Approach to Building Count Task

Objective:

The goal is to develop a pipeline to accurately count the number of buildings in a drone image and provide an annotated image showing the detected buildings.

Approach and Techniques Used:

1. **Model Selection**:
    - **Object Detection Model**: I used multiple models for detecting and counting buildings to make the counting more accurate and see what is the best-trained model, modern object detection models such as YOLO (You Only Look Once) are used. YOLO is chosen because of its balance between speed and accuracy, making it suitable for real-time object detection tasks. and to make a more accurate model we need to train a specific model on a specific dataset that looks like this image, we could use a spacenet dataset and to train this dataset we need a lot of time and resources to do it.
2. **Image Processing Pipeline**:
    - **Loading the Image**: The image is loaded and converted to RGB format for compatibility with the detection models.
    - **Model Loading and Prediction**: Multiple YOLO models, each configured differently, are loaded. Predictions are made on the image using these models. The image is sliced and processed in chunks to handle large images effectively.
    - **Post-processing**: Predictions are filtered based on confidence scores, and bounding boxes or masks are drawn on the image. The count of buildings is calculated and displayed.
    - **Saving and Displaying Results**: The annotated image is saved to the output directory, and a pop-up window displays the image for visual inspection.
3. **Models Used**:
    - **YOLOv8 Variants**: Several YOLO models are utilized, including versions with different scales (small, medium, large) and configurations, each pre-trained on specific datasets.
4. **Techniques and Tools**:
    - **Object Detection Models**: YOLOv8 models for building detection.
    - **Image Slicing and Overlapping**: To handle large images and ensure complete coverage, the image is sliced, and overlapping areas are processed.
    - **Post-processing**: Bounding boxes and masks are drawn to visualize detected buildings.
5. **Challenges Faced**:
    - **Model Performance**: Different YOLO models may have varying performance and accuracy. The choice of models and their configurations can impact detection accuracy.
    - **Large Image Handling**: Processing large drone images requires efficient slicing and overlapping to avoid missing objects and reduce memory consumption.

- **Threshold Tuning**: Determining the appropriate confidence score threshold for building detection can affect the number of false positives/negatives.
- **Integration and Optimization**: Ensuring smooth integration of model loading, image processing, and result saving can be complex, especially with multiple models and large images.

Detailed Breakdown

1. `inference.py`:
   - **Functionality**: The script orchestrates the entire pipeline. It loads the image, processes it using various models, draws predictions, and saves the annotated image.
   - **Key Functions**:
     - `process_image()`: Handles loading, prediction, drawing, and saving of results.
     - `main()`: Loads configuration and initiates processing.
   - **Challenges**:
     - Managing multiple models and ensuring that each performs well.
     - Handling GUI operations and saving results efficiently.
2. `download_models.sh`:
   - **Functionality**: Downloads the pre-trained YOLO models from remote sources and saves them locally.
   - **Challenges**: Ensuring that the correct models are downloaded and managing file paths.
3. `config.yaml`:
   - **Functionality**: Defines the configuration for models, including paths, thresholds, and slicing parameters.
   - **Challenges**: Correctly setting paths and thresholds to match the model requirements and image characteristics.
4. `model.py`:
   - **Functionality**: Contains the logic for loading the detection models using the `sahi` library.
   - **Challenges**: Handling model loading errors and ensuring that models are correctly initialized.
5. `config.py`:
   - **Functionality**: Manages loading configuration from a YAML file.
   - **Challenges**: Ensuring that the configuration file is correctly formatted and loaded.
6. `image.py`:
   - **Functionality**: Provides utilities for image loading, prediction counting, drawing annotations, and saving images.
   - **Challenges**: Handling image processing, including drawing and mask application, efficiently.

Summary

Your approach involves using advanced YOLO object detection models to count buildings in drone images. The techniques employed include slicing images for processing, applying various YOLO models, and visualizing results with annotations. Challenges include managing model performance, handling large images, tuning detection thresholds, and integrating various components of the pipeline effectively.

Submission

1. **Source Code**: Included Python code implementing the above approach.
2. **Annotated Image**: Final annotated image showing detected buildings with bounding boxes.
3. **Building Count**: Total count of buildings provided in the report or printed in the output.

Note: The models weights can be downloaded from the shell script “download_model.sh”. This approach ensures accurate building detection, provides clear visual annotations, and efficiently handles large images, meeting the task requirements effectively.