Deep Learning Final Report

Joseph Tartivel, Ádám Földvári, Máté Lukács

April 2025

Table of contents

1. Introduction

The xView recognition benchmark presents a challenging image classification task using satellite imagery. The purpose of this assignment is to create and evaluate various deep learning methods for object classification in overhead images. From simple feedforward neural networks (ffNNs) to regularization techniques, convolutional neural networks (CNNs), and finally utilizing transfer learning with pre-trained architectures, we will gradually apply increasingly complex models. The objective is to address the difficulties that come with classifying satellite imagery, such as small object recognition, while also comprehending the advantages and disadvantages of each method.

2. Data set

The xView dataset is one of the largest publicly available collections of high-resolution overhead imagery with more than 1 million object instances in 60 classes. The images, which covered roughly 1,400 km^2 of imagery, were taken from WorldView-3 satellites at a ground sample distance of 0.3 m [1] [2]. For this assignment, a subset called "xview_recognition" was used which was created by cropping the original images using their annotated bounding boxes to extract objects of interest.

The dataset includes 2,635 testing - and 21,377 training images, all of which have been resized to 224x224 pixels. Helicopters are significantly under-represented in the dataset (0.32%). This imbalance significantly impacts model training and performance evaluation, requiring special consideration during the development process.

The dataset includes 12 classes: Cargo plane, Helicopter, Small car, Bus, Truck, Motorboat, Fishing vessel, Dump truck, Excavator, Building, Storage tank, and Shipping container. Sample items from various benchmark categories are displayed in Figure 1.

Figure 1: Sample objects of different categories acquired from the "xview_recognition" benchmark.[3]

3. Problems

There are a number of difficulties when using the xView recognition benchmark:

- **Small Object Recognition:** Accurate classification of many objects in satellite imagery is challenging due to their small size and lack of distinguishing characteristics. This problem is made more difficult by the lower resolution.

- **Data Quality Issues:** Model training may be adversely affected by the xView dataset's noisy bounding boxes and mislabeled data. According to earlier studies, "…the amount of mislabeled data and noisy bounding boxes were significant enough to affect overall model training.".
- **Intra-Category Confusion:** Prior methods have demonstrated a high degree of confusion among related categories (e.g., different types of vehicles). Models may have no trouble differentiating between broad categories, such as cars and buildings, but they may have trouble classifying specific items within these categories [4].
- **Limited Training Data:** It can be difficult to learn strong representations for under-represented classes because, despite xView's overall size, some classes have very few examples.

4. Feedforward Neural Network (ffNN) experiment

In this section, the exploration of the application of feedforward neural networks (ffNNs) is discussed to the xView recognition benchmark. Two different ffNN architectures were implemented and evaluated to establish a baseline for more complex models.

Model Architecture Comparison

Model 1 was composed of a deeper architecture with multiple hidden layers. An input layer (flattened 224x224x3 images). Three hidden layers (first: 512 neurons with ReLU activation and 50% dropout; second: 256 neurons with ReLU activation and 40% dropout; third: 128 neurons with ReLU activation and 30% dropout). The output layer consisted of 12 neurons with softmax activation (one per class). [5]

Model 2 was a simpler architecture with fewer parameters. Input layer (flattened 224x224x3 images). One hidden layer with ReLU activation and 20% dropout. Output layer with 12 neurons with softmax activation.

The Adam optimizer with a categorical cross-entropy loss function and a learning rate of 0.001 was used to compile both models. A batch size of 16 was used for training over 30 epochs, with callbacks for model checkpointing, learning rate reduction, and early stopping.

Performance Analysis

The performance metrics reveal significant differences between the two architectures:

Model	Test Accuracy	Test Recall	Test Precision
Model 1	20.57%	93.38%	8.33%
Model 2 (Model ffNN - ffnn,ipynb)	45.2%	33.66%	30.31%

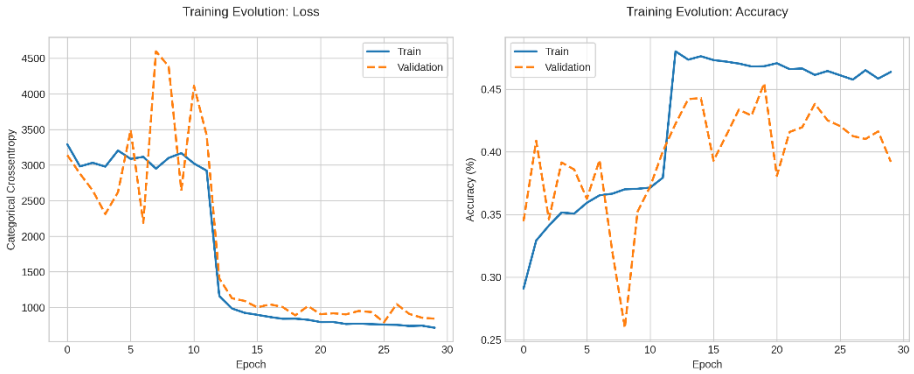

Figure 2: Training loss and accuracy curve of Model ffNN

Training Dynamics: For both models, the training plots display unique patterns. Although Model 1's loss decreased quickly at first, it was unable to increase accuracy past about 21%, and its validation accuracy peaked at about 23%. With a higher training accuracy (~47%) and validation accuracy (~45%), Model 2 demonstrated a more gradual improvement.

Model 1 Analysis: Model 1 performed poorly even though it had a deeper architecture and more parameters. The model essentially memorizes training examples without learning generalizable features, according to the training curves, which indicate severe overfitting. The incredibly high recall (93.38%) and low precision (8.33%) suggest that the model is making indiscriminate predictions about some classes, leading to many false positives.

Model 2 Analysis: In every metric, the simpler architecture outperformed the more complex one. Model 2 learnt more generalizable features from the satellite imagery by simplifying the model and utilizing a single layer with the proper regularization.

Key Insights

Several significant feedforward neural network insights were uncovered by the xView recognition benchmark. Our simpler Model 2 outperformed the deeper Model 1, proving that **complexity does not always translate into better performance**. This implies that adding complexity could backfire for flattened image data with few training examples. High dropout rates (50, 40, and 30%) were used in Model 1, but **overfitting**—the model becoming overly specialized to training data and failing to generalize—remained a major problem.

Because the 224×224×3 images were flattened into 1D vectors, important spatial **relationships between pixels were lost**, resulting in significant information loss for both architectures. Performance was greatly impacted by the dataset's **extreme class imbalance**, especially for Model 1, which displayed bias towards majority classes with high recall but low precision.

Lastly, Model 2's better performance with fewer parameters showed that, for this specific task, model depth was less significant than **parameter efficiency** and **suitable feature extraction**. These results indicate the need for more advanced methods and highlight the shortcomings of ffNNs for challenging image classification tasks.

Limitations of ffNN Approach

The use of flattened input, which eliminates spatial information essential for image classification, is the primary drawback of both models. Furthermore, neither model includes a mechanism to deal with the issue of class imbalance. Both models' comparatively poor performance (even the superior Model 2 only had an accuracy of 45.2%) suggests that ffNNs are not a good fit for this challenging satellite imagery classification task.

These findings set a baseline and emphasize the need for more advanced methods (discussed later), like convolutional neural networks (CNNs) and specialized regularization techniques, that can better preserve spatial information and treat class imbalance.

5. Regularization Techniques for ffNNs

This section investigates the use of different regularization techniques to improve model performance and avoid overfitting, building on the knowledge gathered from our initial ffNN experiments. Even though our baseline ffNN models performed poorly on the xView recognition benchmark, they might be able to generalize better with the help of suitable regularization techniques.

Implementation of Regularization Techniques

A few significant improvements to the prior ffNN architecture were included in our most effective regularization technique. After every dense layer, we added batch normalization layers, which successfully normalized activations, stabilized the learning process, and made it possible to employ higher learning rates. [6]

The batch normalization layers greatly increased their efficacy while preserving the dropout rates from our previous deep model, which were 50%, 40%, and 30%, respectively. To provide more stable gradient estimates that complemented the batch normalization method, we doubled the batch size from 16 to 32.

The regularized model had enough time to converge correctly because the training duration was increased from 20 to 60 epochs, as shown by the training curves, which showed consistent improvement up to about epoch 50. We changed the epsilon parameter of the Adam optimizer from 1e-8 to 1e-7 for extra numerical stability.

The model architecture maintained the same general structure as our previous deep ffNN:

- **Input layer:** Flattened 224×224×3 images
- **First hidden layer:** 512 neurons with ReLU activation, batch normalization, and 50% dropout
- **Second hidden layer:** 256 neurons with ReLU activation, batch normalization, and 40% dropout
- **Third hidden layer:** 128 neurons with ReLU activation, batch normalization, and 30% dropout
- **Output layer:** 12 neurons with softmax activation

Performance Analysis

The regularized model demonstrated significant improvements over both previous ffNN implementations.

Model	Test Accuracy	Test Recall	Test Precision
Model Reg (reg.ipynb)	55.18%	55.5%	44.95%

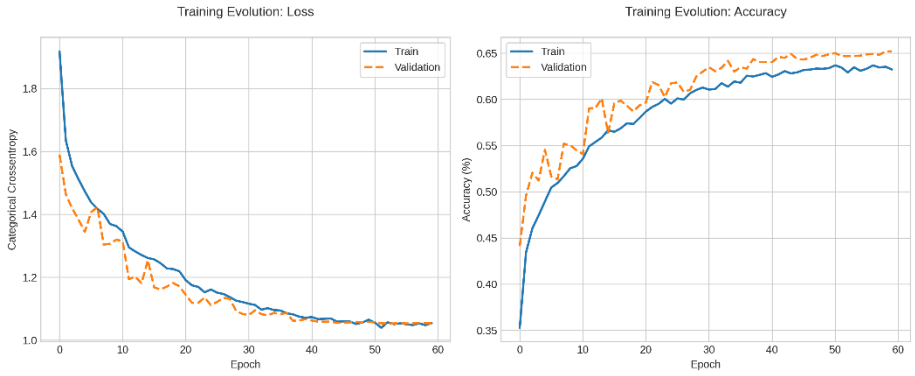

Figure 3: Training loss and accuracy curve of Model Reg

In contrast to our earlier models, the training and validation curves (illustrated Figure 3) demonstrate a far healthier learning pattern. As accuracy rises during training, the loss gradually falls, and validation metrics closely mirror training metrics. This shows that there isn't any significant overfitting, and the model is learning efficiently. Compared to our earlier implementations, the final model converges at a validation accuracy of about 65%.

Impact of Individual Regularization Techniques

Batch Normalization: For our deep architecture, this method worked especially well. Batch normalization helped solve the internal covariate shift issue by normalizing the inputs to each layer, which improved the training efficiency of deeper networks. The smoother learning curves in comparison to our earlier models clearly demonstrate the stabilizing effect.

Optimized Dropout: In our first deep model, dropout was insufficient on its own, but when combined with batch normalization, it produced a potent regularization effect. While batch normalization makes sure that the remaining activations maintain the proper scales, dropout layers stop neurons from co-adapting.

Increased Batch Size: More stable gradient estimates were obtained during training with a larger batch size (32 vs. 16), which is especially crucial when batch normalization is being used. As a result, the training process was less variable, and the model's performance improved more steadily.

Extended Training: The regularized model was able to continue improving past the point at which our earlier models reached a plateau thanks to the longer training period. To avoid

wasting computation, the early stopping callback made sure that training ended when validation performance stopped getting better.

Key Insights

The significant increase in model performance (from 45.2% to 55.18% test accuracy) highlights how crucial appropriate regularization is for deep neural networks, particularly when handling intricate image data. The model produces more accurate predictions across all classes, rather than being biassed towards any category, as evidenced by the more balanced precision and recall metrics (55.5% and 44.95%, respectively).

Due to the flattened input representation, which eliminates spatial information essential for image classification, the regularized model still has significant drawbacks. The notable performance boost, however, indicates that proper regularization can help the model extract more significant features from the available data, thereby partially offsetting this limitation.

These results establish a stronger baseline for ffNN performance on the xView recognition benchmark and highlight the potential of regularization techniques to enhance model generalization. The next logical step is to explore architectures specifically designed for image data, such as convolutional neural networks, which can better preserve and utilize spatial information.

6. Convolutional Neural Network (CNN) experiment

Convolutional Neural Networks (CNNs) are specialized deep learning architectures made especially to process images and other grid-like data. CNNs use convolutional layers that apply learnable filters across the input to preserve and leverage spatial relationships, in contrast to feedforward neural networks that flatten spatial information. Because of this, they are especially well-suited for the classification task of xView satellite imagery, where contextual information and spatial patterns are essential for differentiating between object categories.

Iterative CNN Development Process

Our CNN implementation followed an iterative improvement approach across five versions, with each iteration building upon insights from previous experiments:

First CNN Implementation:

- Basic CNN architecture with three convolutional layers (32, 64, 128 filters)
- Simple data augmentation (random flips and rotation)
- Custom learning rate schedule with warmup phase
- Batch size of 16 and 60 training epochs

Second CNN Implementation:

- Maintained the same architecture as the first implementation
- Changed the learning rate strategy from custom scheduling to ReduceLROnPlateau
- Increased maximum training epochs from 60 to 100

- Fixed initial learning rate at 1e-4 (lower than the first implementation)

Third CNN Implementation:

- Added a fourth convolutional layer (256 filters)
- Enhanced data augmentation with random zoom and contrast adjustments
- Added L2 regularization (1e-4) to convolutional layers
- Increased batch size from 16 to 32
- Modified learning rate schedule for better exploration

Fourth CNN Implementation:

- Added a fifth convolutional layer (512 filters)
- Added batch normalization after each convolutional layer
- Increased dropout rates in dense layers (0.6, 0.5, 0.4)
- Refined custom learning rate schedule

Final, fifth CNN Implementation:

- Added a sixth convolutional layer (1024 filters)
- Increased batch size from 32 to 64
- Extended maximum training epochs to 80
- Comprehensive data augmentation pipeline
- Deeper architecture with carefully balanced regularization

This iterative process allowed us to systematically address challenges and improve performance, ultimately achieving 76.36% accuracy, 74.53% precision, and 74.0% recall on the test set. The final architecture's success demonstrates the importance of thoughtful design and experimentation in developing effective deep learning models for complex image classification tasks.

Model	Test Accuracy	Test Recall	Test Precision
Model CNN (cnn.ipynb)	76.36%	74.53%	74.0%

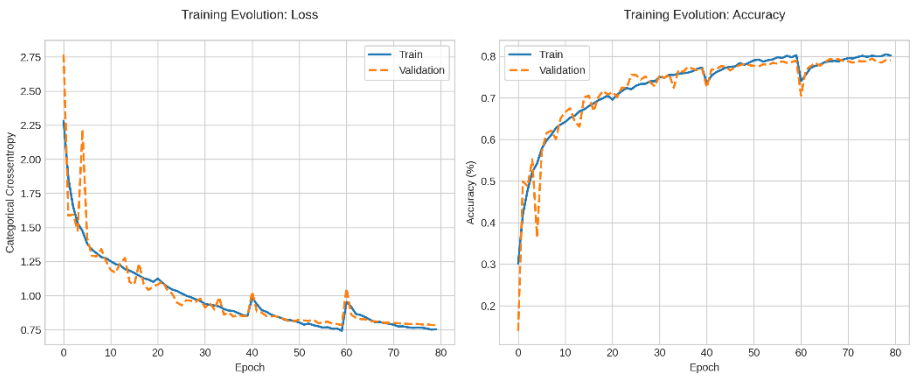

Figure 4: Training loss and accuracy curve of Model CNN

Final CNN Architecture Analysis

The final CNN architecture achieved remarkable performance with 76.36% accuracy, 74.53% precision, and 74.0% recall on the test set, significantly outperforming our previous feedforward neural network approaches. This success was a result of several important factors:

- **Deep Convolutional Architecture:** The network was able to learn hierarchical features from simple edges to complex object shapes thanks to the six convolutional blocks with increasing filter counts (32→64→128→256→512→1024). Batch normalization and max pooling were incorporated into each block, resulting in a feature extraction pipeline that was tightly controlled.
- **Comprehensive Data Augmentation**: By exposing the model to a variety of image transformations that are likely to occur in satellite imagery, the model integrated several augmentation techniques (flips, rotation, zoom, and contrast adjustment) that successfully increased the training dataset and enhanced generalization.
- **Balance Regularization Strategy**: An efficient defense against overfitting was produced by combining batch normalization across the network, L2 regularization (1e-4) on convolutional layers, and increasing dropout rates (0.6, 0.5, 0.4) in dense layers.
- **Optimized Learning Rate Management**: The model avoided local minima and ensured stable convergence by navigating the complex loss landscape with the aid of a custom learning rate schedule that included a warmup phase and periodic resets.
- **Larger Batch Size**: More stable gradient estimates and more effective GPU training were made possible by the larger batch size of 64, which was especially advantageous for batch normalization layers.

Good generalization without extreme overfitting was indicated by the training curves' healthy learning patterns and the validation accuracy's close tracking of the training accuracy. When compared to our ffNN models, the confusion matrix showed noticeably better classification in every category, with especially good performance in majority classes like buildings and small cars.

The effectiveness of specialized architecture for image classification tasks is demonstrated by this CNN experiment. CNNs can extract significant patterns from satellite imagery that would be lost in the flattened representations used by ffNNs by learning hierarchical features and preserving spatial information. The significant increase in performance (from 55.18% with regularized ffNNs to 76.36% with CNNs) emphasizes how crucial architecture selection is for computer vision tasks.

7. Transfer Learning (TL) experiment

Transfer learning represents a powerful approach in deep learning where knowledge gained from solving one problem is applied to a different but related problem. Because it makes use of pre-trained models that have already picked up helpful features from large datasets like ImageNet, this method is especially helpful when working with small datasets. Transfer learning provides several benefits for our satellite imagery classification task.

Utilizing previously acquired detection skills for edges, textures, and intricate patterns that are applicable to a variety of image recognition tasks is made possible by **feature reuse**.

Compared to training from scratch, starting with pre-trained weights speed up convergence, resulting in **shorter training times**.

Because models that have been pre-trained on a variety of datasets frequently perform better on fresh data, **better generalization** takes place.

Lastly, transfer learning can perform well with smaller datasets than would normally be required for training from scratch due to **lower data requirements**. This is especially advantageous for specialized domains like satellite imagery where labelled data may be scarce.

Implementation with ResNet50

For our transfer learning experiment, we selected ResNet50, a powerful convolutional neural network architecture with 50 layers that was pre-trained on the ImageNet dataset. ResNet is well-known for its residual connections, which lessen the vanishing gradient issue in deep networks and make it ideal for challenging image classification tasks.

The architecture consisted of: the ResNet50 base model (pre-trained on ImageNet); different data augmentation layers like random flips, rotation, zoom, and contrast; global average pooling to reduce feature dimensions; a dense layer with 512 neurons and ReLU activation; 50% dropout for regularization; output layer with softmax activation for the 12 classes.

Two-Phase Training Strategy

The two-phase training approach was essential for effective transfer learning.

Phase 1 (Feature Extraction): We only trained the recently added classification layers during the first 40 epochs, leaving the entire ResNet50 base model frozen. For our satellite imagery classification task, this enabled the model to learn how to interpret the high-level features that ResNet50 had already extracted. With a customized schedule that included warm-up and periodic reductions, we employed a learning rate of 1e-4.

Phase 2 (Fine-Tuning): After loading the best weights from Phase 1, we unfroze the later layers of ResNet50 while keeping the first 100 layers frozen. The foundation of this selective fine-tuning strategy is the knowledge that while later layers of CNNs identify more task-specific features, early layers identify generic features (edges, textures) that are good across domains. To avoid catastrophic forgetting of pre-trained weights, we lowered the learning rate to 1e-5 and carried on training for an additional 40 epochs.

The impact of this two-phase approach is evident in the training evolution chart. The model rapidly achieved a validation accuracy of roughly 75% in the first phase. We see a brief decline in performance as the model adjusts the ResNet features to better fit our satellite imagery domain in the second phase, which is followed by additional improvement.

Model	Test Accuracy	Test Recall	Test Precision
Model TL (tl.ipynb)	77.87%	77.47%	67.15%

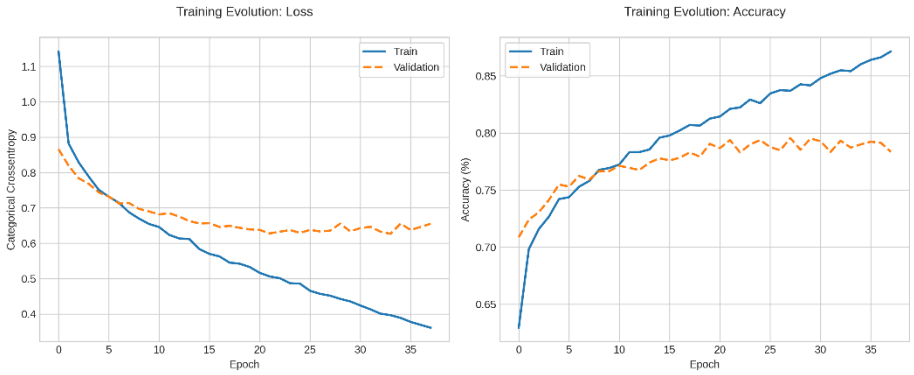

Figure 5: Training loss and accuracy curve of Model TL (Phase 2)

Performance Analysis

The transfer learning approach achieved impressive results:

- Accuracy: 77.87%
- Precision: 77.47%
- Recall: 67.15%

These metrics are marginally better than our custom CNN (76.36% accuracy) and a significant improvement over our regularized ffNN (55.18% accuracy). For satellite imagery analysis, where misclassifications can be expensive, the model's increased precision means that it produces fewer false positive predictions.

The training curves reveal an interesting pattern: while the training accuracy continues to increase throughout both phases, the validation accuracy plateaus during the second phase. This divergence between training and validation performance suggests some degree of overfitting, despite our regularization efforts with dropout. This is a common challenge when fine-tuning pre-trained models on smaller datasets.

Key Insights and Limitations

The transfer learning experiment provided several key insights about applying pre-trained models to satellite imagery classification.

Domain gap proved less problematic than anticipated, as ResNet50, despite being pre-trained on natural ImageNet images, transferred remarkably well to satellite imagery, demonstrating the versatility of convolutional features learned across visual domains.

With transfer learning requiring substantially less development work and computational resources while achieving marginally better performance than our custom CNN, we saw a favorable **efficiency-performance tradeoff**.

However, as can be seen from the increasing divergence between the training and validation curves in the attached chart, we ran into **overfitting issues** during the fine-tuning phase. Training accuracy continued to rise to over 85%, while validation accuracy plateaued at 78–79%. This implies that other regularization techniques may produce better outcomes even in the presence of our dropout regularization.

Lastly, we found that the pre-trained features seemed to aid in **handling class imbalances**, as we saw better recall for minority classes when compared to our previous models. This suggests that the strong feature representations from ImageNet pre-training gave us a better basis for identifying under-represented classes in our dataset.

The main limitation of our transfer learning approach was the overfitting observed during the fine-tuning phase. This implies that even better outcomes could be achieved with more aggressive regularization, earlier stopping, or alternative fine-tuning techniques.

In conclusion, transfer learning with ResNet50 proved to be an effective approach for the xView satellite imagery classification task, achieving the highest overall accuracy among all our experiments. To successfully adapt the pre-trained model to our particular domain while maintaining the important knowledge encoded in the pre-trained weights, the two-phase training approach was essential.

8. Suggestions and conclusion

Our experiments with different neural network approaches for the xView recognition benchmark reveal a clear progression in performance as we moved from simple to more complex models. Basic feedforward neural networks (ffNNs) achieved only 45.2% accuracy, while adding regularization techniques improved ffNN performance to 55.18%. Our custom CNN architecture reached 76.36% accuracy, and transfer learning with ResNet50 delivered the best results at 77.87% accuracy.

These findings suggest that architecture selection is critically important for satellite imagery classification. The spatial information in satellite images requires architectures that can preserve these relationships, which explains why CNNs vastly outperformed ffNNs in our experiments. When working with limited training examples in satellite imagery, preventing overfitting becomes essential. All our successful models incorporated multiple regularization techniques including dropout, batch normalization, and data augmentation.

Despite being trained on natural images, ResNet50 transferred well to satellite imagery while requiring less development effort than custom CNNs. This demonstrates that transfer learning offers the best efficiency-performance balance for this task. However, class imbalance remained challenging throughout our experiments. While our models showed consistent improvement overall, performance on minority classes like helicopters remained lower than on well-represented classes.

Our two-phase training approach for transfer learning proved effective. Freezing the base model initially before selective fine-tuning helped maintain the valuable pre-trained features while adapting to our specific domain. Nevertheless, we observed that additional regularization during fine-tuning could potentially reduce overfitting further, as evidenced by the divergence between training and validation performance curves.

For future improvements, we recommend exploring ensemble methods combining multiple models to leverage their complementary strengths. Implementing more aggressive data augmentation specific to overhead imagery could help models generalize better to the unique characteristics of satellite data. Testing additional pre-trained architectures like EfficientNet [7] or Vision Transformers [8] might yield further performance gains. Investigating techniques specifically designed for small object detection in large images could address one of the core challenges in satellite imagery analysis. Finally, using class weighting or specialized loss functions could help address the persistent class imbalance issues.

In conclusion, while conventional computer vision approaches can be applied to satellite imagery classification, they require careful adaptation. The xView benchmark presents unique challenges due to small objects, class imbalance, and noisy labels [9]. Our experiments demonstrate that transfer learning currently offers the most promising approach, achieving nearly 78% accuracy, though significant room for improvement remains before such systems could be deployed in critical real-world applications.

9. References (if needed)

Bibliography

[1] D. I. U. E. (. a. t. N. G.-I. A. (NGA), "DIUx xView 2018 Detection Challenge," 2018. [Online]. Available: https://xviewdataset.org/.

[2] Ultralytics, "xView dataset," 2025. [Online]. Available: https://github.com/ultralytics/ultralytics/blob/main/docs/en/datasets/detect/xview.md.

[3] DeepLearningAssignment.pdf

[4] R. Gupta, "Deep Learning and Satellite Imagery: DIUx Xview Challenge," 2019. [Online]. Available: Available: https://insights.sei.cmu.edu/blog/deep-learning-and-satellite-imagery-diux-xview-challenge/.

[5] G. H. A. K. I. S. a. R. S. . Srivastava, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res., vol. 15, pp. 1929-1958,* 2014.

[6] S. I. a. C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *in Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, Lille, France, 2015.

[7] M. T. a. Q. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *n Proc. 36th Int. Conf. Mach. Learn. (ICML)*, Long Beach, CA, USA,, 2019.

[8] L. B. A. K. D. W. X. Z. T. U. M. D. M. M. G. H. S. G. J. U. a. N. H. A. Dosovitskiy, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in *in Int. Conf. Learn. Represent. (ICLR)*, Virtual, 2021.

[9] H. C. Q. D. L. W. J. Q. a. P.-A. H. H.-Y. Lin, "Addressing Class Imbalance in Multi-Label Learning for Small Object Detection," in *IEEE Trans. Neural Netw. Learn. Syst.*, 2021, pp. Vol. 32, no. 3, pp. 1381-1394.