracy: 0.8772 - val_loss: 0.6726 - val_accuracy: 0.8123
Epoch 10/40
439/439 [==============================] - 18s 41ms/step - loss: 0.3042 - accuracy: 0.8928 - val_loss: 0.6881 - val_accuracy: 0.7947
Epoch 10: early stopping
```

Accuracy

In [ ]:
```
# Evaluate the model on the test set
loss1, accuracy1 = model.evaluate(test_ds, verbose=0)
print(f"Test Accuracy: {accuracy1:.2f}")
```

```
Test Accuracy: 0.79
```

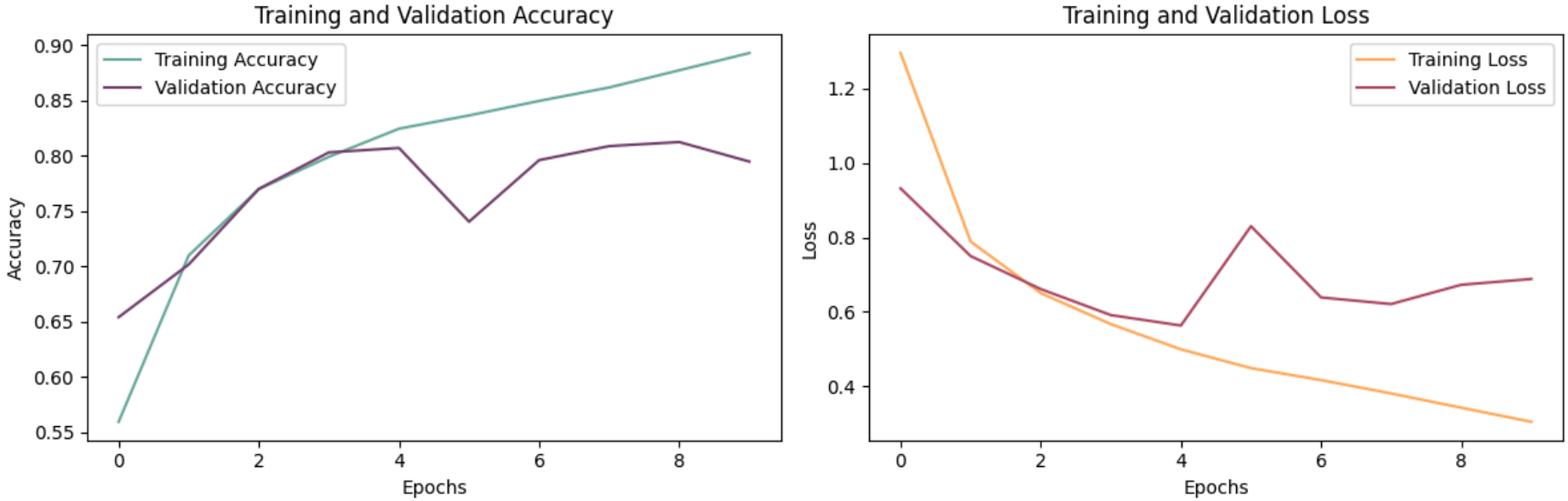
Save Model

In [ ]:
```
# Save the model to a file
model.save("model.h5")
print("Model saved as 'model.h5'")
```

```
Model saved as 'model.h5'

/opt/conda/lib/python3.10/site-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`.
This file format is considered legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')`.
  saving_api.save_model(
```

Plotting training and validation loss over epochs

In [ ]:
```
# Assuming `history` contains the training history
history_df1 = pd.DataFrame(history.history)

# Create a figure with 1 row and 2 columns for side-by-side plots
fig, axes = plt.subplots(1, 2, figsize=(12, 4), sharex=True)

# Plot accuracy on the first subplot
axes[0].plot(history_df1['accuracy'], "#6daa9f", label='Training Accuracy')
axes[0].plot(history_df1['val_accuracy'], "#774571", label='Validation Accuracy')
axes[0].set_title('Training and Validation Accuracy')
axes[0].set_xlabel('Epochs')
axes[0].set_ylabel('Accuracy')
axes[0].legend(loc='best')

# Plot loss on the second subplot
axes[1].plot(history_df1['loss'], "#ffad5a", label='Training Loss')
axes[1].plot(history_df1['val_loss'], "#af4b64", label='Validation Loss')
axes[1].set_title('Training and Validation Loss')
axes[1].set_xlabel('Epochs')
axes[1].set_ylabel('Loss')
axes[1].legend(loc='best')

# Adjust layout for better spacing
plt.tight_layout()
plt.show()
```



User Input

In [ ]:
```
def predict_image(filename, model):
    img_ = image.load_img(filename, target_size=(228, 228))
    img_array = image.img_to_array(img_)
    img_processed = np.expand_dims(img_array, axis=0)
    img_processed /= 255.

    prediction = model.predict(img_processed)

    index = np.argmax(prediction)

    plt.title("Prediction - {}".format(str(classes[index]).title()), size=18)
    plt.imshow(img_array)
```

In [ ]:
```
from tensorflow.keras.preprocessing import image
predict_image('/kaggle/input/intel-image-classification/seg_pred/seg_pred/1003.jpg', model)
```

```
Prediction - Sea
```

In [ ]:
```

```