LANDFORM DETECTION FROM SATELLITE IMAGING USING MULTICLASS CLASSIFICATION WITH MLRSNET

Afrin Prio [redacted]

Xiyan Zhao [redacted]

Siaa Gor [redacted]

Lu Gan [redacted]

—-Total Pages: 9

1 INTRODUCTION

Satellite imagery classification is a critical machine learning application for many industries. ML models that can accurately analyze and classify satellite images provide valuable information for urban planning, natural disaster management and prevention, climate change and population monitoring, and more (Wu et al., 2023). This project aims to create an accurate, 22-class single label classification model capable of detecting basic landforms in urban and rural areas, using the MLRSNet dataset that features images from across the world.

The large MLRSNet dataset contains over 100k labeled datapoints, of which we sample 7k for this final report having learned that accuracy is not boosted with more images. The class number of 22 from 40 has also changed from our progress report, given reflection on how overlapping imagery in classes impacted model strength. At our current primary model level, we achieved 96% test accuracy at 0.1 test loss, far better results than progress report and as good as expected. On completely unseen data, the model is also successful, achieving around 850 correct predictions our of 1000 points. In combination with changes to the model and a better dataset, our performance improved significantly from previous results.

The primary model is built on EfficientNet-B2 architecture which is proven to be capable of handling large-scale image data with speed. Particularly, we found training times for EfficientNetB2 faster than our AlexNet baseline even with much more parameters in the former. Our initial approach used custom CNN layers on top of EfficientNet defaults to keep some model layers flexible; however, results have shown these additional layers introduced inaccuracies. Instead, by unfreezing layers of EfficientNet we achieved the final results.

Our baseline model utilizes AlexNet combined with a Support Vector Machine(SVM) architecture. As a classic and fairly accurate CNN model, AlexNet is a reliable baseline for evaluating the performance of the newer, deeper EfficientNet architecture. Both models were tested with and without adaptive median filtering (AMF) to demonstrate whether AMF would positively impact the models, with our findings highlighted in the quantitative and qualitative results.

With this report, we aim to demonstrate improvements in accuracy and efficiency in the primary model, ensure a comprehensive evaluation of the model's performance, and present the first stages of a usable product in industry applications.

2 ILLUSTRATION

The illustration in Fig 1 presents our final model, with EfficientNetB2 base and our fine-tuning details.

3 BACKGROUND AND RELATED WORK

Satellite imagery analysis is a rich and innovative field, and deep learning is a new but impactful tool introduced to this field. Our project leverages many recent studies in this area. Firstly, an approach from "A Deep Learning Approach for Population Estimation from Satellite Imagery" (Robinson(2017) uses the VGG-A CNN architecture to analyze Landsat satellite images, ultimately matching the images to U.S. Census data of population density. Their paper directly inspired our project idea, and the important applications of this project. Next, the paper "MLRSNet: A Multi-label High Spatial Resolution Remote Sensing Dataset for Semantic Scene Understanding" (Qia et al.(2020) was crucial as it provided a comprehensive dataset of satellite images for classification. This paper offered various CNNs to train on this dataset and metrics to evaluate performance beyond accuracy, such as the ability to distinguish between similar items (e.g., highways and runways). It elaborated rigorous methods for creating their multilabel

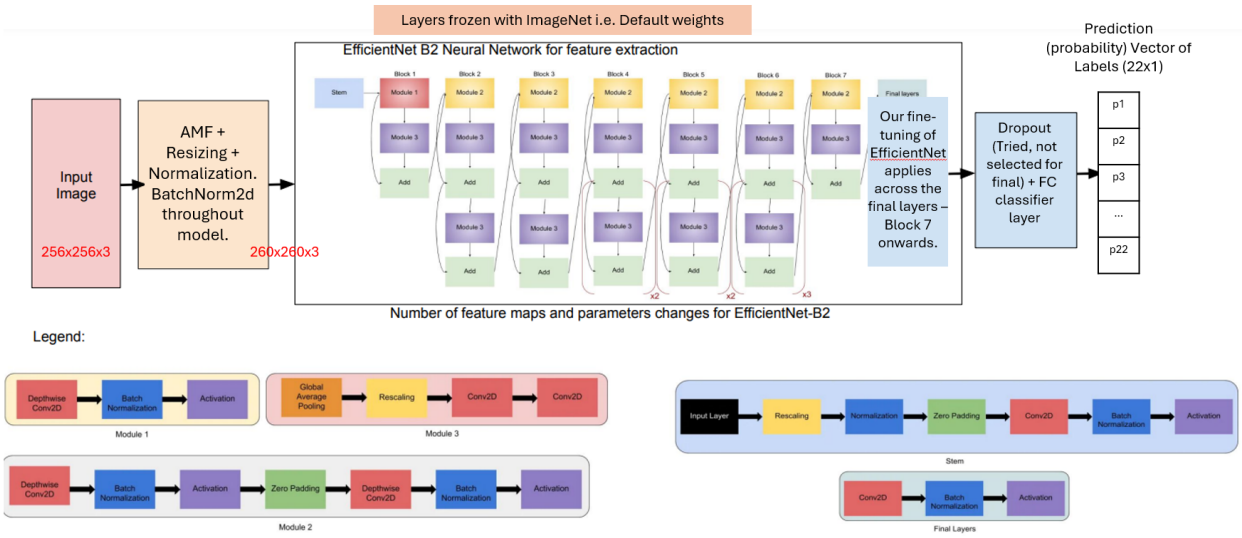


Figure 1: The architecture of our approach is illustrated through this figure. Credits: The EfficientNet-B2 model image (part of the diagram) was according to article: (Agarwal(2020)).

dataset, including global image collection, large-group labeling of dataset, and exclusion of certain images. This is a guiding work for our project, providing a robust foundation for dataset management and evaluation. Then, "Deep Network Architectures as Feature Extractors of Remote Sensing Images" (Stoimchev et al.(2023)) contributed to our understanding of various CNN models and recent advancements, particularly providing the key result in which they compared recent architectures to find EfficientNet performed best on a project similar to ours. Coincidentally, the same dataset we discovered in the previous paper was employed for part of this one.

The fourth paper relevant to our project is "Multilevel Classification of Satellite Images Using Pretrained AlexNet" (Atchaya(2023)), in which we discovered the AlexNet model for a baseline. It demonstrates the adaptability of AlexNet - as an older CNN architecture - in satellite image analysis. Although the research used an older and smaller dataset, it provided a starting point, and we adapted their code to fit our larger and more recent dataset. This paper also introduced the Adaptive Median Filter that we go on to rely on in our data processing, as elaborated in those sections.

The fifth and final paper important to cite is the EfficientNet introductory paper, titled "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" (Tan(2019)). This paper helped in understanding the core model we chose and its architecture decisions. As well, the paper was important in discovering modern standards for CNNs, the levels of resources, time, and dataset size required in training our model, and the compound model scaling that makes EfficientNet 'efficient'.

By integrating insights and methodologies from these foundational works, we have grounded our model development in proven research with room for creativity and tackling specific challenges to our dataset.

4 Data Processing

Having selected MLRSNet as specified, we had 100k+ images to work with across 46 classes and a variety (0.1-10 m/px) of resolutions. As mentioned through our data processing sections in the Proposal and Progress Report, the number of classes was the first detail to change in pre-processing. For the final model, we reduced classes to 22 - this is explained further in the upcoming notes on this section.

At 22 classes, the model trained on the 20k images targeted from our progress report and a 7k images dataset. It was evident from losses, accuracies, and qualitative evaluation of which classes saw most correct predictions that the 20k dataset offered little benefits over the faster and more refine-able 7k dataset. The latter was then used for both primary and baseline, and offered training times of under 2 hours for most sets of hyper-parameters.

In terms of image processing, the first step was to resize images to 260x260x3 as the input size for EfficientNetB2. RGB normalization was then tried, to assist in removing shadows or highlights and receive clearer images for the model. Ultimately RGB normalization did not return significantly better results than raw images (still 95%+ accuracies), possibly because of the low resolution of images and grainy boundaries between objects in satellite imagery. Either way, this step was not included for the final model, which proved suitable for the next preprocessing step of AMF.

As mentioned in the progress report, AMF requires '2D' colored i.e. grey-scale image inputs as it cannot work with 3D of R, G, and B channels of coloration. Therefore, in this preprocessing step the dataset was converted to gray-scale for the median filter, where the filter essentially reduced distortion typical of satellite images. Particularly with the variance in resolution of our dataset, this proved beneficial.

Returning to the discussion on pruning the dataset, the first observation towards pruning below 10k was that data

presents itself in sets. Images # 00049-00066 of class 'river' for example are all of the same river, taken with the same satellite, at different angles but fundamentally the same object. This meant taking the first x images of the 22 classes to reach 10k didn't offer significant benefit, likely why we were seeing lower accuracies with that setup. Taking 7k randomized images instead offered variety and therefore a better dataset, at which point classes were reduced.
Some classes, such as 'meadow', were immediately eliminated after the Progress Report, knowing these labels overlapped with others causing unintentional imbalance. After pruning by degree of overlap and number of images available (i.e. classes with only 1600 images available were pruned), classes were arranged in groups by potential similarity in images. If two classes were incredibly similar, such as 'eroded_farmland' and 'farmland', one was kept to represent both. The final chosen classes remained clustered in similarity groups as shown in Fig [the excel classes fig], but none were redundant or irrelevant to remote sensing missions, and the selection was still challenging for the model.
Finally, as with every iteration of the data processing, manual removal and replacement of images was performed on the final dataset, with images containing no discernible features or incorrect labeling (e.g. 'sparse_residential_area' mapped to 'parking_lot') taken out one final time.

5 Architecture

The first observation for Primary Model construction in this report in that our own CNN layers were introducing incorrectness in prediction rather than assisting in it. We knew EfficientNet has several million parameters, and consequently very precise feature extraction, but the additional CNN layers were meant to leave some layers entirely flexible so as not to rely fully on default weights. By systematically unfreezing EfficientNet's final layers, we found that unfreezing up to and including a small part of Block 6 gave us high enough accuracies without any need for CNNs. Therefore, trainable layers were selected as per Fig [1], where Block 7 is left open, and within Block 6 the Conv2D layers in the Block's final iteration specifically are selected to remain open.
In addition, we discovered that the torchvision implementation of EfficientNet Team (2024) allowed us to specify a batch normalization layer to be used through the model where normalization is present, and we specified the recommended BatchNorm2D layer to be used.
A further alteration to the model for this report includes introducing dropout - experimenting with different values, a dropout of 0.1 offered a boost to accuracy, changed to 0.15 after hyperparameter tuning. However, dropout did not have an incredibly significant effect, and instead pointed towards overfitting of the model with 99%+ training but lower test accuracies. In the final, dropout was not kept as a selection.
Once these selections were made, the above model as described was trained on raw images and images with AMF. Our primary model is ultimately the one using AMF, but building the two models offers application to different areas of remote sensing, as well as a point of comparison for whether AMF is effective.

6 Baseline Model

The baseline model is used so that the performance of the primary model can be evaluated based on the baseline accuracy and F1-scores. For an imbalanced dataset, the F1 score is a better comparison if more data is added but for a balanced dataset, i.e. in our case, accuracy is likely sufficient. Various combinations of the baseline models were explored including using AlexNet without the Adaptive Median Filter (AMF), AlexNet with AMF, and SVM models that incorporate the AlexNet pre-trained features as shown in Figure 8. The SVM model was chosen as the baseline due to the higher accuracy; a greater than 85 percent result is expected from our Proposal research, where we discovered a study also dealing in satellite imagery where this model resulted in an 87.5% accuracy for 8 classes (**?**). A diagram that displays the baseline model performance for this project is depicted in Figure 9 with the true and predicted values, but this set of predictions is to be discussed further in the results section. The features from

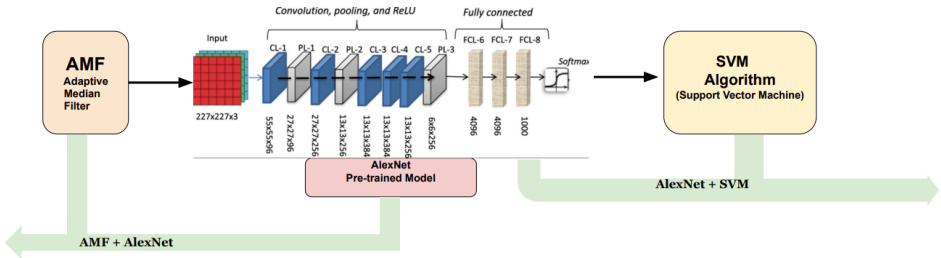
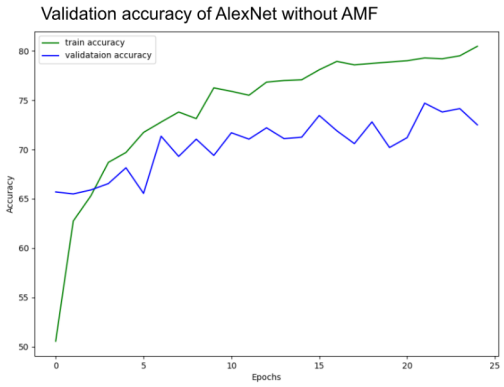


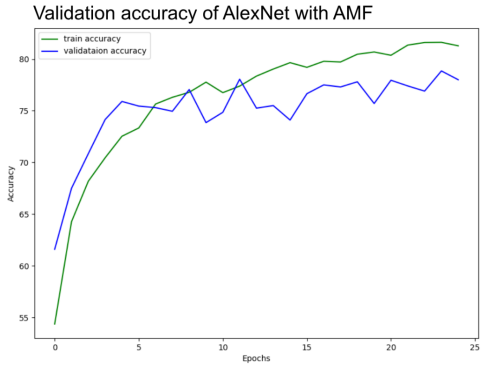
Figure 2: The diagram of the proposed baseline model. Credits: ResearchGate for the AlexNet model

the AlexNet pre-trained model were extracted and saved in a specified folder; these tensors were then loaded using DatasetFolder, to be converted to a data loader during training. A fully connected classifier on top of Alexnet was also

created and trained. With and without the AMF, the training time for this process was about 2 hours on a 10k image dataset. The validation accuracy without the AMF was about 73% and with AMF about 78% as shown in Figure 3a and Figure 3b respectively.



(a) This figure depicts the train and validation accuracy of AlexNet without AMF applied.



(b) The figure depicts the train and validation accuracy of AlexNet with AMF applied.

Figure 3: Figures for train and validation accuracy with and without AMF applied.

In order to separate the different classes to make appropriate predictions and minimize error, an optimal maximum marginal hyperplane was created in a multidimensional space by using a Support Vector Machine (SVM). A challenge that was faced in the AMF and AlexNet combination for the baseline model was the non-zero and highly fluctuating loss. The loss result in a significantly high validation loss and poor performance. Thus, an SVM algorithm was used to select a maximum margin hyperplane by selecting the maximum segregation from closest data points (**?**). Before drawing a hyperplane to separate the classes, kernel transformations are used in combination with SVMs; the kernel transforms the input data into a higher-dimensional version to classify non-linear data accurately.
Various forms of kernels are used to test performance such as linear, radial basis (to map to an infinite dimensional space), and polynomial; for this project's baseline model, a linear kernel was sufficient (**?**). The SVM model from Scikit-learn is easy to implement and requires minimal tuning as a machine learning algorithm; among the few parameters that one may decide to alter are the kernel and the cost that is inputted. The total time to classify, and train or fit the model took about less than an hour. Additionally, the AlexNet features were converted to numpy arrays, flattened, and transformed which was a challenge during the process. Before passing these arrays of extracted numpy inputs and labels to the SVM, saved features from AlexNet were visualized from the train loader. The results of the AlexNet pretrained features with the SVM led to a validation accuracy of 90.5%.

7 Quantitative Results

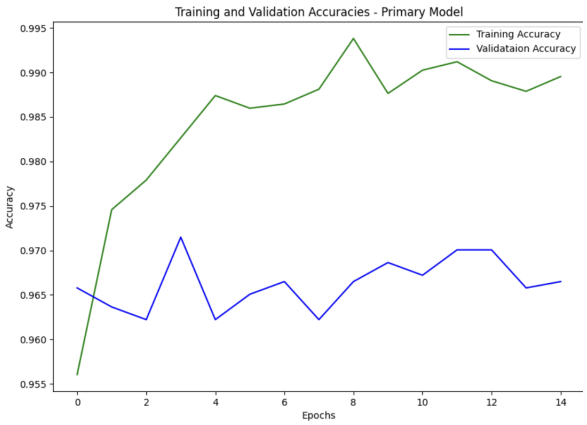
The final model for AlexNet was obtained after several hyperparameter tuning by changing the architecture by using dropout, increasing epochs, changing the learning rate, and batch size alteration. Training results are also shown in the baseline model figures. Three datasets were used: Small (4000 images), medium (7000), and large (about 63,000 images) for training.

Table 1: Final AlexNet model's validation and training accuracy compared to the SVM model.

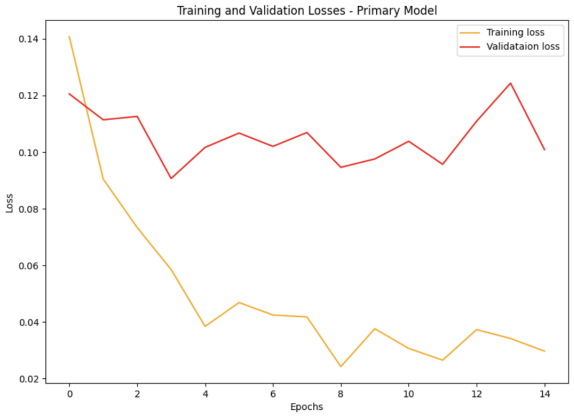
	AlexNet (without AMF)	**SVM**
Best Validation Accuracy	75.20703933747412	89.2059553349876
Test Accuracy	71.84265010351967	90.46199701937406

Moving on to the Primary Model, it is noted that loss and accuracy are sufficient to describe performance of the model quantitatively, as this is a classification task and the dataset splits of training, validation, and test contain enough images (roughly 4200, 1400, 1400 respectively) for the accuracy percentages to be significant. These metrics are presented in table 2. In addition, the training and validation curves are also presented (Fig 4,5) as quantitative results, demonstrating these metrics over the optimum selected epoch number of 15 at learning rate of 0.001. Loader size/batch size was optimized at 25 for the final model, with even slightly higher sizes like 50 surprisingly dropping accuracy significantly by the 15th epoch.
The most significant detail from the accuracies and losses is that validation metrics reach a high point early in the Epochs, oscillating around that point with slight improvement until the end. It is noted that over very short epoch amounts (5-10), validation reaches saturation quickly but training displays room for improvement. This factored into our selection of 15 epochs as the optimum.

The primary model performs comparably with and without AMF, with the test dataset results being slightly worse for the model with AMF. For applications that can accept gray-scale images, the model with AMF would prove faster due to the reduced image size and equally as accurate, with color-dependent applications benefiting from our primary model.

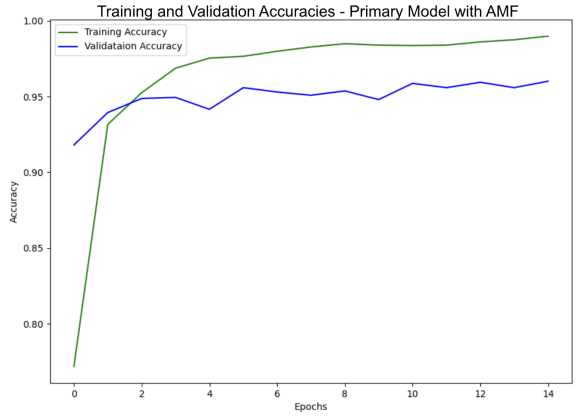


(a) Training and validation accuracy of our Primary Model without AMF applied.

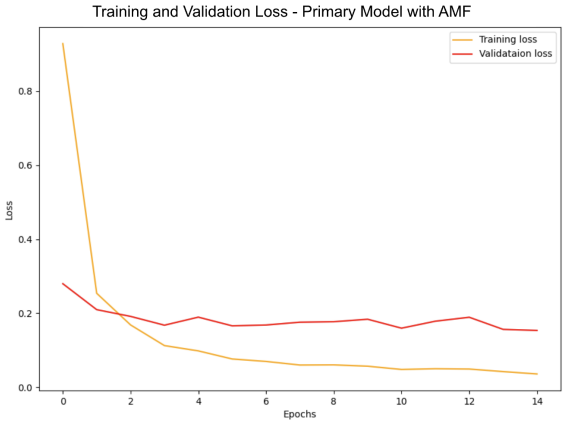


(b) Training and validation loss of our Primary Model without AMF applied.

Figure 4: Figures for training and validation accuracy of our Primary Model with and without AMF applied.



(a) Training and validation accuracy of our Primary Model with AMF applied.



(b) Training and validation loss of our Primary Model with AMF applied.

Figure 5: Figures for training and validation accuracy of our Primary Model with and with AMF applied.

Table 2: Comparison of Training, Validation, and Test Performance Metrics for Primary Model and Primary Model with AMF.

Model	**Training Accuracy**	**Validation Accuracy**	**Test Accuracy**	**Training Loss**	**Validation Loss**	**Test Loss**
Primary Model	0.990	0.967	0.969	0.030	0.101	0.113
Primary Model with AMF	0.990	0.960	0.958	0.036	0.153	0.137

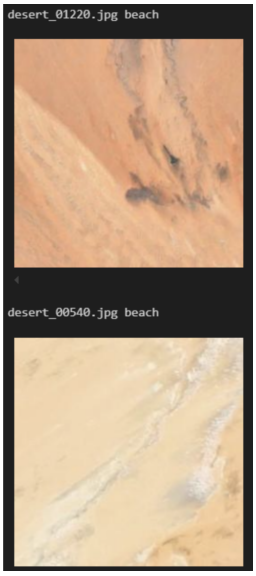
8 Model Evaluation on New Data

The final quantitative area of evaluation comes in the form of evaluating on new data, which was accomplished twofold. Firstly, as we had a very large dataset available, a set of 1000 images were randomly selected from the leftover images, and passed through the model. A script was written to count all correct predictions on those images, and print the true label vs. predicted label if our model got the answer wrong. Running this script 5 times with 1000

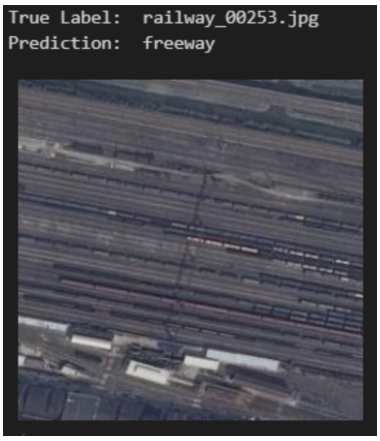
random, unseen images each time produced results of 752, 803, 903, 860 and 891 correct predictions. Interestingly, moving the allowable range of the random selection to the end of the MLRSNet dataset always resulted in poorer performance, likely because the end of each class holds images lowest in quality and not belonging to any sets of the kind mentioned in Section 4. Regardless, this implied that our dataset performed slightly poorer on new data than on the sample used to train our model, which prompted looking into the images that it guessed wrong.

Of the wrongly guessed images, it was obvious that the class groupings from Table 3 were most often confused. 'Desert' and 'beach' images from Fig 6a for example saw confusion, as well as 'railway' and 'freeway' classes (Fig 6b) The most alarming mistakes were from classes that had similar line structure in the image but completely different coloration and content, like Fig 7a's 'river' detected 'beach' or 'desert' as 'snowberg' in Fig 7b. Essentially, this issue ties back to the complexity of this project and the flaw with classification based only on images; in fact, while researching earth science classification projects we did discover that spectral information and various other factors such as temperature and topography are typically included to reliably perform this task in industry.

As a due diligence step, after introducing new data from our dataset we also took a small batch of 30 images from a completely different dataset (RESISC45), observing what labels our model provides for those images. The most unreasonable prediction was several detections of forest-like hills as 'snowbergs', with $\frac{5}{8}$ mistakes owing to that error. However, this meant the other 22/30 images were predicted reasonably to what a human might call the given image, with airports as an example being labeled parking lots, or beaches as lakes.



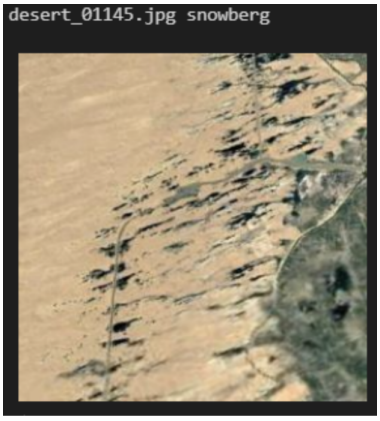
(a) 'Desert' and 'beach' classes confusion



(b) 'railway' and 'freeway' classes confusion

Figure 6

(a) 'river' and 'beach' classes confusion



(b) 'desert' and 'snowberg' classes confusion

Figure 7

9 Qualitative Results

The confusion matrix is used to visualize how the baseline model is performing using colours. The AlexNet features combined with the SVM model are chosen as the baseline. The bright yellow colour indicates that all of the baseline model's predictions were correctly predicted whereas the colour blue depicting otherwise. This test set involved more than 100 images per class. The goal is to achieve the yellow colour in the diagonal of the confusion matrix; for instance, 'bridge' was the true label and 102 of them were correctly predicted as bridge images.

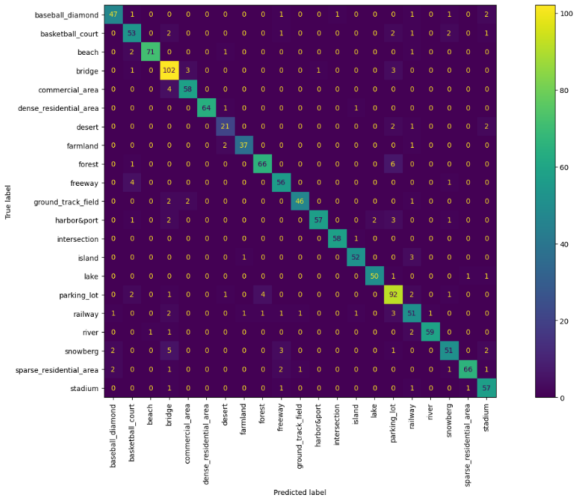


Figure 8: The confusion matrix for the test set is composed of unseen images for some of the classes

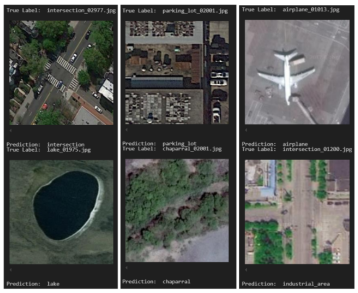


Figure 9: The true labels are located at the top of the image, and at the bottom of the image are the prediction of the AlexNet baseline model for various classes.

Some more outputs of the AlexNet model are shown with their true and predicted labels in Figure 11. The SVM model has only few yellow or correct predictions and most of them are light green for the test set. This shows that even though the accuracy of the test set was about 90.5%, there are still few classes that are not classified correctly. The F1 score, precision, and recall would be other measures to evaluate the model further. Since this is the baseline, the results were acceptable as a benchmark where the final model will perform better than the baseline.

Turning again to the Primary model, we used a very similar script to Section 8 but instead of new images we sent randomized images from the original testing and validation subsets to detect which classes were most frequently confused with each other. This produced the color-coded results in Table 3. The rates of confusion are discussed in quantitative analysis, but qualitatively, the major issue in frequently confused images remains to be object structure. 'Basketball_court' for example is predicted for red, rectangular shapes, such that even sparse residential areas are sometimes labeled to be courts due to red houses in the image. This points to another immediate issue - that of actually overlapping features within the picture. If an actual basketball court is present in a residential area image, for example, our model would not be incorrect in predicting either the court or the house. There was no way to account for these false negatives in our training process, but ideally a multilabel classification project could handle this issue. The final challenge in qualitative analysis is demonstrated by figure 6a, where detail-less or unicolor images struggle to pass through model training with decisive impact on parameters. To deal with issues of this sort in real life, a color filter can be applied to take out things like full-blue water body images before model training; this sort of filtration for our dataset would unfortunately lead to imbalance, and was therefore not applied.

baseball_diamond	basketball_court	beach	commercial_area	dense_residential_area	desert	farmland	forest	freeway	ground_track_field	harbor & port
intersection	island	lake	parking_lot	railway	river	snowberg	sparse_residential_area	stadium	swimming_pool	tennis_court

Table 3: Color-coded table of categories.

10 Discussion

On the 7k dataset, the model is performing far better than was predicted by the progress report. EfficientNet has proved a decidedly powerful model, and fast to train as well, taking around an hour for even our most extensive set of hyperparameters at Epoch = 25. The most surprising thing was which classes sometimes get confused and due to what reasons - 'river's and 'freeway's for example do not sound like common landforms but do get mispredicted occasionally. The biggest improvement to training came from the fine-tuning layers allowed to unfreeze, and from erasing any additional CNN blocks on top of EfficientNet.
As well, it is noted that the performance on new data is averaging at around 85% in five trials, which is not as high as would be hoped. Part of this is again attributed to missing datapoints such as spectral information or temperature data that would realistically be available in industry, but the part attributed to our model's inefficiency with new data is still significant. With that said, suggestions to improve the model include the ideas mentioned in qualitative results - pre-detecting unicolour images, using highest possible resolutions of image sensors, and limiting classes of detection further could all assist in better performance for launching this product in the world. From a pure ML standpoint without context of industry application, the model does meet our expectations overall.

11 Ethical Considerations

The application of satellite imagery carries significant ethical implications, including privacy, data protection, and the potential for technology misuse in military or law enforcement applications. While the resolution of satellite imagery typically prevents the identification of individuals, the collection and analysis of such data can inadvertently expose sensitive information about demographic movements and densities. This is particularly critical in areas inhabited by vulnerable populations who might not have consented to data collection. Furthermore, we must ensure adherence to international regulations and respect national boundaries, especially when processing images that include identifiable properties or culturally sensitive landmarks. Data security measures must be implemented in these projects to prevent unauthorized data breaches that could lead to privacy violations (Politzer(2021). As projects like ours shift to increased accuracy, the risk of misusing these data for surveillance or control increases, underscoring the need for stringent data controls.

Another key ethical issue is potential bias in the dataset. The transition to single-label classification from multilabel required us to put even greater importance towards preventing bias, both from a technical and ethical standpoint. For example, a model predominantly trained on data from specific regions could misrepresent or fail to recognize similar structures in other parts of the world, leading to inaccurate classifications and potential misapplications in real-world scenarios.
Finally, although our project is designed for earth-science and observation applications, law enforcement and military use of similar technology necessitates a cautious approach. We must specifically remain guarded against harmful possible uses of our technology, standing against uses of our work to facilitate the creation of detrimental systems.
By addressing these ethical considerations, we aim to hold ourselves responsible for our work and continue development of satellite image analysis technology without advancing ethically doubtful objectives, recognizing our duty to safeguard the rights and privacy of global communities.

12 Project Difficulty / Quality

The EfficientNet-B2 model would be suitable to solve the challenge of classification of satellite images with noise. During the process of satellite image generation via sensors, salt and pepper noise interferes with the classification of the deep neural network. Even small amounts of noise may lead to inaccurate classification; however, with the Adaptive Median Filter and using the EfficientNet's varied number of feature maps, the noise would be able to reduce and the parameters may increase. One advantage and disadvantage we found when using the AMF was that it only operated on grayscale images. This means that the regardless of colour found in the satellite image, the classification of the location or object remains the same; it may be dangerous if the colour is used to distinguish between the change occurring at a particular location is not detected as it is important for agriculture or crop health. As a result, this project is challenging as it not only involves removing the noise from the satellite but ensuring that the results are reliable;

therefore, after hyperparameter testing and various methods to reach a high accuracy, the final model EfficientNet B2 achieves a reasonable test accuracy.

13 Link to Github or Colab Notebook

Link to Colab Notebooks:
https://colab.research.google.com/drive/1nK_VRnOWQEdGxAo2EJhOD9yN1dU01-6o?usp=sharing
https://drive.google.com/file/d/1rKuBSim2yEpxaPkn2nUHlyJqcG4rvDST/view?usp=sharing
https://drive.google.com/file/d/11N04qzD10z4Zaubnh7sOMTYPZY8-LbFF/view?usp=sharing
https://drive.google.com/file/d/12oU3AI3J-vRlka4knY0L5US1xWTQzmGr/view?usp=sharing

References

(Agarwal. Complete Architectural Details of all EfficientNet Models — towardsdatascience.com. https://towardsdatascience.com/complete-architectural-details-of-all-efficientnet-models-5fd5b7 2020. [Accessed 06-06-2024].

(Atchaya. Multilevel Classification of Satellite Images Using Pretrained AlexNet Architecture — link.springer.com. https://link.springer.com/chapter/10.1007/978-3-031-34222-6_17, 2023. [Accessed 06-06-2024].

(Politzer. Why we need to think about ethics when using satellite data for development. https://www.devex.com/news/why-we-need-to-think-about-ethics-when-using-satellite-data-for-development-9 2021. Accessed: 06-06-2024.

Panpan Zhu (Qia et al. Mlrsnet: A multi-label high spatial resolution remote sensing dataset for semantic scene understanding. *arXiv*, abs/2010.00243, 2020. URL https://arxiv.org/pdf/2010.00243. Accessed: 06-06-2024.

Bistra Dilkina (Robinson, Fred Hohman. A deep learning approach for population estimationfrom satellite imagery. *ACM Digital Library*, Volume Number, 2017. doi: 10.1145/3149858.3149863. URL https://dl.acm.org/doi/epdf/10.1145/3149858.3149863. Accessed: 06-06-2024.

Marjan (Stoimchev, Dragi Kocev, and Sašo Džeroski. Deep network architectures as feature extractors for multi-label classification of remote sensing images. *Remote Sensing*, 15(2), 2023. ISSN 2072-4292. doi: 10.3390/rs15020538. URL https://www.mdpi.com/2072-4292/15/2/538.

(Tan. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv*, abs/1905.11946, 2019. URL https://arxiv.org/abs/1905.11946. Accessed: 06-06-2024.

PyTorch Team. Efficientnet model implementation in pytorch. https://github.com/pytorch/vision/blob/main/torchvision/models/efficientnet.py, 2024. Accessed: 08-15-2024.