CropSegmentation: M2

1. M1 corrections

After the feedback and instructions received during the presentation of the first milestone, the following changes were made:

1. metadata was added to the model card
2. a link from the model card to the dataset card was added
3. a link from the `README` of the repo to the model card was added
4. the model card was moved into the `model` folder and renamed `README`
5. the dataset card was moved into the `data` folder and renamed `README`
   a. `.gitignore` was modified accordingly

2. Data versioning with DVC

1. **Initialize DVC:** DVC was initialized for the project creating necessary configurations to keep track of data versions.
2. **Configure Remote Storage:** Configured a remote storage on Google Drive to store DVC-tracked files, facilitating shared access and backup of data.
3. **Add Raw Data:** The raw data was added to DVC, initially excluding it from source control to prevent accidental upload to GitHub. Made adjustments to the `.gitignore` file to properly handle data directory and added data to DVC for versioning.
4. **Update Raw Data:** The raw data was updated resulting in a change in the hash value in DVC files indicating a new version of the data.
5. **Create Global Config File:** Created a global configuration file to centralize and manage configuration settings for data paths, model training parameters, and other project-wide settings.
6. **Update README.md:** Updated the project's `README` file to reflect the current project structure.
7. **Update requirements.txt**: Updated the project's dependencies to ensure compatibility and reproducibility.
8. **Upload extract_data.py:** Uploaded a script to handle data extraction from a .zip file, facilitating the setup of the data processing pipeline.
9. **Upload utils.py:** Uploaded utility functions necessary for the project.

10. **Update make_dataset.py:** Updated the script that prepares the dataset for training and testing.
11. **Upload model.py:** Uploaded the script defining the machine learning model.
12. **Update train_model.py, predict_model.py, make_dataset.py:** Fixed issues.
13. **Add Processed Data:** Added processed data to DVC for versioning.
14. **Add Trained Model Dump to DVC:** Versioned the trained model using DVC.
15. **Add Metrics Reporting:** Added a mechanism to report training and validation metrics.
16. **Create DVC Pipeline:** Created a DVC pipeline to automate and version the data preparation, training, and evaluation process.
17. **Modify DVC Configuration to Optimize Space and Time**: Due to the blocking of the Google Drive account owing to excessive traffic, modifications were made to the `dvc.yaml` file to reduce the redundancy of files and to only version essential files, minimizing the occupied space. A notable change included using `persist: true` in the DVC configuration for the trained model file. This change allows the execution of the pipeline without the need to retrain the model, significantly reducing the time required to run the pipeline.

3. MLflow Integration

1. **Integrate MLflow in predict_model.py:** Integrated MLflow into the evaluation script to log metrics, parameters, and artifacts facilitating experiment tracking.
2. **Update requirements.txt for MLflow:** Added MLflow to the project's dependencies to ensure reproducibility and ease of setup for experiment tracking.
3. **Perform Hyperparameter Tuning Experiments:** In order to perform experiment tracking with MLflow, four different experiments were conducted by varying the model's hyperparameters. MLflow was used to ensure systematic logging and tracking of different configurations and their corresponding outcomes, with a structured approach to experiment management. The experiments were conducted with the following configurations:
    - Experiment 1: learning rate 0.001, batch size 50
    - Experiment 2: learning rate 0.1, batch size 25
    - Experiment 3: learning rate 0.001, batch size 25
    - Experiment 4: learning rate 0.1, batch size 50
4. **Update DVC Files for Experiments:** Updated `dvc.lock` and `dvc.yaml` files to reflect the different experiments conducted. `The dvc.lock` file was updated with the outputs of different experiments, including model files and metrics. In `dvc.yaml`, the output files were updated to include the `persist: true` flag, ensuring the

preservation of model files across different pipeline runs. These updates ensure a smooth interaction between DVC and MLflow, where DVC manages the data and model versions while MLflow handles experiment tracking.

5. **Modify Configuration and Training Script for Hyperparameter Tuning:** Updated `src/config.py` to include a dictionary of hyperparameters for grid search, and modified `src/models/train_model.py` to implement a loop over the hyperparameter grid. This setup facilitated the systematic exploration of hyperparameter space and the logging of different experiment configurations and outcomes with MLflow.

6. **Update DVC Configuration for Metrics:** Updated `dvc.yaml` to include the metrics files generated during the model evaluation phase. This addition allows DVC to track and version the metrics files, allowing a comprehensive examination of model performance across different hyperparameter configurations.

7. **Enhance Metrics Logging in predict_model.py:** Modified `src/models/predict_model.py` to update the metrics logging mechanism to ensure structured logging of evaluation metrics, which is essential for assessing and comparing model performance.
8. **Update project folder structure:** The `README` file was updated to reflect the structure of the project. In particular, an `mlruns` directory was added for MLflow experiment tracking data, and, within the `src` directory, the model architecture was separated from the training and prediction script, improving modularity and clarity.
9. **Execute MLflow experiments:** The pipeline was executed and the generated `mlruns` folder, containing the logs and artifacts from the experiments, was uploaded to the repository.

4. DagsHub

1. **Create repository:** A repository was created within the se4ai2324-uniba organization on DagsHub and connected to the GitHub repository of the project.
2. **Experiment Tracking Integration:** The tracking of experiments conducted in the project and logged with MLFlow was enabled for DagsHub.