Multi label encoding for layout classification

This approach uses a multi-label classification scheme to represent each of the four classes as a vector of length five, where each dimension could be common among the four classes (See the table). This approach may be advantageous as we have some overlapping attributes between our classes. The encoding table can be refined by the humanities scholars.

Multi label encoding table

Class	Page width text line	Half page width text line	Page height vertical separator	Half page height vertical separator	Multi fonts
0	0	0	0	0	0
1	1	0	0	0	0
2	0	1	1	0	0
3	1	1	0	1	1

Training

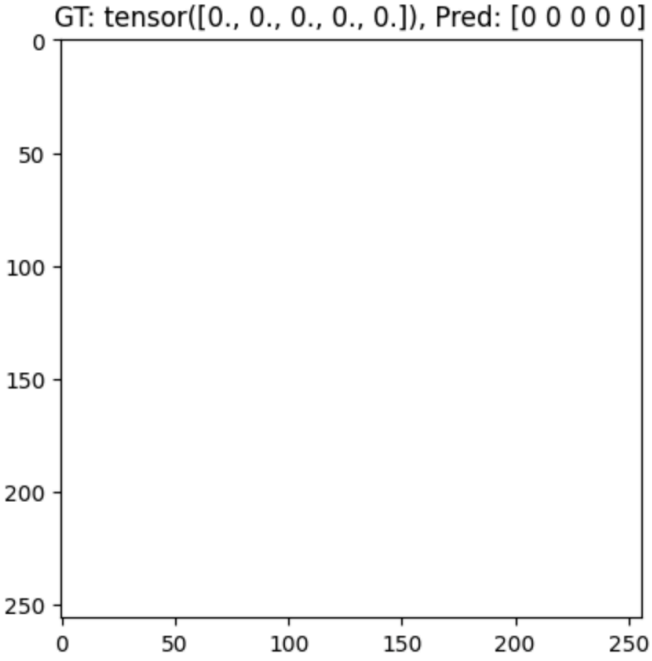
The model was trained on a dataset of 1352 images, which was divided into three sets: a training set (80%), a validation set (10%), and a test set (10%).

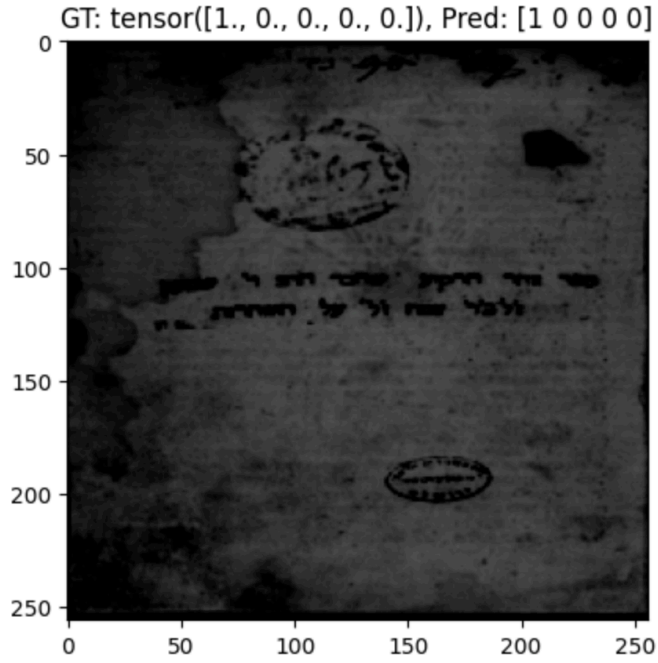
Evaluation

The evaluation loops through the predicted labels to assign them to a predicted class. If a predicted label doesn't match any of the defined classes, it finds the nearest class based on Euclidean distance. Then it calculates the true positives, true negatives, false positives, and false negatives by comparing the true and predicted class lists.
The confusion matrix is:
[[29 1 0 0]
[0 41 0 0]
[0 0 25 0]
[0 0 2 38]]

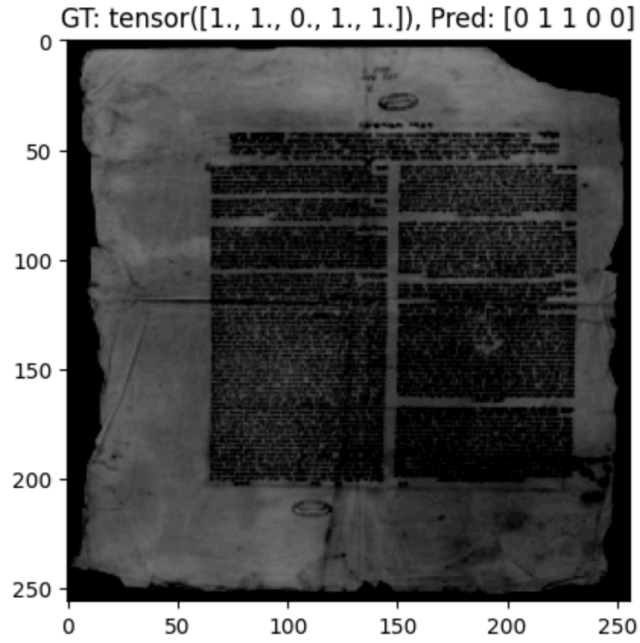
The accuracy is: 0.978

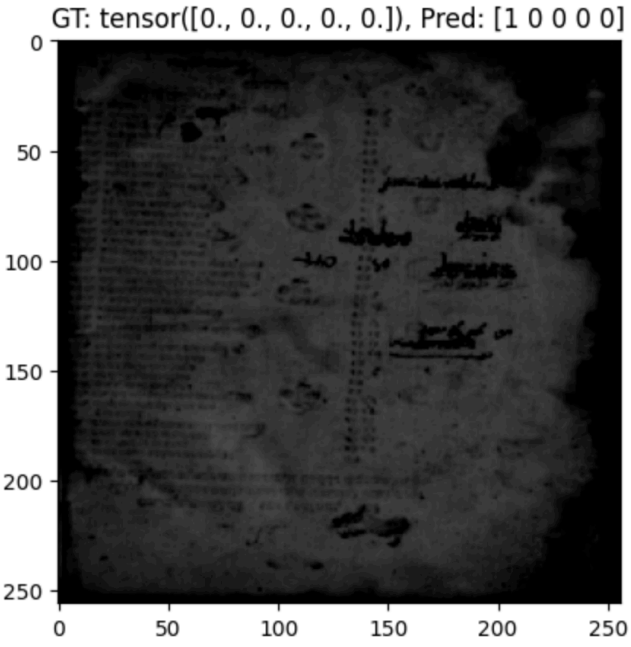
Visual results

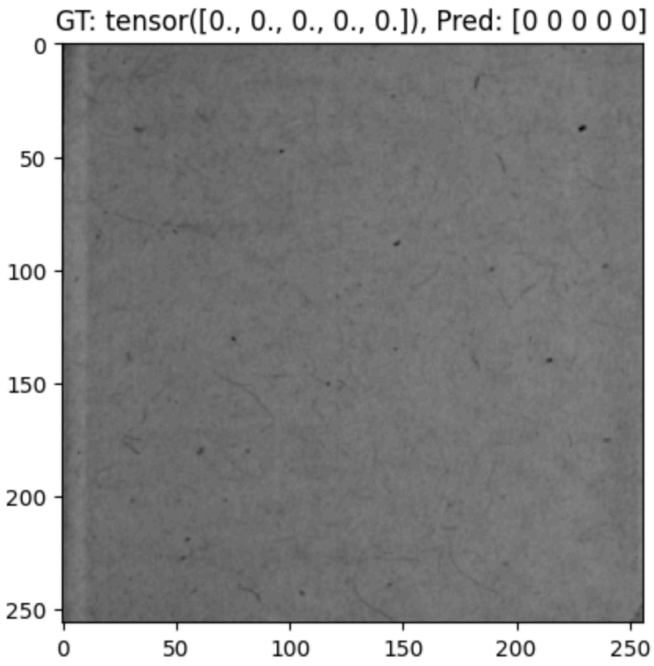


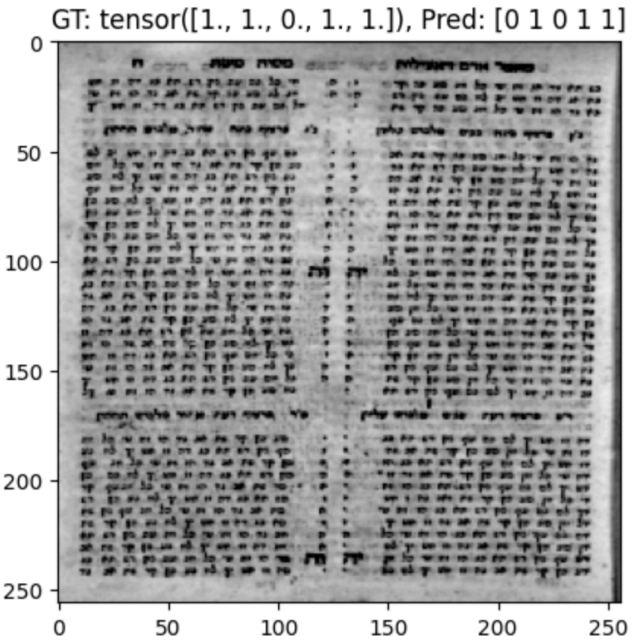


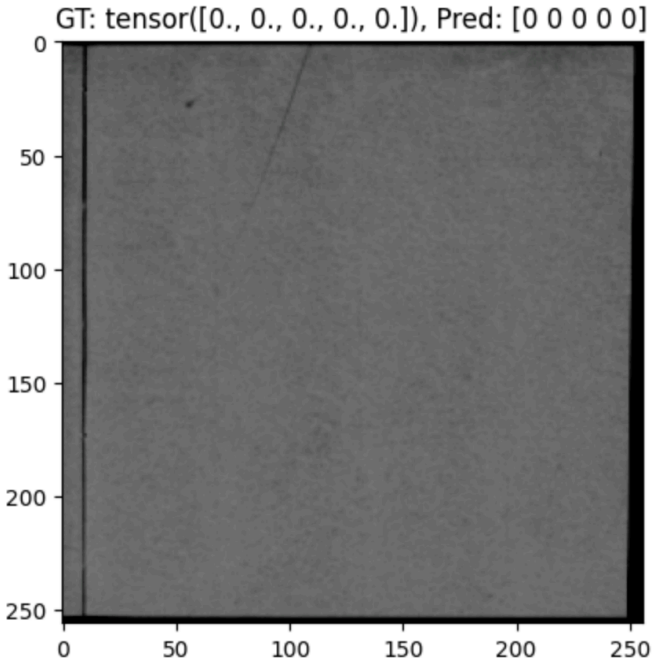
Error cases

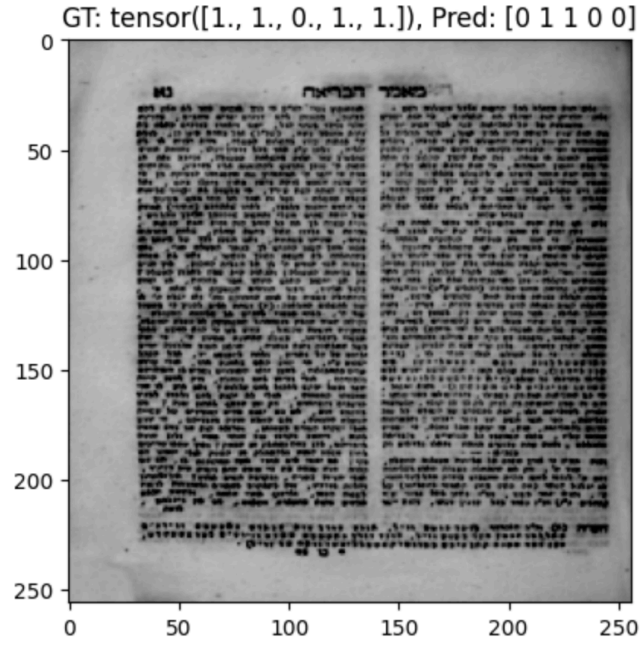


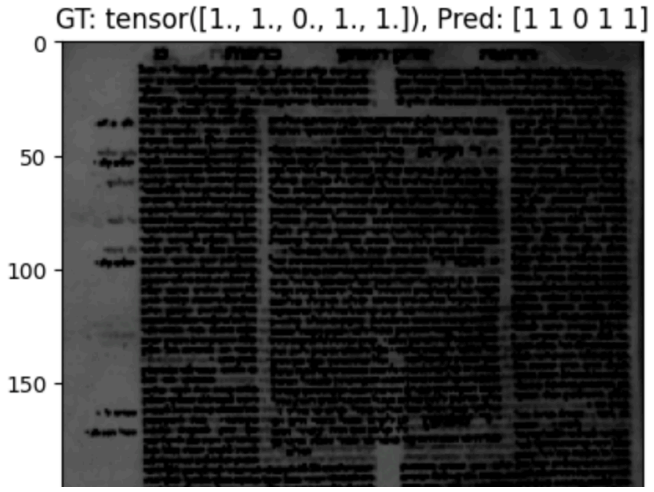


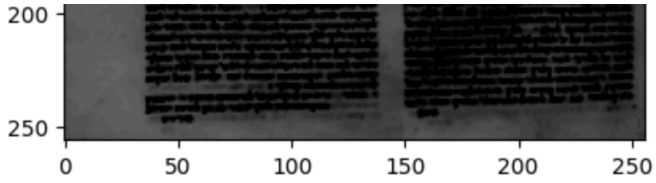

GT: tensor([0., 0., 0., 0., 0.]), Pred: [0 0 0 0 0]


GT: tensor([1., 1., 0., 1., 1.]), Pred: [0 1 0 1 1]

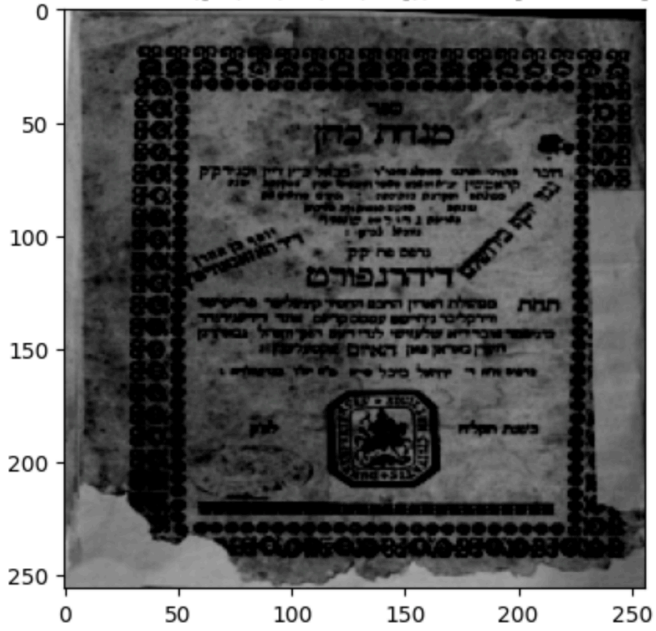

GT: tensor([0., 0., 0., 0., 0.]), Pred: [0 0 0 0 0]


GT: tensor([1., 1., 0., 1., 1.]), Pred: [0 1 1 0 0]


GT: tensor([1., 1., 0., 1., 1.]), Pred: [1 1 0 1 1]



GT: tensor([1., 0., 0., 0., 0.]), Pred: [1 0 0 0 0]



GT: tensor([0., 1., 1., 0., 0.]), Pred: [0 1 1 0 0]

GT: tensor([1., 1., 0., 1., 1.]), Pred: [1 1 0 1 1]