Overview:

The challenge involves classifying images into two categories: Road and Field. We have images for training and testing, organized into Train and Test folders. The Train folder has 108 road and 45 field images, with varying resolutions. The Test folder contains 9 unlabeled images. The goal is to create a model that accurately classifies these images into the two categories. I will use the Half Total Error Rate (HTER) matrix to measure the model's performance. This metric provides a clear view of how well the model identifies both classes while avoiding misclassifications.

Solution:

Deep learning approach using convolutional neural networks (CNNs) is used to tackle the problem. The model will include conv2d layers and fully connected layers, take the input is an image and predicts which class It belongs to. This means that if output value > 0.5 then the class of image is 1 and if output value is < 0.5, class is 0. Then, the model is trained and valuated using HTER. The target is to minimize this error rate.

Model is implemented, trained and validated using python in Google Colab.

$$FAR = FPR = FP/(FP + TN)$$
$$FRR = FNR = FN/(FN + TP)$$
$$HTER = (FAR + FRR)/2$$

Where:
FP: False positive
FN: False Negative
TN: True Negative
TP: True Positive

Implementation:

Dataset:

The dataset offered by the jury poses some challenges due to its relatively small size and imbalanced nature. In particular, there is an imbalance in the distribution of labels, with a significant majority of images labeled as "road," constituting around 70% of the dataset.

There are many methods to solve this problem such as up-sampling minor class, data augmentation, transfer learning, adaptable loss weight… Due to the limit of time and resources, I decided to use up-sampling method. This solution can increase the size of image and also make it more balance. After upsample more images both in 2 classes, the dataset now has 321 and 267 images in “class” and “field” respectively. The images were then resized to dimensions of (128, 128) and normalized to conform to the format of a torch tensor.

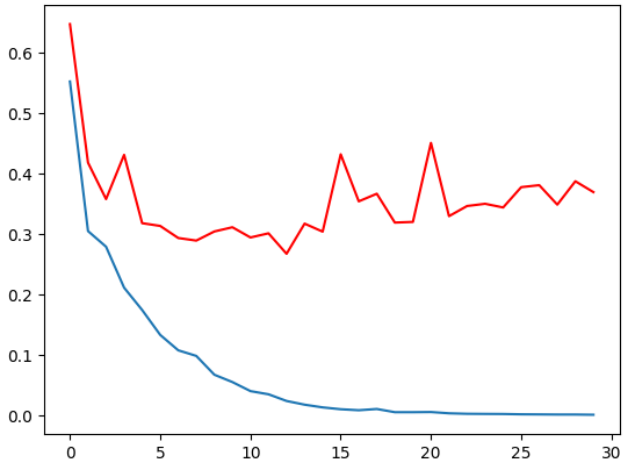
Hyperparameter:

Learning rate: 1e-5
BATCH_SIZE_VALIDATION = 8
BATCH_SIZE_TRAINING = 8
NUM_EPOCHS = 30
optimizer = Adam
loss_fn =BCEWithLogitsLoss (Binary cross entropy with sigmoid active function included)

Model:

1. First approach:

Model is constructed using 3 layers of CNN, followed by Maxpool layers. Each one of them using kernel size 5x5, and the output channels are 64, 128, 256 respectively. This is followed by 3 fully connected layers with output size are 1024, 128, 1 respectively.

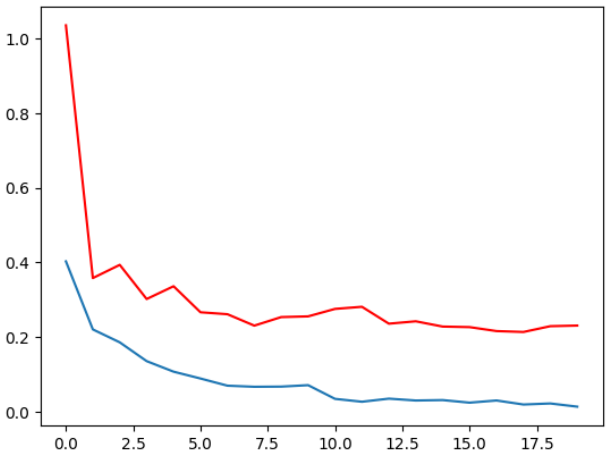


Blue: Training loss
Red: Validation loss
The model is overfitting since there is a big gap between validation loss and training loss.

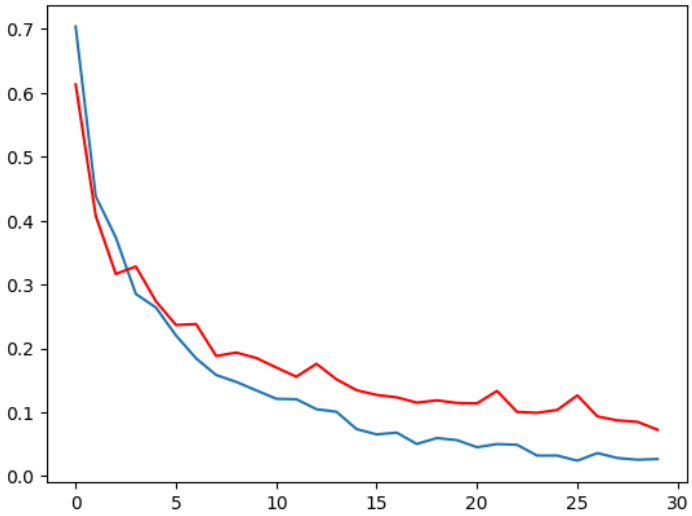
Solution: Reduce the complexity of the model.

2. Improvement:

I tried to reduce the complexity of the model by reducing output channel of conv2d layers 32, 64 and 128 respectively. One fully connected layer is removed and the model has 2 dense layers left with output size are 128, 1.



There is still 0.25 gap between validation and training loss. I applied some batch-norm after conv2 and reduce the dense layer output size to 64 and 1. Also dropout with p=0.2



FP = 0

TP = 48

FN = 2

TN = 67

HTER = 0.0196078431372549 ~ Accuracy 98%

The training curves exhibit a balance between overfitting and underfitting, accompanied by a commendable accuracy of 98%. Moreover, the predictions demonstrate a well-distributed classification between the two classes so I decided to stop the training here.

For further information, please refer to my implemented code. Colab

Future Improvement:

- Changing hyper parameter and the architecture of the network to reduce resources and enhance current results
- Data augmentation to up-sample data in class negative to balance the dataset
- Using transfer learning with some back-bone networks to solve the problem of small dataset.