Report on Satellite Image Classification

Introduction

This report outlines the methodology, results, and conclusions of a project aimed at classifying satellite images using deep learning techniques. The dataset utilized consists of 4 distinct classes of images obtained from sensors and Google Maps snapshots. The primary goal is to develop an automated system capable of interpreting remote sensing images efficiently.

Methodology

Data Preparation

1. **Data Extraction**:
   The dataset was initially stored in a zip file which was extracted to facilitate access to the images.

   ```
   !unzip /content/archive_3.zip -d /content/extracted_files
   ```

2. **Path Definition**:
   A function was implemented to define the file paths of images and their corresponding labels based on the directory structure.

   ```
   def define_paths(dir):
       ...
   ```

3. **DataFrame Creation**:
   A DataFrame was created to organize the file paths and labels for easier manipulation and access during training and testing.

```
def create_df(dir):
    ...
```

4. **Data Splitting**:
   The dataset was divided into training, validation, and testing subsets using an 80-20 split for training versus validation/testing, followed by a 50-50 split of the remaining data.

```
train_df, test_valid_df = train_test_split(df, test_size=0.2, random_state=42)
test_df, valid_df = train_test_split(test_valid_df, test_size=0.5, random_state=42)
```

5. **Data Generators**:
   The Keras `ImageDataGenerator` was employed to augment the training data, enabling better generalization of the model.

```
train_gen = ImageDataGenerator(...).flow_from_dataframe(...)
valid_gen = ImageDataGenerator(...).flow_from_dataframe(...)
test_gen = ImageDataGenerator(...).flow_from_dataframe(...)
```

Model Architecture

A Convolutional Neural Network (CNN) was constructed with the following architecture:

- **Convolutional Layers**: Four convolutional layers with ReLU activation functions were utilized, each followed by max pooling to reduce dimensionality.

- **Dropout Layers**: Added to mitigate overfitting, dropout layers were strategically placed after certain convolutional layers.
- **Flatten and Dense Layers**: The model concludes with a flattening layer followed by a fully connected layer leading to the output layer with softmax activation to handle multi-class classification.

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=input_shape))
...
model.add(layers.Dense(num_classes, activation='softmax'))
```

Model Compilation and Training

The model was compiled using the Adam optimizer and categorical crossentropy as the loss function.

```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

A custom callback function was implemented to stop training when the accuracy reached 99%. The model was trained for 10 epochs with validation data.

```
history = model.fit(train_gen, epochs=10, validation_data=valid_gen, callbacks=[callbacks], verbose=2)
```

Performance Evaluation

After training, the model's performance was evaluated on the test dataset using accuracy and loss metrics. Training and validation accuracies and losses were plotted to visualize the model's learning process.

Results

1. **Training and Validation Accuracy**:
    - The model achieved high accuracy, with indications that it was effectively learning the patterns in the data.
    - Plots displayed the training and validation accuracy, showing a consistent improvement over the epochs.
2. **Loss Evaluation**:
    - Loss metrics were plotted to assess convergence and stability during training, revealing a downward trend indicative of learning.
3. **Confusion Matrix and Classification Report**:
    - A confusion matrix and classification report can be generated to provide insight into the model's performance on each class.
    - accuracy: 0.8731 - loss: 0.7052

```
model.evaluate(test_gen)
predictions = model.predict(test_gen)
```

1. **Final Model**:
    - The trained model was saved for future use.

```
model.save('my_model.h5')
```

Conclusions

The project successfully demonstrated the classification of satellite images using a deep learning approach. Key findings include:

- The CNN architecture effectively learned to distinguish between the different classes of satellite images.

- Data augmentation strategies enhanced the model's generalization capabilities.
- The achieved accuracy indicated the potential for automated interpretation of remote sensing images, contributing to ongoing research and applications in this field.