Machine Learning sensors fusion of LiDAR and HS for classification map

Francesca Stefano and Giovanni Schianchi

November 26, 2023

1 Description

Pytorch implementation of the paper for ML sensor fusion of different sensor for classication map. Multi-stream CNNs are commonly used in multi-source remote sensing data fusion. In this project we propose a strategy that enables single-stream CNNs to approximate multi-stream models using group convolution. The proposed method is applied to ResNetv18, ResNetv50 and tb_CNN, and evaluated on MUUFL data sets, obtaining good results. On a practical level, data from LiDAR and HS were merged to obtain classification maps based on the different predefined categories of the dataset.

2 Dataset

The MUUFL Gulfport dataset is an HS-LiDAR dataset. It was collected over The University of Southern Mississippi, Gulf Park Campus, Long Beach, MS, USA. The dataset contains co-registered HS and MS-LiDAR data, with 64 and 2 bands, respectively. The wavelength of HS data spectral bands ranges from 375 to 1050 nm. The dataset contains 325×220 pixels, with a spatial resolution of 0.54 m across track and 1.0 m along track. In the ground truth labels, there are 11 classes.

#	Class	# Training samples	# Test samples
1	Trees	100	23246
2	Mostly Grass	100	4270
3	Mixed Ground Surface	100	6882
4	Dirt and Sand	100	1826
5	Road	100	6687
6	Water	100	466
7	Building Shadow	100	2233
8	Building	100	6240
9	Sidewalk	100	1385
10	Yellow Curb	100	183
11	Cloth Panels	100	269
	Total	1100	53687

Figure 1: CLASSES IN MUUFL DATASET

3 Implementation

For ResNet18, we use image patch of 11×11 as training and test samples, while for ResNet50, we use an image patch of size 17×17. We make random train-test split in each replica under different random seeds and also we use Two-branch CNN that is a multi-stream model. All the models are trained on a single GPU.
For ResNet18 we used as hyperparameters : epochs : 22, learning rate : 0.02, optimizer : sgd, batch size : 48, sample radius : 5 and learning schedule : [200, 400].
For ResNet50 we used as hyperparameters : epochs : 22, learning rate : 0.01, optimizer : adam, batch size : 64, sample radius : 8 and learning schedule : [300, 350].
For TB we used as hyperparameters : epochs : 22, learning rate : 0.001, optimizer : adam, batch size : 48, sample radius : 5 and learning schedule : None.

3.1 Code

3.1.1 Main.py

- **Data Preparation:** It calls the _get_dataset function to load the dataset. The function returns the training and testing data, which are then assigned to the variables X, y, X_test, and y_test respectively. It also initializes arrays conf_mats, p_scores, r_scores, f1_scores, k_scores, and oa_arr with zeros.
- **Model Preparation:** It prepares the model and optimizer. It calculates the class weights based on the distribution of classes in the target variable y. It also sets the loss function (criterion) based on whether the program is configured to mask undefined classes in the target variable.
- **Model Training:** It trains the model by calling the train function and assigns its return values to the variables model and losses.
- **Model Testing and Evaluation:** It tests the trained model and evaluates its performance. It calculates various evaluation metrics such as confusion matrix, precision, recall, F1 score, Cohen's kappa score, and overall accuracy.

3.1.2 Train.py

The train function is the main function that trains the model. It takes several parameters such as the including the model, a data loader for the training data, an optimizer, a loss function (criterion), and several optional parameters .It creates a MultiStepLR scheduler that adjusts the learning rate at the specified milestones. It then enters a loop that runs for the specified number of epochs. In each epoch, it iterates over the training data, performs forward and backward propagation, and updates the model parameters. After each epoch, it also updates the learning rate if a schedule is provided. The function returns the trained model and an array of losses for each epoch.

3.1.3 Test.py

This script has test_clf. The test_clf function is used to generate predictions for a given test dataset, with an optional parameter to provide the true labels (y_test). It takes as input a trained model, a test dataset (X_test), and optional parameters for the radius of the samples (sample_radius) and the batch size for testing (batch_test). The function first initializes an empty list to store the predictions and a list to store the test samples. It then loops over the test dataset, taking samples of the specified radius, and feeds these samples to the model in batches to generate predictions. The predictions are then added to the list of predictions. If the y_test parameter is provided, the function only generates predictions for the samples where the true label is greater than 0. The function returns the predictions as a 1D array if y_test is provided, otherwise it reshapes the predictions to match the shape of the test dataset.

3.2 Results

	ResNet18	ResNet50	TB_CNN
Accuracy	90.545	79.455	97.818
Losses	0.253	0.860	0.073
k_scores	0.726	0.683	0.823
oa_arr	0.786	0.75	0.863

Table 1: Results MUUFL

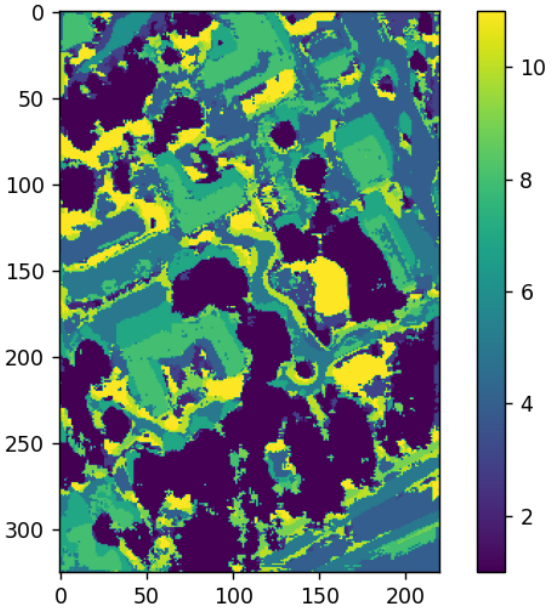

Figure 2: Result Map of ResNet18

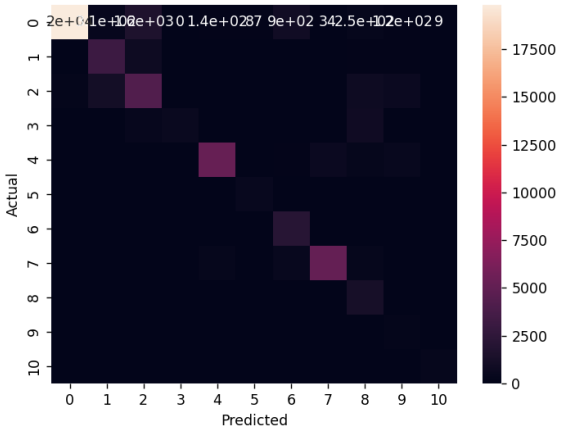

Figure 3: Confusion Matrix of ResNet18

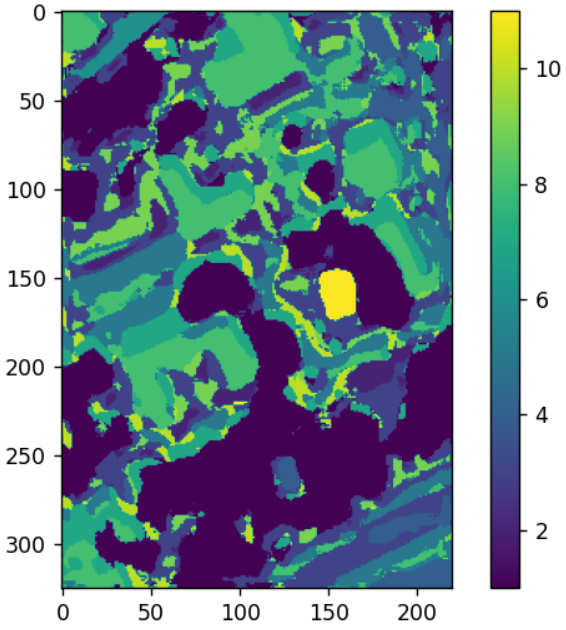

Figure 4: Results Map of ResNet50

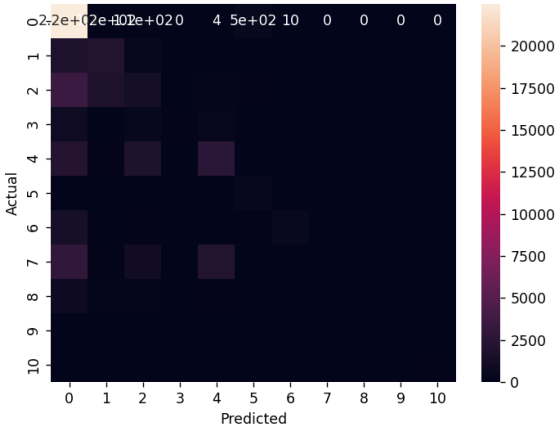

Figure 5: Confusion matrix of ResNet50

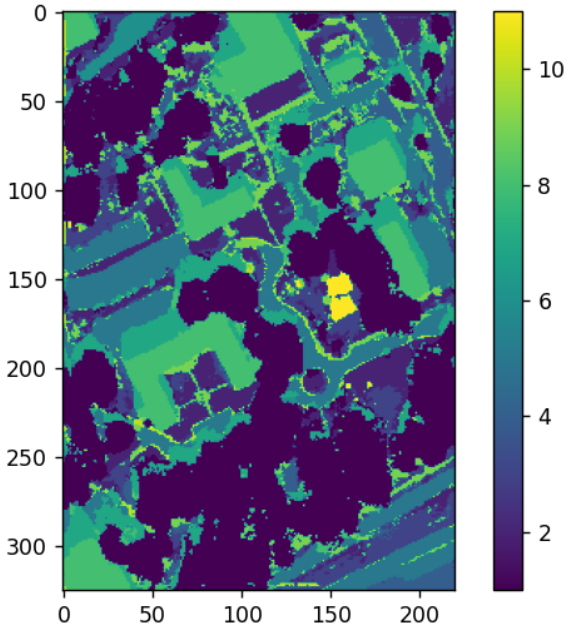

Figure 6: Results Map of TB CNN

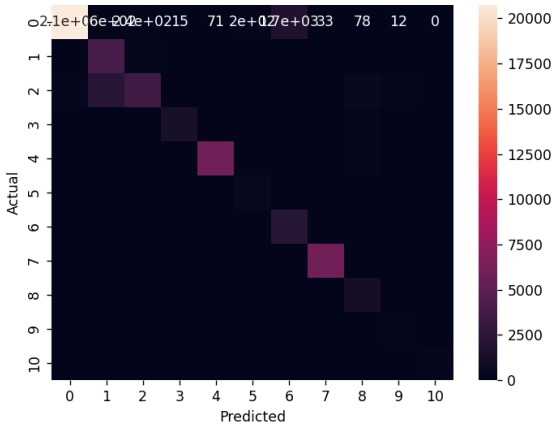

Figure 7: Confusion matrix of TB CNN

	trees	grass	surface	sand	road	water	building shadow	buildings	sidewalk	curb	panels
p_scores	0.983	0.698	0.600	0.991	0.889	0.799	0.594	0.881	0.318	0.104	0.749
r_scores	0.852	0.740	0.608	0.261	0.784	0.982	0.933	0.823	0.859	0.945	0.988
f1_score	0.913	0.718	0.604	0.414	0.833	0.881	0.726	0.851	0.464	0.188	0.852

Table 2: Results ResNet18 for each category

	trees	grass	surface	sand	road	water	building shadow	buildings	sidewalk	curb	panels
p_scores	0.983	0.700	0.249	0.639	0.367	0.377	0.617	0.892	0.245	0.029	0.403
r_scores	0.967	0.488	0.177	0.076	0.372	0.933	0.199	0.003	0.004	0.016	0.568
f1_score	0.760	0.470	0.197	0.142	0.415	0.475	0.316	0.660	0.008	0.024	0.689

Table 3: Results ResNet50 for each category

	trees	grass	surface	sand	road	water	building shadow	buildings	sidewalk	curb	panels
p_scores	0.975	0.612	0.854	0.910	0.937	0.679	0.513	0.949	0.508	0.296	0.843
r_scores	0.892	0.895	0.500	0.718	0.890	0.991	0.957	0.953	0.855	0.983	0.998
f1_score	0.932	0.727	0.631	0.803	0.913	0.806	0.668	0.951	0.638	0.455	0.914

Table 4: Results TB CNN for each category