Problem Statement: Watermark Detection in Images (7 days max)

You are tasked with creating an end-to-end solution to detect and localize watermarks in a set of images. The goal is to build, train, and deploy a machine learning model that can identify the presence of Rumah123 or 99.co watermarks and highlight their locations in images.

Instructions:

1. **Dataset**:

- A dataset of images, some with visible watermarks and some without, will be provided.
- Each image will be labeled to indicate the presence or absence of a watermark.
- Optionally, bounding box annotations or segmentation masks indicating the location of the watermark will be included for the images with watermarks.
- You are expected to perform all necessary data preprocessing to ensure the model can generalize well to unseen images.

2. **Tasks**:

Step 1: Data Preprocessing & Exploration

- Analyze and clean the dataset. Ensure that all images are appropriately processed (e.g., resizing, normalization) for training.
- Perform data augmentation (e.g., flips, rotations, noise) to increase the robustness of your model.
- Visualize the distribution of images with and without watermarks and analyze any challenges such as class imbalance.

Step 2: Model Selection & Training

- Build a machine learning or deep learning model that can detect the presence of watermarks in images.
- You can use object detection models (e.g., Faster R-CNN, YOLO) if working with bounding box annotations, or segmentation models (e.g., U-Net) for pixel-level detection.
- Train the model and evaluate its performance using appropriate metrics such as precision, recall, F1-score, and Intersection-over-Union (IoU) for location accuracy.
- Document your process for hyperparameter tuning and the rationale behind model architecture selection.

Step 3: Model Deployment

- Deploy the trained model as a REST API using Flask or FastAPI.
- The API should accept an image as input and return whether a watermark is detected and, if so, its location in the form of bounding boxes or a heatmap overlay.
- Containerize the application using Docker.

Step 4: Kubernetes Deployment

- Provide a Dockerfile to containerize the model and API.
- Include Kubernetes YAML manifests (Deployment, Service, etc.) for deploying the container to a Kubernetes cluster.
- While you don't need to deploy the application on a Kubernetes cluster for the take-home test, ensure that your instructions and deployment files are production-ready.

3. Bonus (Optional):

- Automate the model training pipeline using Airflow or a similar orchestration tool.
- Set up a CI/CD pipeline to automatically build and deploy the Docker container when changes are pushed to a version control repository (e.g., GitHub).

4. Deliverables:

- **Code**: Provide a GitHub repository link containing the code for the model, API, and Kubernetes deployment files.
- **Documentation**: Include a README file with detailed instructions on how to:
- Run the code and model locally.
- Deploy the model using Docker and Kubernetes.
- Use the API for watermark detection.
- **Report:** Submit a brief report (2-3 pages) that explains:
- Data preprocessing, model selection, and performance metrics.
- Deployment steps and any challenges faced during the process.

5. Evaluation Criteria:

- **Model performance:** How accurately the model detects and localizes watermarks in images.
- **Deployment readiness:** The ability to package the model using Docker and provide appropriate Kubernetes deployment files.
- **Code quality:** Clean, modular code that follows best practices (version control, testing, etc.).
- **Documentation:** Clear and concise instructions in the README and accompanying report.
- **Bonus features:** Extra implementations such as monitoring, automation, or CI/CD pipelines.

***Notes**: Images Zip file attached on email.