Rapid Prototyping of Containerized Multimodal Deep Learning Solutions for Geospatial AI Applications

---

Overview

Develop a secure, containerized Python application implementing state-of-the-art computer vision (CV) and vision-language models (VLM) to perform:

- Image retrieval
- Segmentation
- AI-assisted labeling
- Object detection
- Visual question answering

Utilize geospatial datasets, including satellite imagery, aerial imagery, full-motion video (FMV), ground photos, and OpenStreetMap data.

Google Colaboratory Link

Project Plan

1. Data Retrieval and Preprocessing
    - **Setup S3 Access with Boto3**
        - Configure AWS credentials securely using IAM roles or environment variables.
        - Implement robust error handling for API calls.
        - Optimize data retrieval with multithreading or multiprocessing.
    - **Data Acquisition**
        - Retrieve satellite imagery, aerial imagery, FMV data, ground photos, OpenStreetMap data.
        - Organize data in a hierarchical file structure.
    - **Image Preprocessing**
        - Implement image chipping to divide large images into smaller tiles.
        - Apply data augmentation: rotation, scaling, flipping, and color adjustments.

        - Normalize pixel values and convert images to required formats (PNG, JPEG).
    - **Annotation Generation**
        - Use AI-assisted labeling tools for initial annotations.
        - Convert annotations to formats compatible with training frameworks (COCO, Pascal VOC).

2. Model Development

- **Transfer Learning**
    - Select pre-trained models: ResNet50, U-Net, DINO, DeiT.
    - Freeze initial layers; retrain top layers with custom data.
    - Use PyTorch or TensorFlow for implementation.
    - Optimize hyperparameters using validation sets.
- **Knowledge Distillation**
    - Implement teacher-student architectures.
    - Use larger models to enhance smaller, deployable models.
- **Object Detection and Segmentation**
    - Utilize Faster R-CNN and Mask R-CNN for object detection.
    - Use U-Net or FCNs for segmentation tasks.
- **Vision Transformers (ViT)**
    - Implement DINO or DeiT models.
    - Compare performance with CNN-based models.
- **Visual Question Answering (VQA)**
    - Use multimodal models combining image and text inputs.
    - Train with adapted VQA v2 datasets for geospatial imagery.

3. GPU Acceleration

- **CUDA Environment Configuration**
    - Install NVIDIA drivers and CUDA toolkit.
    - Verify GPU availability in PyTorch/TensorFlow.
- **Model Optimization**
    - Use mixed-precision training with NVIDIA Apex.
    - Implement data parallelism for multiple GPUs.

4. Containerization and Security

- **Dockerfile Creation**
    - Base image: python:3.8-slim.
    - Install necessary dependencies only.
    - Copy the application code with proper permissions.
- **Container Hardening**
    - Run applications as non-root users.
    - Use multi-stage builds to minimize image size.
    - Scan images with Clair or Anchore.
- **CI/CD Pipeline Automation**

        - Set up GitLab CI/CD with .gitlab-ci.yml.
        - Include stages: build, test, scan, deploy.
        - Use GitLab Runner for pipeline execution.

5. Deployment and Orchestration

- **Kubernetes Manifests**
    - Write YAML files for Deployment, Service, ConfigMap, and Secret resources.
    - Configure CPU and memory resource limits.
- **Helm Deployment**
    - Create Helm charts with templates.
    - Parameterize values.yaml for configuration.
- **OpenShift Deployment**
    - Adapt manifests for OpenShift specifics.
    - Use OpenShift CLI (oc) for deployment.
- **Container Orchestration**
    - Configure horizontal pod autoscaling.
    - Use Operators for application lifecycle management.

6. Explainable AI (XAI) and Validation

- **XAI Implementation**
    - Use Grad-CAM or LIME for model visualization.
    - Integrate XAI outputs into the application interface.
- **Verification and Validation**
    - Create test datasets with known outcomes.
    - Implement unit tests using pytest.
    - Measure accuracy, precision, recall, and F1-score.
- **Documentation of Methodologies**
    - Provide detailed explanations of model architectures.
    - Justify model and parameter selections.

7. Interoperability with ONNX

- **Model Conversion**
    - Use torch.onnx.export() or tf2onnx.
    - Verify models with ONNX Runtime.
- **Model Optimization**
    - Apply quantization and pruning.
    - Ensure efficient inference across platforms.

8. Documentation and Demonstrations

- **Comprehensive Documentation**
    - Setup instructions, user guides, and API documentation.
    - CI/CD and deployment process details.
- **Demonstration Scripts**
    - Jupyter notebooks showcasing capabilities.

        - Examples of retrieval, segmentation, detection, and VQA.
    - **Results Presentation**
        - Slides and reports summarizing performance.
        - Highlight XAI benefits in model interpretation.

Deliverables

- **Source Code Repository**
    - Hosted on GitLab with version control.
    - Includes Python scripts, Dockerfiles, manifests, Helm charts, and CI/CD configurations.
- **Container Images**
    - Secure, scanned Docker images ready for deployment.
    - Tagged and stored in a private registry.
- **Documentation**
    - User manuals, technical documentation, API guides.
- **Demonstration Materials**
    - Jupyter notebooks, sample datasets, XAI visualizations.
- **Validation Reports**
    - Test results with performance metrics.
    - Security audit reports.
- **ONNX Models**
    - Converted optimized models with usage instructions.

Timeline

- **Week 1**: Data retrieval setup (S3 access, data acquisition).
- **Week 2**: Image preprocessing and annotation.
- **Week 3**: Model fine-tuning with transfer learning.
- **Week 4**: Knowledge distillation implementation.
- **Week 5**: Object detection and segmentation models.
- **Week 6**: Vision Transformers integration; CUDA setup.
- **Week 7**: GPU optimization; containerization begins.
- **Week 8**: Container hardening; CI/CD pipeline setup.
- **Week 9**: Deployment manifests; Helm charts creation.
- **Week 10**: OpenShift deployment; orchestration configuration.
- **Week 11**: XAI techniques, verification, and validation.
- **Week 12**: ONNX conversion; documentation drafting.
- **Week 13**: Demonstrations preparation; results presentation.

- **Week 14**: Finalize deliverables, testing, and validation reports.

Technical Requirements

- **Languages and Tools**
  - Python 3.8+, PyTorch 1.8+, TensorFlow 2.x.
  - Boto3, NumPy, OpenCV, CUDA Toolkit 11.x.
- **Containerization**
  - Docker Engine 20.x, Kubernetes 1.20+, OpenShift 4.x.
  - Helm 3.x, GitLab CI/CD.
- **Hardware**
  - NVIDIA GPUs (Tesla V100, RTX 3090).
  - Servers with adequate CPU and RAM.
- **Security Tools**
  - Clair, Anchore, dependency-check tools.

Acceptance Criteria

- **Functionality**
  - Accurate performance of all specified tasks.
  - Models achieve >85% accuracy.
- **Security**
  - Containers pass vulnerability scans with no critical issues.
  - Secure handling of credentials and data.
- **Performance**
  - Efficient GPU utilization.
  - Scalable deployment supporting concurrent users.
- **Documentation**
  - Precise, detailed, and up-to-date.
  - Tested deployment and usage instructions.
- **Explainability**
  - XAI provides clear model decision insights.
  - Integrated visualizations in the interface.

Team Structure

- **Project Manager**
  - Oversees timeline and deliverables.
- **Machine Learning Engineers**
  - Model development and XAI implementation.
- **DevOps Engineers**
  - Containerization, CI/CD, deployment.
- **Data Engineers**
  - Data retrieval, preprocessing, storage.
- **Security Analysts**
  - Security audits, vulnerability assessments.
- **Technical Writers**
  - Documentation and demonstration materials.

Resources and Access

- **Data Sources**
  - S3 buckets with geospatial datasets.
  - Necessary licenses and permissions.
- **Compute Resources**
  - GPU-equipped cloud instances (AWS EC2 P3).
  - Kubernetes/OpenShift cluster access.
- **Software Licenses**
  - Compliance with organizational policies.
- **Security Clearances**
  - Active TS/SCI for all team members.
  - Protocol adherence for sensitive information.

Risk Management

- **Data Sensitivity**
  - Encryption for data at rest and in transit.
  - Access controls and audit logging.
- **Model Performance**
  - Mitigate overfitting with cross-validation.
  - Update models with new data regularly.
- **Deployment**

    - Test in staging environments.
    - Implement rollback strategies.
- **Compliance**
    - Stay updated with regulations.
    - Document compliance measures.

Conclusion

This detailed project plan outlines the development of a secure, efficient, and explainable geospatial AI application. It integrates advanced machine learning techniques, robust DevOps practices, and stringent security measures to address key intelligence questions and enhance analytic workflows, fulfilling all job requirements.