MGI Project Report

Group 10

GRS34806: Deep Learning

Authors:
Thomson Chow 1245201
Takayuki Ishikawa 1372351

Wageningen University

WAGENINGEN
UNIVERSITY & RESEARCH

Contents

1 Introduction

1.1. Research question

- Which fine-tuned model yields the most accurate predictions for land-use classification in high-resolution satellite imagery when applied to small datasets?

The rapid development of remote sensing technology has opened new opportunities for land-use classification, critical in urban planning, environmental monitoring, and resource management. Yet, manual analysis is impractical due to the vast and dynamic data. Deep learning, particularly through fine-tuning, addresses this challenge by reusing pre-trained models for new tasks, reducing the need for large labeled datasets and computational resources.

This project investigates the effectiveness of fine-tuning various pre-trained models on the UC Merced (UCM) Land Use Dataset, which includes 21 distinct land-use classes [YN10]. It also examines a multi-label extension, where each image contains multiple land-use classes, mimicking real-world scenarios with overlapping land-use patterns [Cha+18]. The study evaluates a range of pre-trained models, and to benchmark the benefits of fine-tuning, a comparison is made with several models trained from scratch.

This report aims to identify optimal strategies for applying transfer learning to land-use classification, considering both single-label and multi-label contexts. It provides insights into the adaptability of different pre-trained models and offers guidance for improving model performance.

2

Methods

This chapter outlines the methodology adopted in this project, including the datasets used, the pre-processing techniques, the deep learning models employed, both pre-trained and custom, and the strategies for fine-tuning. The evaluation metrics and experimental settings are also described to provide a comprehensive overview of the approach to solving the research question.

2.1. Dataset Description

The study uses the UC Merced (UCM) Land Use Dataset, containing 21 distinct land-use classes with 100 high-resolution images in each category (Task 1, see Figure A.1 in appendix). Each image has a 256 x 256 resolution with 0.3-meter spatial resolution, totaling 2100 images.

For the multi-label scenario (Task 2), the UCM dataset extension developed by Chaudhuri et al. (2018) is used. This dataset identifies 17 common object classes in the original 2100 images, with each image annotated with 1 to 7 object classes (see Figure A.5 in Appendix A). This leads to an imbalanced distribution of annotations across the dataset (see Table 2.1). This imbalance requires careful consideration during model training to ensure accurate predictions.

(a) The number of images within each annotation

Class	Number of Images
airplane	100
bare-soil	718
buildings	691
cars	886
chaparral	115
court	105
dock	100
field	103
grass	975
mobile-home	102
pavement	1300
sand	294
sea	100
ship	102
tanks	100
trees	1009
water	203

(b) Number of images by annotation count

Annotation Count	Number of Images
0	0
1	224
2	426
3	512
4	442
5	365
6	119
7	12
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0

Table 2.1: Annotation status for task 2

2.2. Pre-processing and Data Augmentation

To address the challenge of overfitting due to the limited size of the dataset, comprehensive data augmentation strategies are employed. Table A.2 in Appendix A outlines the specific transformations used for data augmentation, these transformations include random resized cropping, horizontal flipping, and random adjustments to contrast. These transformations not only augment the training data but also simulate varying conditions that satellite images might exhibit, thus enhancing the robustness of the models trained. The specifics of these transformations and their parameters are meticulously optimized based on preliminary experiments using a Custom CNN Model. During training of the baseline models, all models used the same transformations and parameters. For fine-tuning models, some model-specific transformations are additionally tested.

2.3. Model Selection and Architecture

We tried several models in task 1 (See Table 2.2) and task 2 (See Table 2.3). In each task, we experiment two training by different setting.

2.3.1. Baseline Models

A variety of baseline models are evaluated:

- Custom CNN (see A.1 in Appendix for its architecture): Designed specifically for this project with a balance between depth and computational efficiency.
- LeNet-5, MobileNet V3: Lightweight models for comparative baseline performance.
- AlexNet, ResNet-18, ResNet-50: Deeper networks to explore the benefits of increased model complexity.

2.3.2. Fine-tuning Pre-trained Models

Several state-of-the-art models are selected for fine-tuning to leverage their pre-trained weights which have been trained on large diverse datasets:

- MobileNet V2, ResNet-50, Vision Transformer (ViT) B/16-base: These models are fine-tuned where only the classifier layers are trained while the other layers retain their pre-trained weights to see how well the features learned from different domains adapt to satellite imagery.

Model	Total Parameters	Trainable	Non-trainable
Baseline models			
Custom CNN	16,785,141	16,785,141	0
LeNet-5	7,395,421	7,395,421	0
MobileNet v3 small	1,539,381	1,539,381	0
AlexNet	57,089,877	57,089,877	0
ResNet-18	11,187,285	11,187,285	0
ResNet-50	23,551,061	23,551,061	0
Fine-tuned models			
MobileNet v2	2,390,549	166,677	2,223,872
MobileNet v3 small	1,003,573	76,565	927,008
ResNet-50	23,773,013	4,727,573	19,045,440
ViT	85,945,877	101,141	85,844,736

Table 2.2: Models used in task 1

2.4. Training Procedure

All training was conducted on Google Colab with Nvidia T4 GPU using Pytorch [Pas+19]. The models are trained using an 80-20 split for training and testing, with 80% of the data used for training, 10% for validation, and 10% for testing. The images were resized to 256 x 256 pixels, with batch size of 8. The Adam optimizer is used due to its effectiveness in handling sparse gradients and adaptive learning rate capabilities.

Model	Total Parameters	Trainable	Non-trainable
Baseline models			
Custom CNN	16,784,625	16,784,625	0
LeNet-5	7,395,081	7,395,081	0
MobileNet v3 small	1,535,281	1,535,281	0
Fine-tuned models			
MobileNet v2	2,390,033	166,161	2,223,872
MobileNet v3 small	1,003,057	76,049	927,008
ResNet-50	23,772,497	4,727,057	19,045,440
ViT	85,945,361	100,625	85,844,736

Table 2.3: Model used in task 2

In the first training of task 1, we used a learning rate of 0.00080 for baseline models, which is optimized for the Custom CNN model, and 0.0001 with dropout rate 0.5 for fine-tuned models. In the second training, we changed to a constant learning rate 0.0001 and included a dropout rate of 0.1 for the baseline models. Training continued for 100 epochs.

In first training of task 2, we used a constant learning rate of 0.001 and no dropouts for baseline models. Training continued for 50 epochs, or until the network converged. If the validation loss did not decrease for 10 consecutive epochs, it is determined that the model has converged and the training was stopped early.

In the second training for task 2, in addition to above setting, we experiment 100 epochs and an adaptive learning rate initially set to 0.001, then divided by 10 when the validation loss did not decrease for 10 consecutive epochs. If the validation loss did not decrease for 20 consecutive epochs, it is determined that the model has converged and the training was stopped early.

Other parameters of the optimizer are set to the default: for example $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The weights in pre-trained models are set as non-trainable except the second training for task 2. The other weights are initialized by Xavier uniform initializer [GB10].

2.5. Evaluation Metrics

For single-label classification, validation loss and accuracy serves as the primary metric. For the multi-label scenario, precision, recall, and F1-score are used to provide a holistic view of model performance across the imbalanced classes. These metrics help in understanding the effectiveness of each model in distinguishing between the various land-use types accurately.

3

Results and Discussion

This chapter presents and analyzes the results obtained from the project's two tasks, focusing on the performance of various models in single-label and multi-label land-use classification using deep learning.

3.1. Results for Task 1: Single-label Classification

In Task 1, the first set of trainings used same augmented data for all models and it highlighted (see Table 3.1) substantial differences in model effectiveness. All models achieved reasonable levels of performance with accuracies exceeding 70% once a suitable learning rate was used.

Then, fine-tuned models outperform baseline models trained from scratch, and achieved higher Accuracy with over 90%. The ViT model has 97% accuracy and this is the highest one.

In the first training, we tested with custom CNN models and got similar results between our custom CNN model and LeNet-5. Using MobileNet v3 small model we got a very low accuracy as the training rate is too high.

Model	Validation Loss	Accuracy	Training Time
Baseline models			
Custom CNN	1.300	0.633	6 minutes
LeNet-5	1.251	0.671	5 minutes
Mobilenet v3 small	3.056	0.033	5 minutes
Fine-tuned models			
Mobilenet v2	0.2652	0.9048	20 minutes
ResNet-50	0.1803	0.9476	21 minutes
ViT	0.1219	0.9714	40 minutes

Table 3.1: Performance of both baseline and fine-tuned models during the first training phase of task 1

In subsequent trainings (see Table 3.2), adjustments by reducing the learning rate (0.0008 to 0.0001) and adding a dropout rate (0.1) in baseline models significantly improved their performances, particularly for MobileNet v3 small. Fined-tuned models used their respective inference transforms instead of the same data transformation with baseline models, then most of them reached an accuracy of over 97%.

Among the baseline models tested, ResNet-18 stood out, likely due to its more recent architecture and relatively shallower depth compared to ResNet-50, making it particularly suitable for smaller datasets. This illustrates the benefit of using newer, more efficient architectures when handling limited data.

Among the fine-tuned models, MobileNet v3 small, ResNet-50, and Vision Transformer B/16-base all achieved comparable accuracies, each surpassing the 97% mark. However, it's important to note that

the training duration for the Vision Transformer was significantly longer than for the other models, which might be a consideration in practical applications where training time is a constraint.

In order to compare fine-tuned models with baseline models accurately, we trained MobileNet v3 small and ResNet-50 from both scratch and pre-trained. This results in the fact that pre-trained model got much higher accuracy than the model trained from scratch even though the model structure is the same.

Another finding is that fine-tuning model has lower validation loss and higher accuracy than training loss and accuracy (See A.2 and A.4) .

Model	Validation Loss	Accuracy	Training Time
Baseline models			
MobileNet v3 small	1.045	0.729	5 minutes
AlexNet	0.741	0.790	5 minutes
ResNet-18	0.537	0.852	6 minutes
ResNet-50	0.839	0.748	8 minutes
Fine-tuned models			
Mobilenet v2	0.1765	0.9571	21 minutes
MobileNet v3 small	0.1491	0.9714	21 minutes
ResNet-50	0.2441	0.9714	21 minutes
ViT	0.0959	0.9714	40 minutes

Table 3.2: Performance of both baseline and fine-tuned models during the second training phase of task 1

3.2. Results for Task 2: Multi-label Classification

In first training for task 2, contrary to task 1, basic models performed far better than fine-tuned models with freezed pre-trained weight and 50 epochs (Table 3.3). Mobilenet v3 small trained from scratch has F1 Score of 73%, which is the highest one. Pre-trained models have quite low F1 Score with less than 10%.

Model	Validation Loss	F1 Score	Recall	Precision	Training Time
Baseline models					
Custom CNN	0.2490	0.5531	0.5117	0.6353	8 minutes
LeNet-5	0.2531	0.5789	0.5500	0.7151	8 minutes
MobileNet v3 small	0.1875	0.7297	0.7033	0.7840	12 minutes
Fine-tuned models					
MobileNet v2	0.3754	0.0658	0.0723	0.0748	11 minutes
ResNet-50	0.4015	0.0889	0.0925	0.0990	11 minutes
ViT	0.3745	0.0703	0.0747	0.1308	18 minutes

Table 3.3: Performance of both baseline and fine-tuned models during the first training phase of task 2

Then, we set 100 epochs and changed the setting in the way that the weights of pre-trained model were unfreezed (initialized with pre-trained weights but updated during the training phase) and the learning rate were divided by 10 when the validation loss didn't decrease in 10 consecutive epochs. This setting was applied in similar research such as [HMZ20].

In this new setting, all of models accuracy increased (Table 3.4). Pretrained models get much better results than previous setting, with fine-tuned MobileNet v3 small model getting the highest F1 Score. Since the ResNet-50 and ViT models have over 23M and 85M parameters respectively and the dataset is small, these model's results are still low comparing to other models.

Although baseline models increase the F1 score, these improvements are not larger than those in pre-trained models.

We trained MobileNet v3 small from both scratch and pre-trained. This results in the fact that pre-trained model got slightly higher accuracy than the model trained from scratch even though the model structure

is the same.

Another finding is that fine-tuning model has lower validation loss and higher accuracy than training loss and accuracy (See A.6 and A.7) .

Method	Validation Loss	F1 Score	Recall	Precision	Training Time
Baseline models					
Custom CNN	0.2329	0.5905	0.5274	0.7697	13 minutes
LeNet-5	0.2172	0.6463	0.6116	0.7642	11 minutes
MobileNet v3 small	0.1212	0.8846	0.8691	0.9048	26 minutes
Fine-tuned models					
MobileNet v2	0.1556	0.8076	0.7747	0.8532	23 minutes
MobileNet v3 small*	0.1072	0.9118	0.8900	0.9393	21 minutes
ResNet-50	0.2358	0.5751	0.5239	0.6926	37 minutes
ViT	0.3451	0.1425	0.1474	0.1541	147 minutes
ViT*	0.3146	0.2389	0.2467	0.2884	147 minutes

Table 3.4: Performance of both baseline and fine-tuned models during the second training phase of task 2. Note: models followed by * are trained using their respective inference transforms

3.3. Discussion

3.3.1. Analysis of Single-label Classification

In task 1, [Ges22] reported a 100% accuracy with this task and [Neu+19] reached 99.61% using a ResNet-50 model. Our current result of 97.14% is comparable to [Cas+15]. The effectiveness of fine-tuned models in Task 1 suggests that even when not trained on satellite imagery, pre-trained models can adapt well to new domains. This adaptability is attributed to the diverse and extensive data on which these models were originally trained.

Despite these high accuracy levels, there is room for further enhancement through more extensive hyperparameter tuning and additional data augmentation strategies. The current experimental setup did not provide sufficient data to definitively conclude which model architecture is the best, indicating a need for further exploration to optimize and fully assess each model's capabilities.

3.3.2. Analysis of Multi-label Classification

In this task [HMZ20] and [HZH21] reached 88.72% and 91.74% in F1 Score with their models. We came close with baseline and fine-tuned Mobilenet v3 small models. However, our other fine-tuned models have far lower F1 Score. We assumed that this result was caused by three reasons. First, pre-trained models are specialized not for multi-label, but single-label scene classification. Second, our training dataset has unbalanced annotation data. some annotations such as tree have sufficient images but others have only 100 annotation, then this unbalance could cause some inaccuracy. Third, some models were not trained using their inference transformers.

3.3.3. Variability and Reproducibility

During our experiments, we observed significant variability in model performance that could be influenced by various factors, such as random weight initialization and small changes in training conditions. To address this, we experimented with setting manual seeds to maintain consistency across training runs. Despite these efforts, the inherent randomness in deep learning processes still led to some fluctuations in results.

Ideally, to ensure the robustness and reliability of our models, a statistical hypothesis test would be employed to determine the number of model runs sufficient to achieve statistically significant results. We noticed [HZH21] trained each models in 10 times. This approach would involve training multiple instances of each model under the same conditions and then using a hypothesis testing framework to compare their performances. The goal would be to establish a confidence interval for the performance metrics.

3.3.4. Hyperparameter Optimization

Hyperparameter optimization played a crucial role in the effectiveness of the models used in our experiments. For instance, for the custom CNN model, we focused on optimizing the learning rate and dropout rate to enhance performance. We utilized the Optuna optimization framework [Aki+19], which employs the Tree-structured Parzen Estimator algorithm by default, to efficiently navigate the search space of potential hyperparameter values. This method proved beneficial in identifying optimal settings that significantly influenced model outcomes. However, to further improve reproducibility and ensure the robustness of our results, a more extensive hyperparameter optimization strategy should be adopted. Employing a grid search methodology could be advantageous.

Additionally, implementing an adaptive learning rate and a warm-up phase in the training process could lead to better convergence behaviors and more stable training dynamics. Learning rate schedulers adjust the learning rate throughout training based on performance, potentially leading to improved model accuracy and training efficiency. The warm-up phase, where the learning rate gradually increases to its initial maximum, can help mitigate the issue of model training starting from a sub-optimal point due to random initialization.

3.3.5. Test Data Usage

In this project, 10% of the total dataset was reserved as test data, intended to provide an unbiased evaluation of the final model's performance. However, due to time constraints, we focused our analysis predominantly on the validation set results and did not utilize the test set for final performance evaluation.

3.3.6. Code and Model Organization

Throughout this project, organizing the code and models efficiently presented several challenges. We encountered issues where certain functions were applicable only to Task 1 and not to Task 2, or vice versa. This situation led to improvements in specific models that were not consistently applied across all experiments, resulting in a lack of uniformity and making direct comparisons between models less reliable.

To address these organizational challenges, adopting Object-Oriented Programming (OOP) principles could be highly beneficial. By designing reusable classes and functions, we can ensure that improvements and modifications made for one model can be easily applied to others. Furthermore, the use of advanced tools like TensorBoard and Weights & Biases (WandB) for logging experiments and tracking model performance could significantly improve project organization.

3.3.7. Training-Test Split

Exploring different data splits, such as a 50-50 training-test split, could provide insights into the models' generalization capabilities beyond what the current 80-20 split has revealed.

3.4. Conclusion

The study highlights the nuanced performance of deep learning models in satellite image classification. Task 1 results indicate a clear benefit of using fine-tuned models over baselines, while Task 2 illustrates the complexities of multi-label settings. The findings suggest that with optimal tuning and training strategies, both tasks show potential for high accuracy, but reproducibility and model consistency remain areas for further study and improvement.

Notably, the baseline MobileNet v3 small model perform worse in task 1 than in task 2, which is supposedly more difficult. It is very likely that by using learning rate schedulers and further optimising the hyperparameters, the accuracy of baseline models in task 1 can also be improve to over 85%.

4

Implementation

4.1. Link to Jupyter Notebook

https://drive.google.com/file/d/1QNSz1yGUwGPYIS9gcTzscMRArfwU_tAX/view?usp=sharing

References

[Aki+19] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.

[Cas+15] Marco Castelluccio et al. *Land Use Classification in Remote Sensing Images by Convolutional Neural Networks*. 2015. arXiv: 1508.00092 [cs.CV].

[Cha+18] Bindita Chaudhuri et al. “Multilabel Remote Sensing Image Retrieval Using a Semisupervised Graph-Theoretic Method”. In: *IEEE Transactions on Geoscience and Remote Sensing* 56.2 (2018), pp. 1144–1158. DOI: 10.1109/TGRS.2017.2760909.

[GB10] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html.

[Ges22] Andrea Gesmundo. *A Continual Development Methodology for Large-scale Multitask Dynamic ML Systems*. 2022. arXiv: 2209.07326 [cs.LG].

[HMZ20] Yuansheng Hua, Lichao Mou, and Xiao Xiang Zhu. “Relation Network for Multilabel Aerial Image Classification”. In: *IEEE Transactions on Geoscience and Remote Sensing* 58.7 (July 2020), pp. 4558–4572. ISSN: 1558-0644. DOI: 10.1109/tgrs.2019.2963364. URL: http://dx.doi.org/10.1109/TGRS.2019.2963364.

[HZH21] Rui Huang, Fengcai Zheng, and Wei Huang. “Multilabel Remote Sensing Image Annotation With Multiscale Attention and Label Correlation”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* PP (June 2021), pp. 1–1. DOI: 10.1109/JSTARS.2021.3091134.

[Neu+19] Maxim Neumann et al. *In-domain representation learning for remote sensing*. 2019. arXiv: 1911.06721 [cs.CV].

[Pas+19] Adam Paszke et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019. arXiv: 1912.01703 [cs.LG].

[YN10] Yi Yang and Shawn D. Newsam. “Bag-of-visual-words and spatial extensions for land-use classification.” In: *GIS*. Ed. by Divyakant Agrawal et al. ACM, 2010, pp. 270–279. ISBN: 978-1-4503-0428-3. URL: http://dblp.uni-trier.de/db/conf/gis/gis2010.html#YangN10.

A

Figures

A.1. Task 1

Figure A.1: 21 land-use classes

Layer (type)	Output Shape	Number of Parameters
Custom CNN Model	[8, 21]	-
├─Sequential (features)	[8, 32, 64, 64]	-
│ └─Conv2d (0)	[8, 16, 256, 256]	448
│ └─ReLU (1)	[8, 16, 256, 256]	-
│ └─MaxPool2d (2)	[8, 16, 128, 128]	-
│ └─Dropout (3)	[8, 16, 128, 128]	-
│ └─Conv2d (4)	[8, 32, 128, 128]	4,640
│ └─ReLU (5)	[8, 32, 128, 128]	-
│ └─MaxPool2d (6)	[8, 32, 64, 64]	-
│ └─Dropout (7)	[8, 32, 64, 64]	-
├─Flatten (flatten)	[8, 131072]	-
├─Sequential (classifier)	[8, 21]	-
│ └─Linear (0)	[8, 128]	16,777,344
│ └─ReLU (1)	[8, 128]	-
│ └─Dropout (2)	[8, 128]	-
│ └─Linear (3)	[8, 21]	2,709

Table A.1: Custom CNN Architecture

Transformation name	Type	Value range	Best
RandomResizedCrop	boolean	[True, False]	True
- Scale (Min, Max)	continuous	([0.08, 1.0], log [Min, 1.0])	(0.6162, 0.7409)
- Ratio (Min, Max)	continuous	([0.75, 1.0], [1.0, 1.33])	(0.9340, 1.1143)
RandomHorizontalFlip	boolean	[True, False]	True
- Flip Probability	continuous	[0.0, 1.0]	0.4186
ColorJitter	boolean	[True, False]	False
- ColorJitter Brightness	continuous	[0.0, 0.5]	/
RandomAutocontrast	boolean	[True, False]	True
- Autocontrast Probability	continuous	[0.0, 1.0]	0.3930
RandomEqualize	boolean	[True, False]	True
- Equalize Probability	continuous	[0.0, 1.0]	0.1071
RandomPosterize	boolean	[True, False]	True
- Posterize Bits	discrete	[4, 8]	8
- Posterize Probability	continuous	[0.0, 1.0]	0.5359
RandomRotation	boolean	[True, False]	False
RandomAdjustSharpness	boolean	[True, False]	False
RandomSolarize	boolean	[True, False]	True
- Solarize Threshold	discrete	[0, 240]	240
- RandomSolarize Probability	continuous	[0.0, 1.0]	0.0920

Table A.2: Hyperparameters in transformations and related values ranges

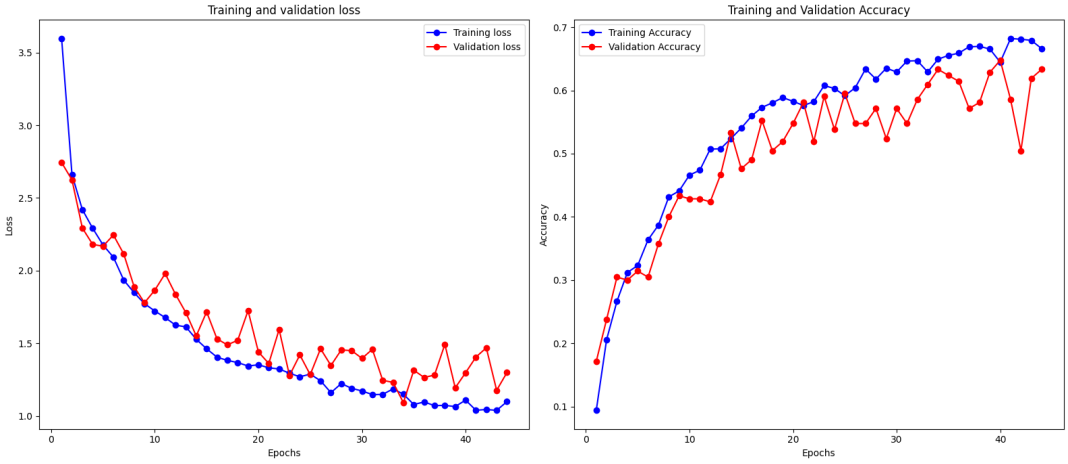

Figure A.2: Custom CNN training history

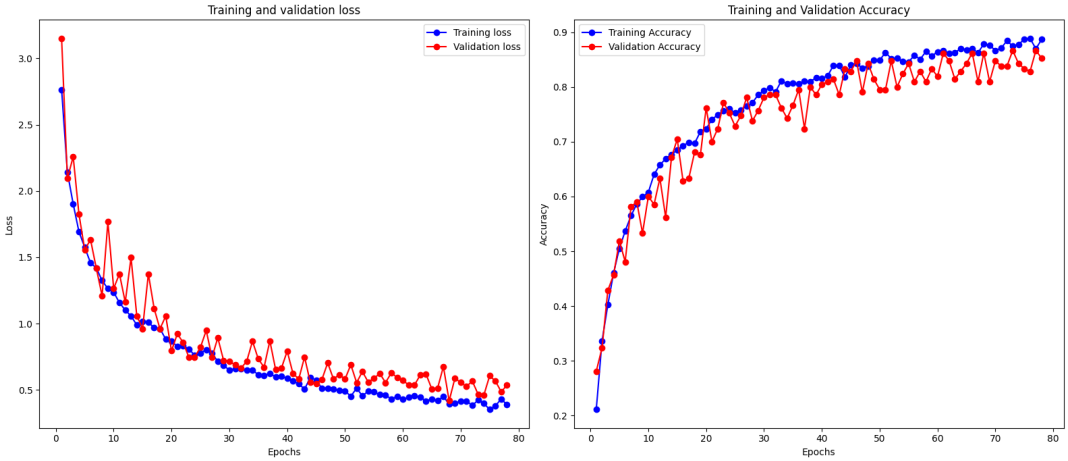

Figure A.3: ResNet-18 training history

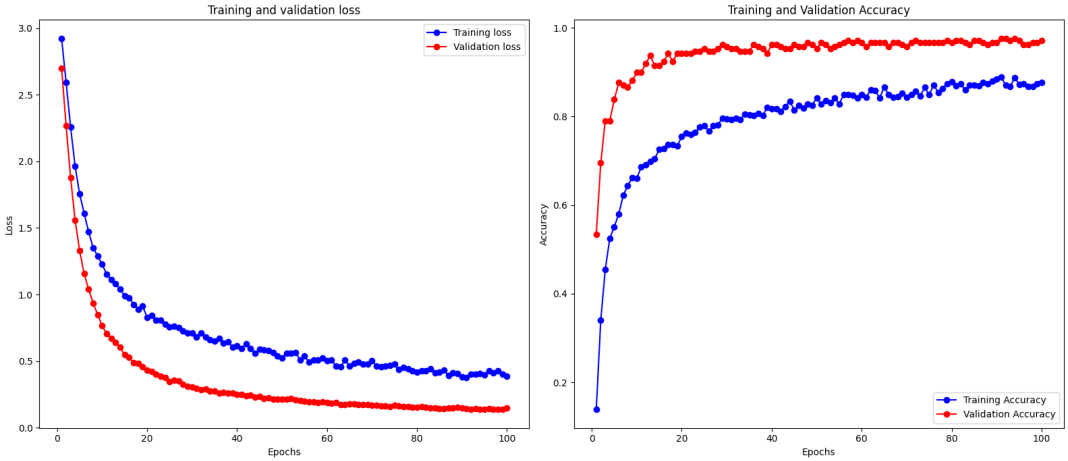

Figure A.4: Fine-tuned MobileNet v3 small training history

A.2. Task 2

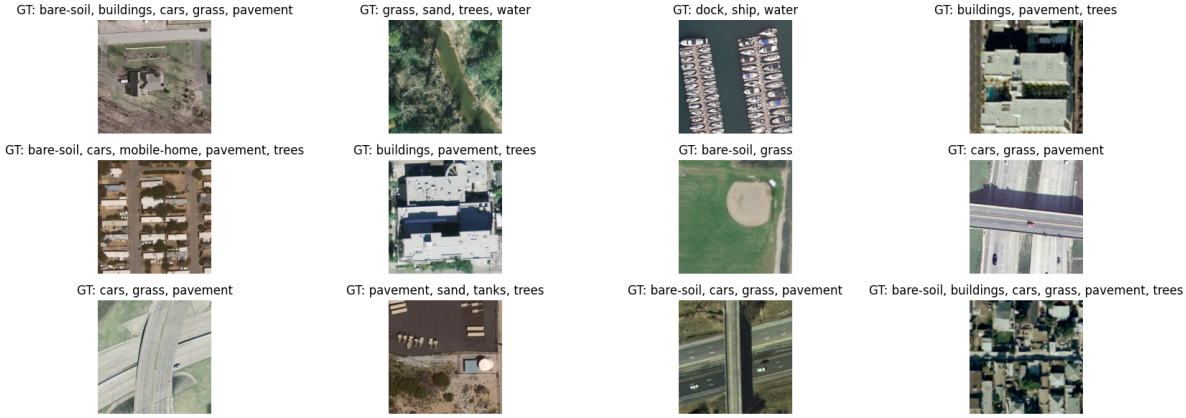

Figure A.5: Land-use classes for multi-labeled case

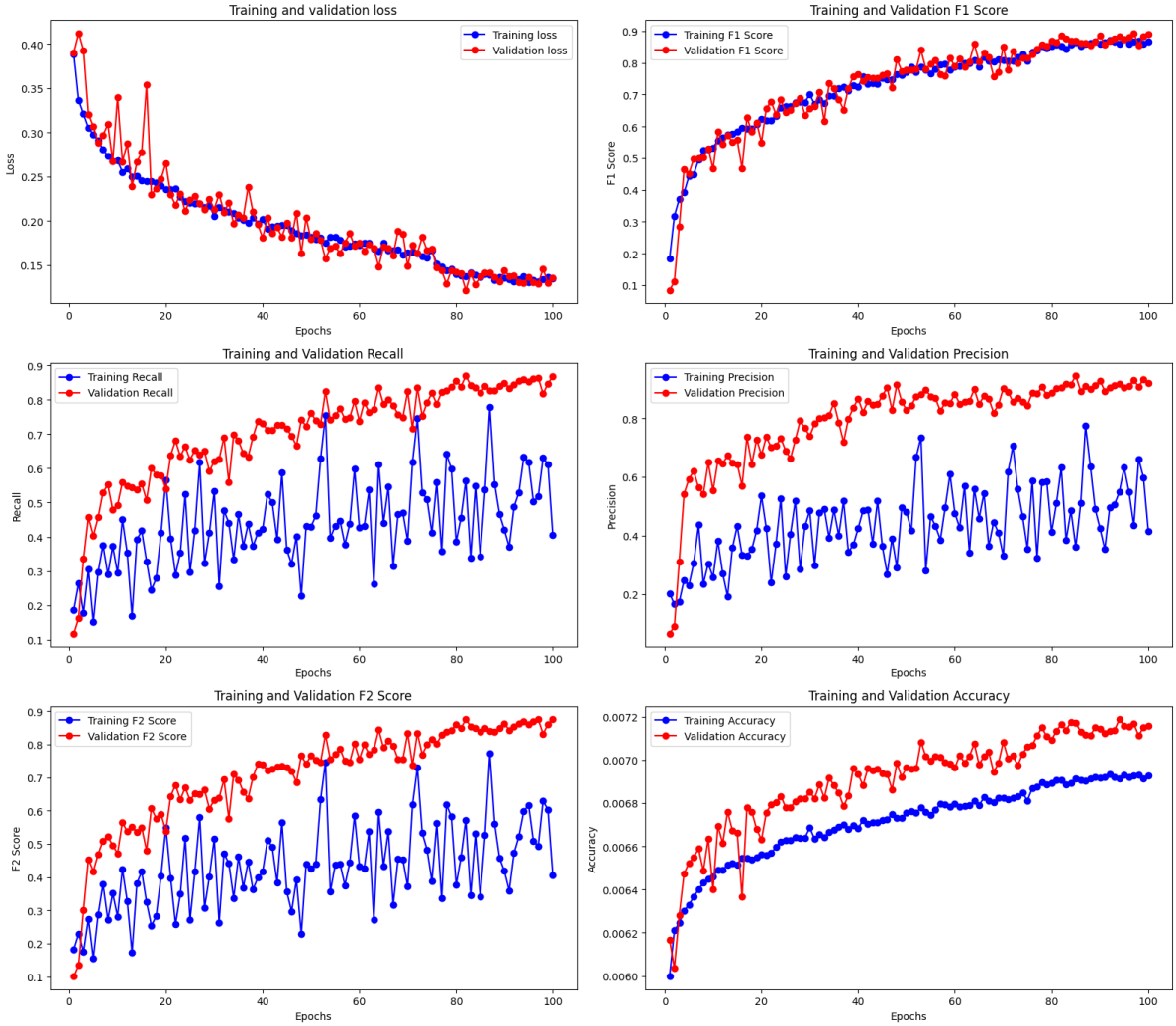

Figure A.6: Baseline MobileNet v3 small training history

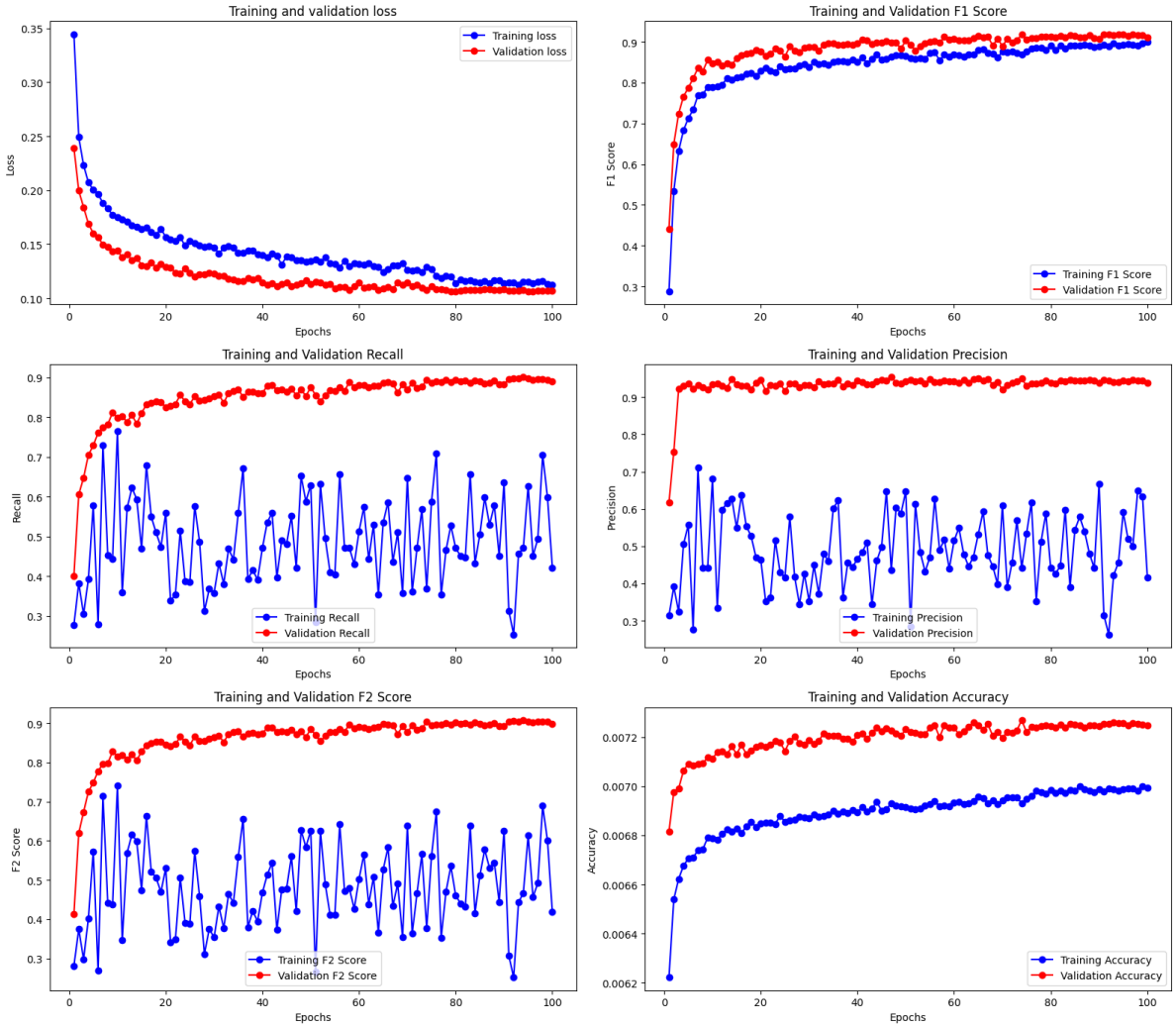

Figure A.7: Fine-tuned Mobilenet v3 small training history

In the best model (MobileNet v3 small after fine-tuning), these are the prediction examples and confusion matrix.

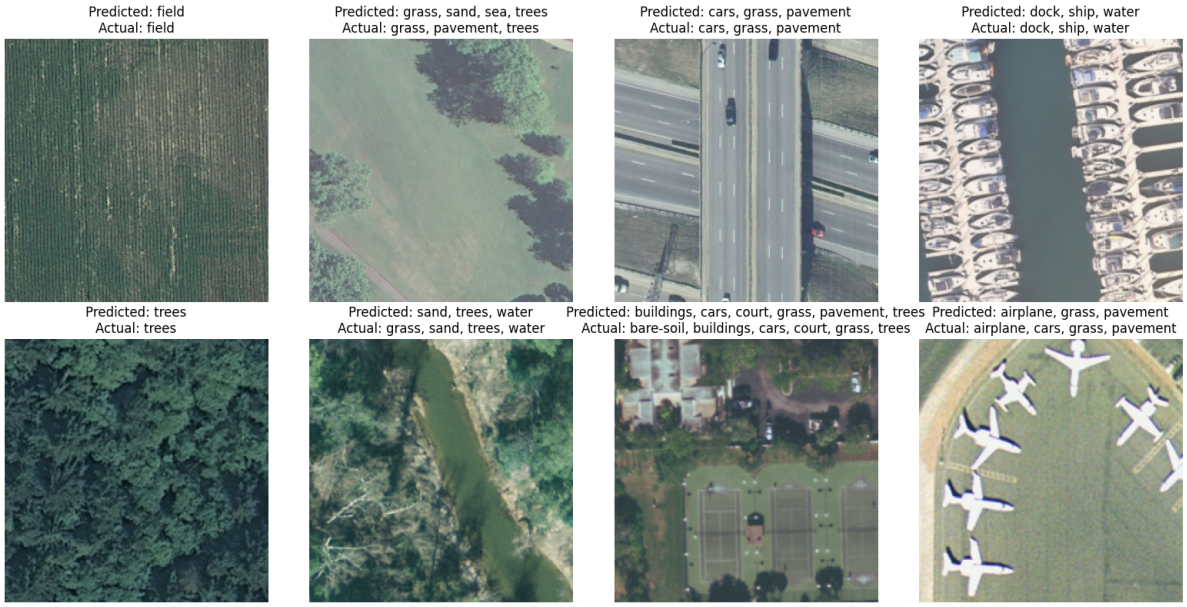

Figure A.8: Prediction by MobileNet v3

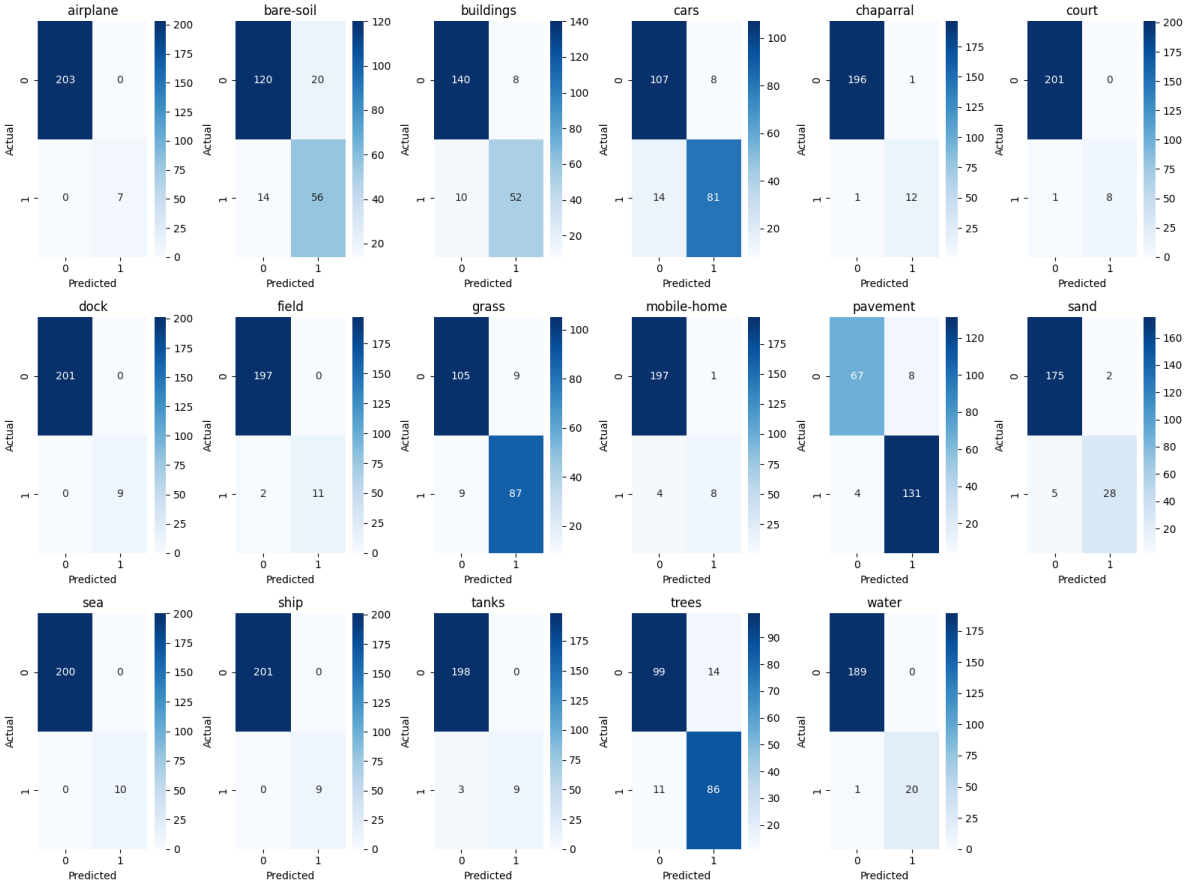

Figure A.9: Confusion Matrix by MobileNet v3