Candidate Name: Salim Abboudi
Candidate email: salim1abboudi@gmail.com

Trimble / Bilberry : AI Engineer

Task1: Technical exercise

In this exercise, we develop a binary classifier called Rofie, using PyTorch to distinguish between field and road images. We explore model architecture, training loss, and hyperparameters, aiming to create an accurate and efficient classifier. The report provides a comprehensive overview of the process, presents results, and includes inferences on new data, offering valuable insights for model evaluation and future enhancements

1 Data

1.1 Dataset

The dataset contains a total of 151 images, with 108 images for fields and 43 images for roads. Developing a neural network classifier based on this limited and unbalanced dataset poses significant challenges, particularly in the context of potential overfitting. To address this, we initially explore the model's performance solely using the given data. This exploratory phase allows us to understand how well the classifier can distinguish between field and road images and assess the impact of limited data on model accuracy and generalization. As a next step, we plan to incrementally augment the dataset by adding more diverse data samples. This augmentation will help us evaluate the classifier's precision and robustness when exposed to a larger and more varied dataset, potentially improving its ability to generalize to new, unseen images. Through this incremental exploration, we aim to refine the *Rofie* classifier and achieve a more reliable and accurate classification performance.

1.2 Data Folder

To effectively manage our data, we devised a structured organization within the main *Data* folder, as shown in the following directory tree: Within the *Data* folder, we have three main subfolders: *train*, *validation*, and *test*. Each of these subfolders contains two subfolders: *field* and *road*, corresponding to the two classes we aim to classify. The *train* folder contains training images, while the *validation* folder holds validation images used to assess model performance. Finally, the *test* folder comprises test images to evaluate the final classifier's accuracy. This organized dataset structure simplifies data labeling and enables seamless integration with the data loader, facilitating the training and evaluation of the *Rofie* classifier.

```
Data
 ├─train
 │  ├─field
 │  └─road
 ├─validation
 │  ├─field
 │  └─road
 └─test
    ├─field
    └─road
```

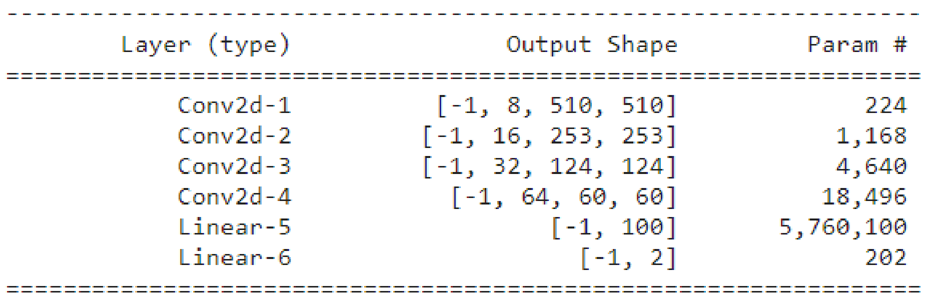
2 Methodology

2.1 Choice of Deep Learning Framework

For this project, we have chosen to utilize the PyTorch deep learning framework. PyTorch provides a dynamic computation graph, enabling easier debugging and flexible model architecture changes. It offers extensive support for GPU acceleration, resulting in faster model training and inference. Moreover, PyTorch's intuitive API and vast community support make it an ideal choice for rapid prototyping and experimentation.

2.2 Model Architecture Selection and Justification

Our CNN architecture, inspired from [4], is designed to learn hierarchical features from input images. To introduce non-linearities, we use ReLU activation functions, enhancing gradient propagation and training convergence. Max-pooling operations are applied to reduce spatial dimensions while retaining essential information.
To prevent overfitting, dropout layers are incorporated, randomly dropping neurons during training to enhance model generalization. Fully connected layers at the end perform the final classification, aggregating learned features for predictions.The final output is a tensor couple containg scores for each class
The selected model strikes a balance between complexity and performance, allowing the *Rofie* classifier to accurately distinguish between field and road images, in fig. 1 we depicts the CNN structure :



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 8, 510, 510]             224
            Conv2d-2         [-1, 16, 253, 253]           1,168
            Conv2d-3         [-1, 32, 124, 124]           4,640
            Conv2d-4           [-1, 64, 60, 60]          18,496
            Linear-5                  [-1, 100]       5,760,100
            Linear-6                    [-1, 2]             202
================================================================
```

Figure 1: Rofie Structure

2.3 Data Preprocessing and Augmentation Techniques

In training the binary classifier, data preprocessing and augmentations play crucial roles. Preprocessing involves resizing all images to a fixed size, ensuring uniformity. For augmentation, we utilize random horizontal flips and color jittering to introduce variety and simulate diverse lighting conditions [5]. These enrich the training data, prevent overfitting, and enhance the classifier's ability to handle real-world variations. Images are converted into tensors, necessary for deep learning models. These techniques collectively improve the classifier's performance and generalization. The Dataloader() class encompasses all required preprocessing steps.

2.4 Hyperparameter Tuning and Reasoning

2.4.1 Data Splitting

In the *Hyperparameter Tuning and Training Process*, we carefully divided the given training dataset into two sets: 20% for validation and 80% for training. This strategic division allows us to evaluate the model's performance on a separate set of data and prevents overfitting during the training process. By having a dedicated validation set, we can assess how well the model generalizes to new and unseen data, ensuring that our classifier is robust and effective.

2.4.2 Batch Size Selection

The selection of an appropriate batch size is crucial for efficient and effective training of the *Rofie* classifier. After careful consideration, we opted for a batch size of 16. This choice strikes a balance between computational efficiency and model convergence. Larger batch sizes can take advantage of parallelization and better utilize GPU resources, leading to faster training times. However, extremely large batch sizes may result in slower convergence and even overshooting the optimal solution. The selected batch size allows us to efficiently train the model while ensuring stable convergence and accurate predictions.

2.4.3 Learning Rate and Learning Rate Scheduler

For the optimization process, we employed the Adam optimizer with a learning rate of 3×10^{-4}. Additionally, we incorporated a learning rate scheduler from PyTorch known as *ReduceLROnPlateau* [3]. This scheduler dynamically adjusts the learning rate during training based on the model's performance on the validation set. By setting some parameters, the scheduler fine-tunes the learning rate to avoid overshooting the optimal value and ensure smoother convergence. This adaptive learning rate strategy allows us to optimize the classifier efficiently and enhance its accuracy on both training and validation data.

2.4.4 Choice of Loss Function

In the *Rofie* classifier, we carefully selected the Negative Log-Likelihood Loss (*NLLloss*) [6] with a sum reduction as our loss function. The *NLLLoss* is particularly well-suited for classification tasks, especially binary classification as in our case. It calculates the negative log-likelihood of the predicted probabilities for each class and penalizes the model when the predicted probability for the correct class is low. This encourages the model to improve its class separation and increases its confidence in making accurate predictions. By using the *NLLLoss*, we guide the model to learn the most discriminative features for distinguishing between field and road images, leading to a robust and precise classification performance.

2.4.5 Training image size

The image size (imgsize) is a critical parameter during the training process of the *Rofie*classifier. In the Dataloader, images are resized to a fixed imgsize to ensure data consistency and facilitate the neural network's computation. When selecting the imgsize, one must strike a balance between capturing sufficient information from the input data and avoiding overfitting, especially when dealing with a limited dataset like the one in the binary classification task of road and field images. On one hand, a larger imgsize allows the model to process more data and potentially capture more intricate features, which can lead to better performance. On the other hand, a larger imgsize increases the memory requirements and computational complexity, and it can potentially exacerbate overfitting, especially when the dataset is not extensive enough to support the additional information. After careful experimentation and analysis, an imgsize of **640** pixels has been found to be the optimal choice for the *Rofie* classifier.

2.5 Training and Validation Process

In fig. 2, we present the general workflow of the classification task. Data is first loaded and transformed (random rotation, flip, brightness, etc.) For validation, we only transform to tensors. The set of hyperparameters is chosen as explained in the section above. Forward and backward propagation are performed to update the weights and improve performance to finally get the optimized model. The entire procedure to train and test data is explained in the Git repository (click here) and import necessary packages.

3 Training Results

The plot in fig. 3 illustrates the accuracy and loss metrics across training epochs for a model. Fluctuations in both metrics are common due to the iterative nature of neural network optimization. Despite these fluctuations, there is an overall increasing trend in accuracy and a decreasing trend in loss, indicating the model's improvement over epochs. The increasing accuracy suggests learning from the training data, while the decreasing loss indicates improved convergence. Validation accuracy reaches approximately 80%, demonstrating the model's ability to make accurate predictions on unseen data. However, the model's performance can be further enhanced by addressing potential overfitting through the addition of diverse data samples and fine-tuning hyperparameters.

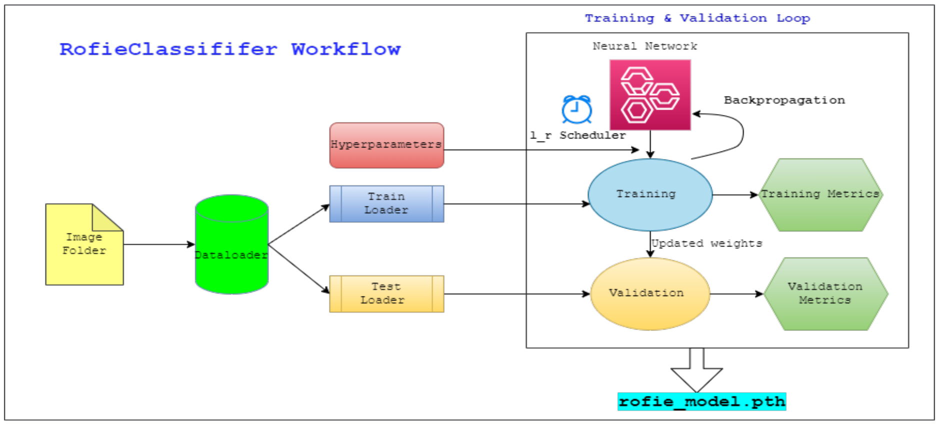


Figure 2: Training Pipeline Workflow

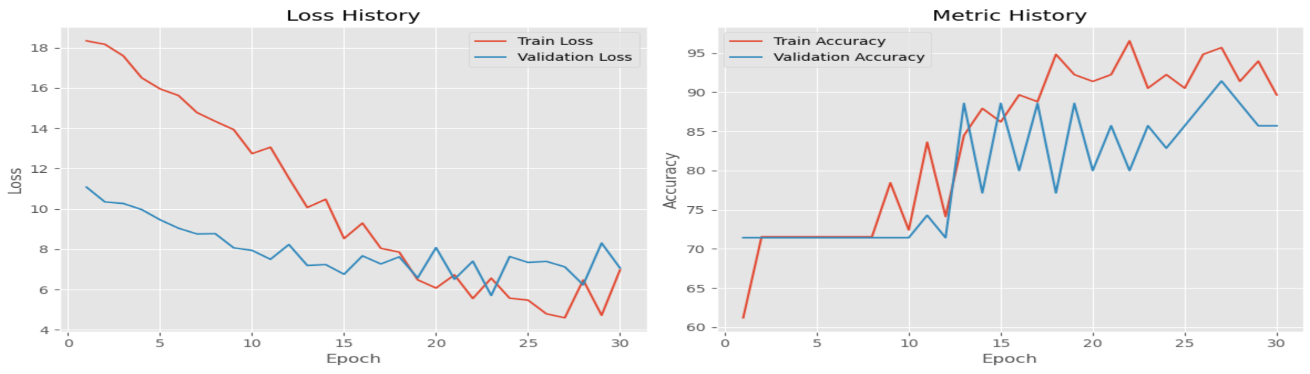


Figure 3: training results

4 Test Results

we can see in fig. 4 that results are pretty satisfying, we got 7/10 good predictions, some images can be tricky, the footpath is misleading,that's why we will improve the model by adding new varied images

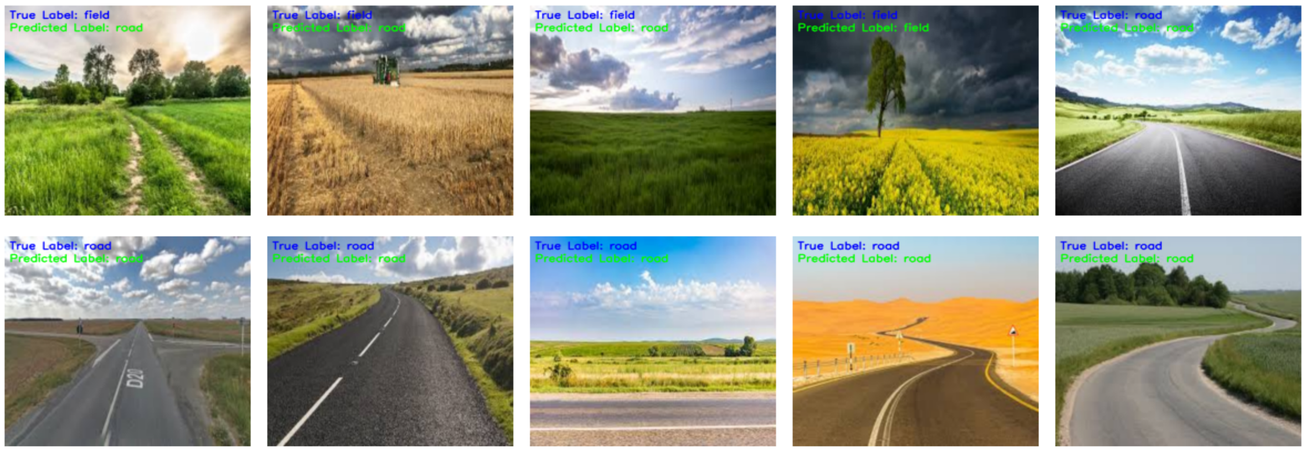


Figure 4: Inferences results

5 Model Improvement

Two reasons made us think of augmenting our dataset

- The raw data provided exhibits a significant class imbalance, with 70% representing roads and only 30% representing fields. This imbalance poses a considerable challenge in binary classification tasks [2], requiring us to address it adequately to ensure accurate results. One approach to mitigate this issue is to augment the dataset by adding more data points for the minority class, in this case, fields. By doing so, we can balance the representation of both classes and enhance the performance

- Moreover, the complexity of the given data is evident, as a majority of the images appear to be misleading. To improve the model's ability to generalize, it is crucial to introduce more diverse and varied images. For instance, incorporating images that depict footpaths in fields or scenes with shrubs or trees alongside roads will help the model better grasp the subtle differences between roads and fields, leading to more accurate and robust classifications. Additionally, we also added some mosaic augmentations [1] by combining multiple images in our dataset to create larger and more realistic scenes, simulating complex scenarios that the model may encounter in real-world applications

We ran the same training pipeline, but this time we trained and validated on a new dataset, which can be found in the Git repository (data). To balance the dataset, we added images of both roads and fields, resulting in 155 images per class. The impact of this simple adjustment can be observed in table 1, where the model's accuracy significantly improved when tested on the first validation set. Moreover, the inclusion of more images in the new dataset enhanced the model's robustness and generalization capabilities.
NOTE that we did not extensively explore this aspect since our focus was on fine-tuning hyperparameters and developing the optimal model while avoiding overfitting. However, if we were to further optimize our model, we believe having at least 500 images for each class would be a crucial requirement.
We can see in fig. 5 the variety of data used to augment the model as well as its good predictions

Table 1: Accuracy Comparison

	Original validation set	Augmented Validation set
Original Model	70.18%	80.56%
Enhanced Model	77.67%	83.71%

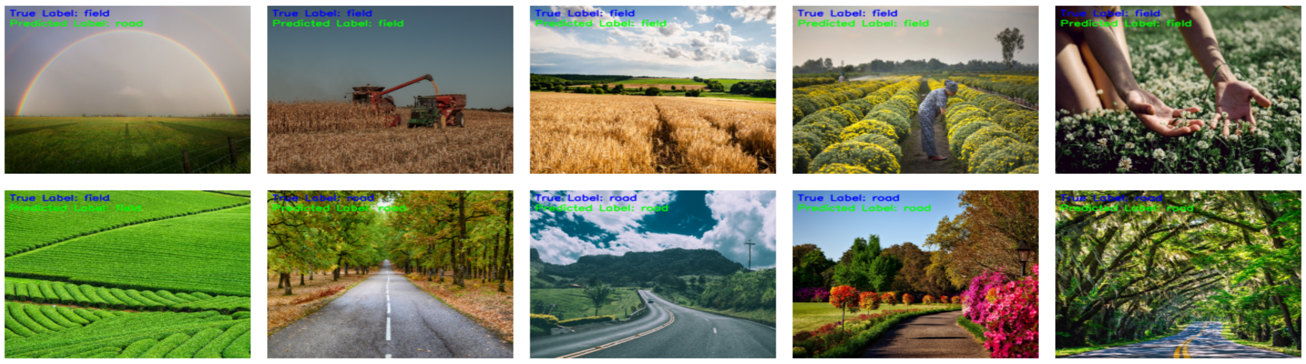


Figure 5: Prediction on new data

6 Conclusion

In this project, we developed a binary classifier called *Rofie* using PyTorch for field and road image classification. The dataset's limited size and class imbalance presented challenges, but incremental data augmentation improved generalization. PyTorch's dynamic computation graph and CNN architecture with ReLU activations were utilized. Data preprocessing and augmentations, including flips and color jittering, enriched the training data. Hyperparameter tuning involved selecting batch size, learning rate, and loss function (Negative Log-Likelihood Loss). The model demonstrated accurate results on training and validation data, but further dataset augmentation improved performance. Future work involves exploring additional data augmentation techniques and expanding the dataset for better classification.

References

[1] Wang Hao and Song Zhili. Improved mosaic: Algorithms for more complex images. In *Journal of Physics: Conference Series*, volume 1684, page 012094. IOP Publishing, 2020.

[2] Zeju Li, Konstantinos Kamnitsas, and Ben Glocker. Overfitting of neural nets under class imbalance: Analysis and improvements for segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2019: 22nd International Conference, Shenzhen, China, October 13–17, 2019, Proceedings, Part III 22*, pages 402–410. Springer, 2019.

[3] Leonie Monigatti. A visual guide to learning rate schedulers in pytorch. *Medium*, Dec 2022.

[4] ANDREY SHTRAUSS. Histopathologic cancer detection. `https://www.kaggle.com/code/shtrausslearning/pytorch-cnn-binary-image-classification/notebook#8-|-Loss-Function-Definition`, 2022. Kaggle Notebook.

[5] Mingle Xu, Sook Yoon, Alvaro Fuentes, and Dong Sun Park. A comprehensive survey of image augmentation techniques for deep learning. *Pattern Recognition*, page 109347, 2023.

[6] Donglai Zhu, Hengshuai Yao, Bei Jiang, and Peng Yu. Negative log likelihood ratio loss for deep neural network classification. *arXiv preprint arXiv:1804.10690*, 2018.