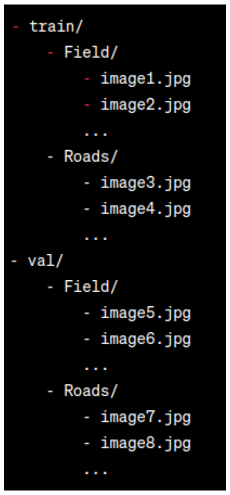
Classification: Field vs Road

I) Introduction:

The goal of the project is to develop an image classification algorithm using deep learning and computer vision techniques to classify fields and roads. Today, computer vision offers a wide range of applications and proposals for performing classification, segmentation, and object detection. Despite the wide range of open-source methods available, the engineer must adapt their strategy according to their problem and the complexity of their task. In my case, I had access to a small dataset.

II) Data:

The dataset was small and split into two folders: "train" and "val"(test_images). The "train" folder had two subfolders for the two classes, "Field" and "Roads". The "val" folder contained 10 images. To prepare for training, I sorted the "val" images into their respective classes, creating "Field" and "Roads" subfolders in the "val" directory. This gave the directory structure shown below:



```
- train/
    - Field/
        - image1.jpg
        - image2.jpg
        ...
    - Roads/
        - image3.jpg
        - image4.jpg
        ...
- val/
    - Field/
        - image5.jpg
        - image6.jpg
        ...
    - Roads/
        - image7.jpg
        - image8.jpg
        ...
```

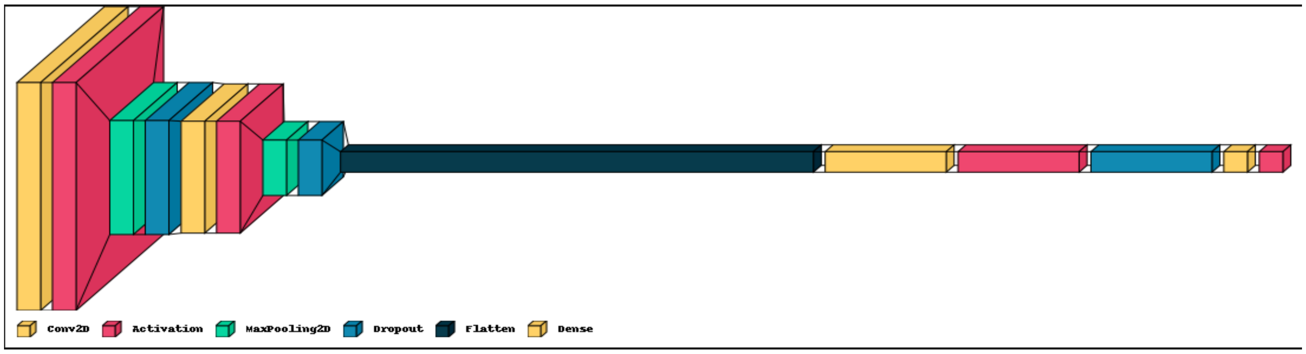
During my EDA, I found mislabeled images of roads in the "Field" folder and resized the images in the training set to a fixed size. The median size of the images was [340, 430] for roads and [183, 275] for fields, so i decided to resize the images to (224,224) with 3 channels. These steps help ensure consistency in the data and can improve the model's performance. In addition to the issues with mislabeled images and varying dimensions, I also found that the dataset was imbalanced. Specifically, there were 2.5 times more images of roads than images of fields. This class imbalance can potentially lead to biased model performance, as the model might become more adept at classifying the overrepresented class. To address the issue of class imbalance, I will be using the focal sigmoid loss function during training. This loss function is designed to give more weight to misclassified examples that are hard to classify correctly, thereby reducing the effect of the majority class and helping the model better learn to classify the minority class accurately. To add more variation to the data, I decided to use some data augmentations such as rotation_range, width_shift_range, height_shift_range, and of course, rescaling the images between 0 and 255.

Another more complex problem that requires careful consideration and problem formulation is the issue of roads running through fields. The question is whether such areas should be classified as roads or fields.

With all the biases inherent in the dataset and its small size, an appropriate model architecture must be chosen to address the problem.

III) Tested approaches & final model:

Aware that a small dataset requires an adapted architecture, I first tested several pre-trained models on the "ImageNet-1K" dataset: VGG16, MobileNet, ResNet18. I also tested SVM. Transfer learning with a dense layer without fine-tuning resulted in poor performance, with an accuracy of only about 58%. I attempted fine-tuning but due to time constraints, I was only able to achieve a maximum accuracy of 65%. Therefore, I decided to write my own architecture.



My model consists of two convolutional layers with a dropout regularization added to each layer to prevent overfitting due to the small dataset size and lack of data variability. This is followed by a single dense layer with 1024 neurons and another dropout. I used the ReLU activation function between the layers because it has been shown to work well in image classification tasks, and it allows the model to converge faster during training. I used a sigmoid activation function for the output layer. Sigmoid activation function is commonly used for binary classification as it maps the output to a probability between 0 and 1, which indicates the likelihood of the input belonging to a particular class. It is suitable for binary classification problems because it provides a probability for each class, with the complement of the output representing the probability for the other class. As mentioned earlier in this report i had an imbalanced class issue, so I used a focal sigmoid loss. I used stochastic gradient descent (SGD) as the optimizer with a learning rate of 0.001, which worked best for my problem based on quick tests.

IV) Conclusion:

With these approaches, I was able to obtain a fairly stable model for the data I had. However, due to time constraints, I was only able to work on it for about 10 hours. I would have liked to test other approaches such as few-shot learning, FCN, different architectures, and other augmentations. Nevertheless, I am convinced that with more data, I could achieve better results. I trained two different models, the first (V1) achieved 100% accuracy on the validation set, and the second (V2) achieved 90%. However, this is not representative as the validation set only consisted of 10 images. Nevertheless, I remain convinced that the V2 model is better and has better generalization performance.