- ## THE MAIN STEPS

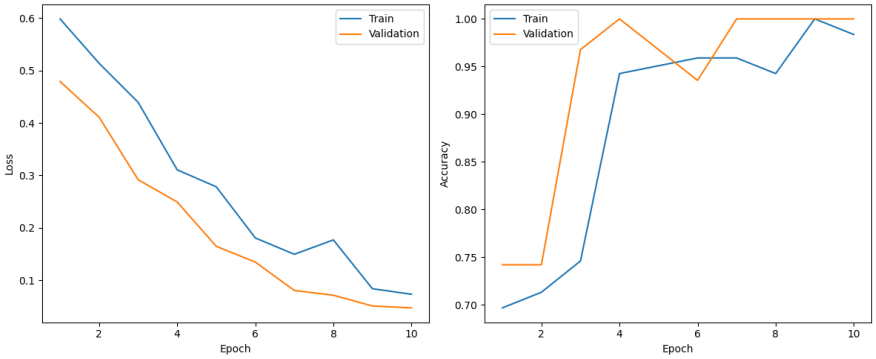
The process begins with the preparation of the data by organizing the images in two separate folders: "train" containing subfolders for the "Field" and "Road" classes, and "test_images" containing test images without predefined classes.I then viewed and analyzed all the "Field" and "Road" images and found that images ***"3.jpg" and "5.jpg" were stored in the "Field" folder even though they are "Road" images***. So I moved them to the "Road" folder.

Next, I used the pre-trained ResNet-50 model (I chose ResNet-50, over ResNet-100, because I took into account the complexity of the task, the availability of computing resources (GPU), and the size of the dataset), available in the PyTorch torchvision library. ResNet-50 is a deep convolutional neural network with 50 layers. It has been pre-trained on a large image database (ImageNet) and can extract useful features from our images. The task in this exercise is a binary classification task ("Field" and "Road"), I have replaced the last layer of the model (classification layer) to correspond to two classes. This means I've replaced the fully connected (FC) layer with a new FC layer with two output neurons.

I've divided the "train" folder into two parts: one for training and the other for validation. This allows me to check the quality of the model during training by verifying how it behaves on a data set it hasn't seen before (validation set).After that, I use data augmentation to increase the diversity of the training data. Images are resized, randomly cropped and subjected to a normalization transformation to prepare them for training.
I then use the training dataset to train the model.This involves running the training images through the model, calculating the loss (difference between model predictions and actual labels) and updating the model weights to minimize the loss.The model is optimized using the CrossEntropyLoss loss function and the SGD optimizer.During training, I monitor the learning curves for loss and accuracy on the training and validation set to detect any overlearning.

After training, I evaluate the model on the validation set, using a confusion matrix to assess its performance. Finally, we save the trained model for future use.
For test images without predefined classes, I use the pre-trained model and make predictions on each image individually.The results of the predictions are displayed, indicating whether each image is classified as "Field" or "Road".

- ## Result



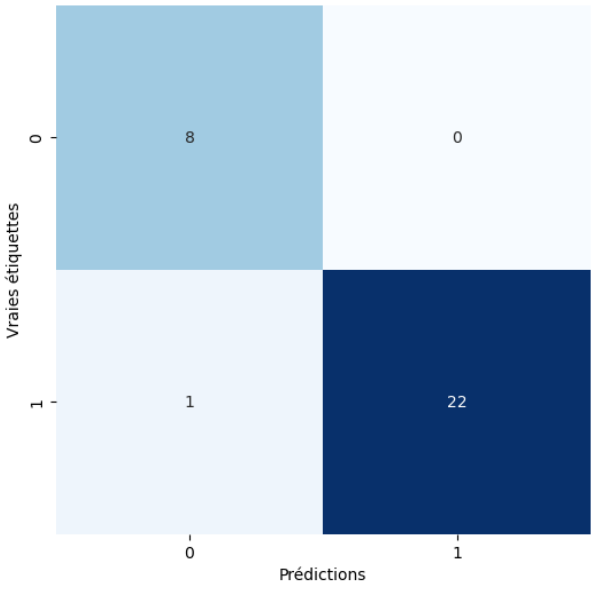
Training and test curves, also known as learning curves, show model performance over time, as the number of iterations (epochs) increases during training.

❖ Training curve (Training Loss / Training Accuracy):

At the start of training, loss is high and accuracy is low, as the model has not yet learned to generalize well. As training progresses, loss decreases and accuracy increases, indicating that the model is learning to fit the training data better.

❖ Test curve (Validation Loss / Validation Accuracy):

The validation set is used to assess the model's ability to generalize on new data that it has not seen during training.
Loss and accuracy on the validation set should follow a similar trend on the training set. However, the performance on the validation set starts to deteriorate.This may indicate overfitting, where the model does not generalize well to new data.



As for the confusion matrix . True positives (TP=8) represent correctly classified "Field" images, true negatives (TN=22) are correctly classified "Road" images.False positives (FP=0, i.e. none) are "Road" images wrongly classified as "Field", and false negatives (FN=1, i.e. only one) are "Field" images wrongly classified as "Road".