CAPSTONE PROJECT REPORT

Karthik Manivannan
STUDENT ID: 1229717

INFO6147 DEEP LEARNING WITH PYTORCH

—

Remote Sensing Image Classification
Using Down Sampled RSI-CB256

—

Professor Mohammed Yousefhussien

INTRODUCTION

The objective of this capstone project is to develop and train deep learning models for classifying remote sensing images using a down sampled version of the RSI-CB256 dataset.

This project aims to compare the performance of a custom Convolutional Neural Network (CNN) with the pre-trained ResNet-18 model to understand their strengths and weaknesses in remote sensing image classification tasks.

RSI-CB256 DATASET

Dataset Description

- **Original Dataset**: The RSI-CB256 dataset contains 45 classes with a total of 24,747 high-resolution images.

- **Down sampled Dataset**: For this project, the dataset was reduced to 15 classes and approximately 12,500 images to facilitate ease of implementation and accommodate available compute power.

Classes

- The original RSI-CB256 dataset contains 45 distinct land cover classes, including a diverse range of environments such as forests, deserts, urban areas, and agricultural sites like coffee plantations and orchards.
- These classes encompass various structures and landscapes, from airports and bridges to parks and schools, for training and evaluating deep learning models in remote sensing image classification.

Images

- The dataset includes 24,747 high-resolution images, each with dimensions of 256x256 pixels, ensuring detailed and quality data for effective model training and evaluation.

- These images are categorized into 45 diverse land cover classes, making the dataset challenging for developing robust deep learning models for remote sensing image classification.

- The diversity and high resolution of the images helps in capturing important details and patterns necessary for accurate classification across various land cover types.

DATASET PREPROCESSING

1. **Resizing**: All images were resized to a consistent resolution of 256x256 pixels.

   - This ensures uniformity in the input dimensions for the neural network, facilitating efficient batch processing and improving the model's ability to learn spatial hierarchies effectively.

2. **Normalization**: Pixel values of the images were normalized using the mean values [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225].

   - This normalization aligns the image data with the pre-trained models' expected input distributions, enhancing convergence during training.

3. **Data Augmentation**: Several data augmentation techniques were applied to increase the diversity of the training data, helping to prevent overfitting and improve generalization.

   - Random Horizontal Flip: Flipping images horizontally with a 50% probability.

   - Random Rotation: Rotating images randomly within a range of -10 to 10 degrees.

   - Random Resized Crop: Randomly cropping and resizing images to 256x256 pixels within a scale range of 0.8 to 1.0 of the original size.

4. **Splitting**: The dataset was split into training, validation, and test sets in the ratio of 80%, 10%, and 10%, respectively. This split ensures a robust evaluation of the model's performance and helps in fine-tuning the hyperparameters.

   - Training Set: Contains 19,797 images, used to train the models.

   - Validation Set: Contains 2,474 images, used to tune hyperparameters and prevent overfitting.

   - Test Set: Contains 2,476 images, used to evaluate the final model performance on unseen data.

Custom CNN

The custom CNN model was designed with the following architecture:

- **Convolutional Layers:** Three convolutional layers with increasing filter sizes (64, 128, 256) to capture spatial hierarchies in the data.
  - First Convolutional Layer:
  - 64 filters, each with a kernel size of 3x3.
  - Second Convolutional Layer:
  - 128 filters, each with a kernel size of 3x3.
  - Third Convolutional Layer:
  - 256 filters, each with a kernel size of 3x3.
  - All with a stride of 1 and padding of 1 to maintain the dimensions.
  - These layers increase the depth of the feature maps, allowing the model to capture more complex patterns and details in the images.

- **Activation Function:** ReLU Activation Function is applied after each convolutional layer to introduce non-linearity into the model
  - This enables it to learn more complex functions and patterns within the data.

- **Max Pooling Layers:** It is added after each convolutional layer.
  - Pool size of 2x2 and stride of 2, which reduces the spatial dimensions of the feature maps by half, thus down-sampling the data and reducing computational complexity.
  - This process helps in extracting dominant features and making the model more robust to spatial variances in the input images.

- **Fully Connected Layers:**
  - The first fully connected layer has an input size of 256 * 32 * 32 and 1024 neurons from the flattened output of the final convolutional layer.
  - The second fully connected layer connects the 1024 neurons to the output layer with 15 neurons, corresponding to the number of classes.
  - Dropout regularization is applied to these layers to prevent overfitting by randomly setting a fraction of input units to 0 during training.

MODEL SELECTION AND ARCHITECTURE

ResNet-18

- The ResNet-18 model is a pre-trained deep learning model from the PyTorch torchvision.models library, known for its effectiveness in image classification tasks.

- The ResNet architecture employs residual connections that help mitigate the vanishing gradient problem, enabling the construction of deeper networks.

➢ **Residual Connections:** These are identity shortcuts that skip one or more layers, allowing the model to learn residual mappings instead of direct mappings.
  - This helps to maintain flow of gradients during backpropagation, making the network more robust and easier to train.

➢ **Convolutional Layers:** ResNet-18 consists of multiple convolutional layers organized into residual blocks.
  - Each block has two convolutional layers followed by batch normalization and ReLU activation.

➢ **Downsampling:**
  - Some residual blocks downsample the input using convolutional layers with a stride of 2, which effectively halves the spatial dimensions while doubling the depth of feature maps.

➢ **Fully Connected Layers:**
  - For this project, the final fully connected layer of ResNet-18 was modified to match the number of classes (15) in the downsampled dataset, enabling the model to classify the specific land cover types present in the dataset.

➢ **Pre-trained Weights**: The model uses weights pre-trained on a large dataset which provides a strong start that helps achieve better performance and faster convergence during training.

MODEL TRAINING & HYPERPARAMETER TUNING

Training Process

- **Optimization:** Both models were trained using the Adam optimizer with a learning rate scheduler (StepLR with step size 7 and gamma 0.1).
- **Metrics:** The training process was monitored using loss and accuracy metrics.
- **Epochs:** Both models were trained for 15 epochs.
- D**ropout:** Dropout layers were included in the fully connected layers of the custom CNN to prevent overfitting.

Hyperparameter Tuning

- Hyperparameter tuning involved experimenting with different learning rates (0.001, 0.0005) and batch sizes (64, 128) to optimize the model's performance.
- The results of the hyperparameter tuning are summarized below:

Model	Learning Rate	Batch Size	Accuracy	Precision	Recall	F1-Score
Custom CNN	0.001	64	0.9224	0.9219	0.9224	0.9220
ResNet-18	0.001	64	0.9888	0.9889	0.9888	0.9888
Custom CNN	0.001	128	0.9424	0.9419	0.9424	0.9416
ResNet-18	0.001	128	0.9888	0.9888	0.9888	0.9888
Custom CNN	0.0005	64	0.9624	0.9623	0.9624	0.9621
ResNet-18	0.0005	64	0.9912	0.9913	0.9912	0.9912
Custom CNN	0.0005	128	0.9680	0.9680	0.9680	0.9679
ResNet-18	0.0005	128	0.9912	0.9913	0.9912	0.9912

Evaluation & Final Model Testing

Evaluation Process

The metrics for the models were calculated on the validation dataset.

- **Custom CNN:** With a learning rate of 0.001 and a batch size of 64, the custom CNN achieved an accuracy of 92.24%, precision of 92.19%, recall of 92.24%, and an F1-score of 92.20%.
- The best performance for the custom CNN was achieved with a learning rate of 0.0005 and a batch size of 128, resulting in an accuracy of 96.80%, precision of 96.80%, recall of 96.80%, and an F1-score of 96.79%.
- **ResNet-18:** With a learning rate of 0.001 and a batch size of 64, ResNet-18 achieved an accuracy of 98.88%, precision of 98.89%, recall of 98.88%, and an F1-score of 98.88%.
- The best performance for ResNet-18 was achieved with a learning rate of 0.0005 and a batch size of 64, resulting in an accuracy of 99.12%, precision of 99.13%, recall of 99.12%, and an F1-score of 99.12%.

Final Model Testing

- The final models were tested on the held-out test dataset to assess their generalization to unseen data.
- The performance are summarized in the tables below:

Custom CNN

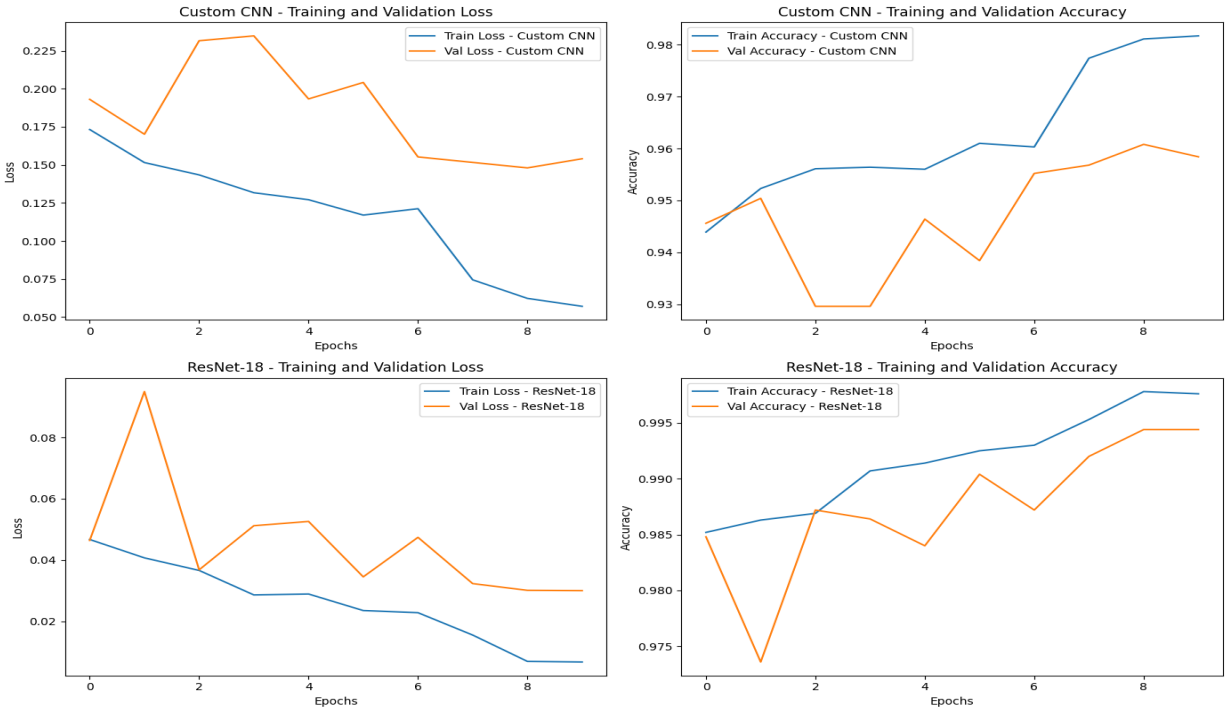
Metric	Validation Set	Test Set
Accuracy	96.24%	95.52%
Precision	96.23%	95.62%
Recall	96.24%	95.52%
F1-Score	96.21%	95.52%

ResNet-18

Metric	Validation Set	Test Set
Accuracy	99.12%	98.80%
Precision	99.13%	98.80%
Recall	99.12%	98.80%
F1-Score	99.12%	98.80%

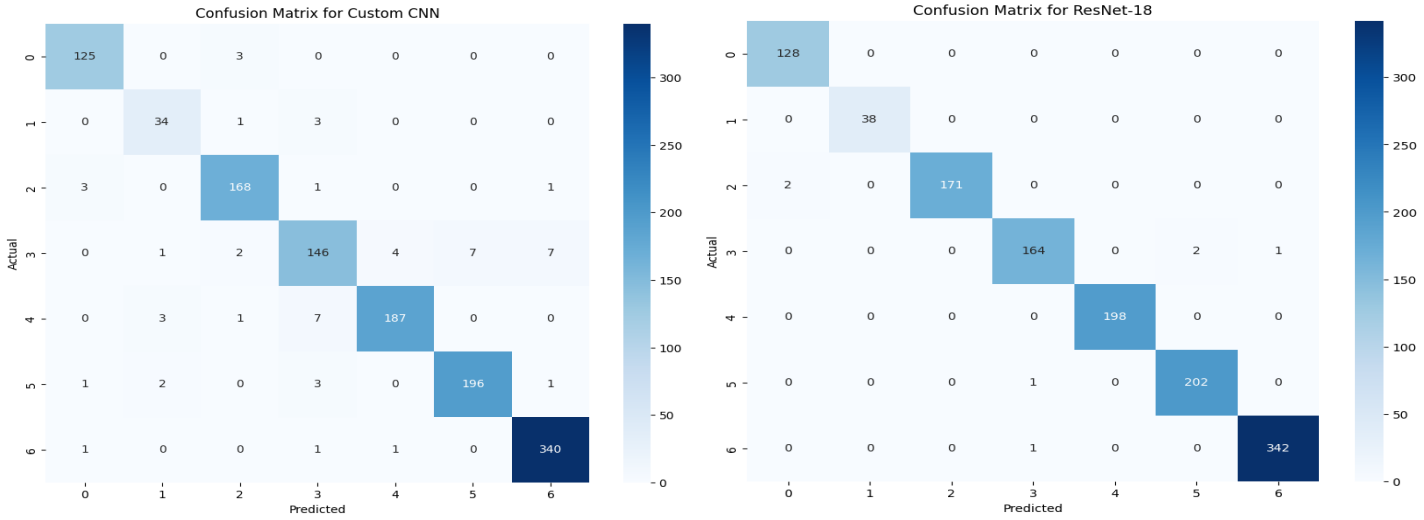
Training and Validation Loss Plots

- This shows how each model's performance metrics evolved over the epochs, with ResNet-18 achieving lower validation loss and higher validation accuracy compared to the Custom CNN, indicating better performance and generalization.



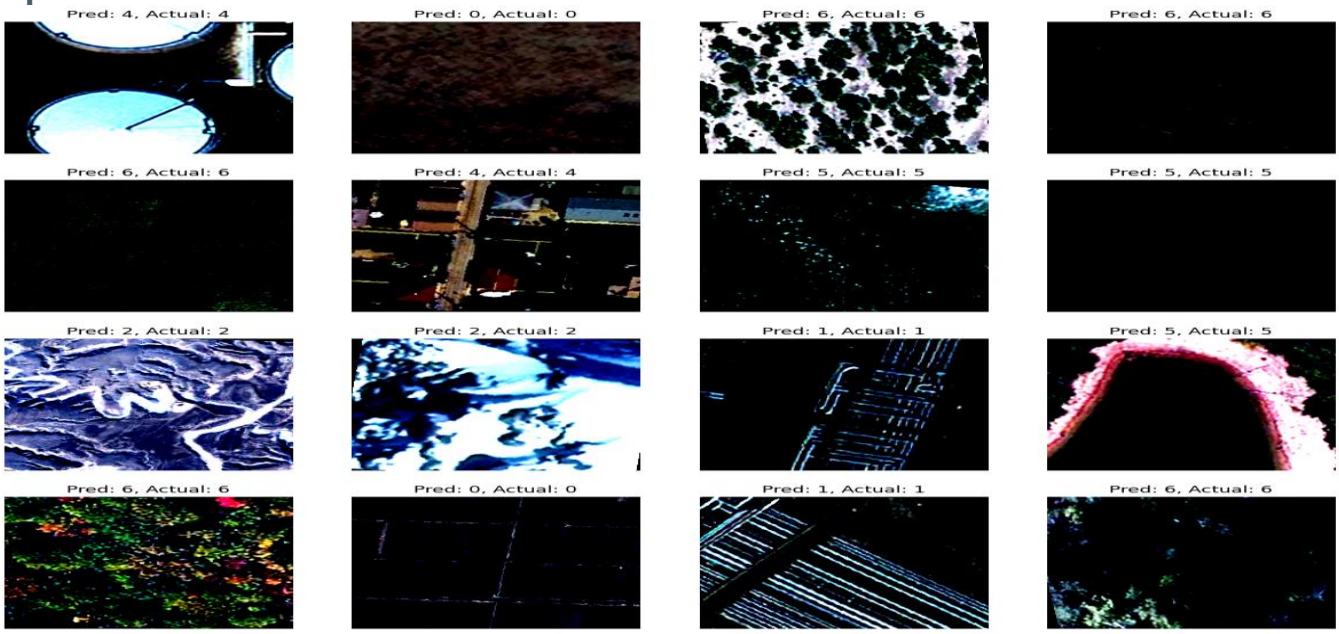
Confusion Matrix

- The Custom CNN model shows a high level of accuracy with some misclassifications in specific classes, whereas the ResNet-18 model demonstrates superior performance with minimal misclassifications, indicating its higher accuracy and better generalization.

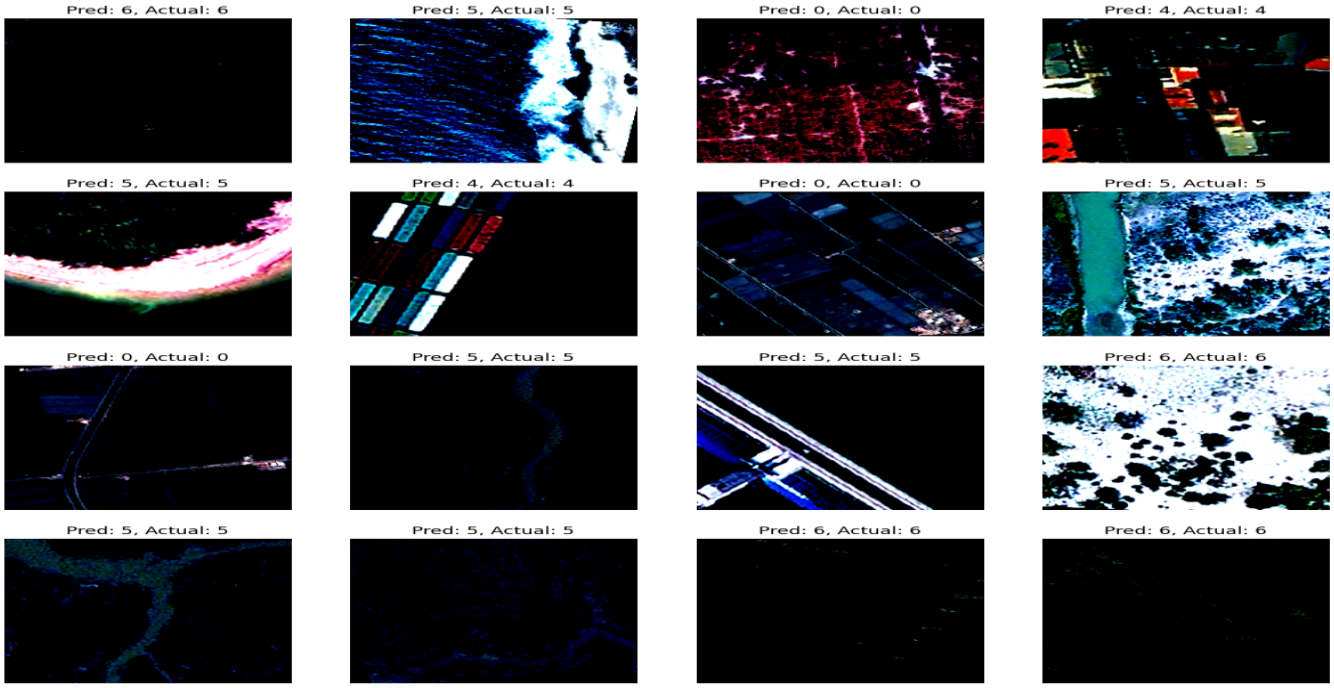


- Visual representations of both model predictions on test images, highlighting correctly and incorrectly classified examples.

Sample Predictions CNN



Sample Predictions ResNet-18



KEY FINDINGS, CHALLENGES AND FUTURE WORK

Key Findings

- **Performance:** ResNet-18 significantly outperformed the custom CNN across all metrics, including accuracy, precision, recall, and F1-score.
- **Comparison**: While ResNet-18 showed superior performance, the custom CNN also achieved high accuracy, indicating that even simpler models can be quite effective with proper design and training.

Challenges

- **Large Dataset:** Managing the large RSI-CB256 dataset required efficient data handling techniques and heavy computational resources.
- **Computational Resources:** GPU acceleration and optimized data pipelines were important, but resource limitations remained a challenge.
- **Hyperparameter Tuning:** Required lots of experimentation and research to find the optimal settings for learning rate, batch size, and epochs. Multiple experiments were necessary to identify the optimal settings for achieving the best performance.
- **Regularization:** Implementing techniques like dropout was crucial to prevent overfitting.

Future Work

- **Advanced Architecture:** Explore more advanced and deeper neural network techniques like DenseNet or transformer models to further enhance classification performance.
- **Data Augmentation:** Implementing additional data augmentation methods could improve model generalization
- **Multi-Modal Integration:** Using data from different sources like multispectral images might improve classification accuracy.
- **Model Ensembling:** Combining multiple models could enhance robustness and accuracy.
- **Diverse Datasets Experimentation:** Testing models on various remote sensing datasets to assess generalizability.
- **Benchmarking:** Comparing with state-of-the-art methods to identify further improvement areas.

CONCLUSION

1. The project effectively demonstrated the superior performance of pre-trained models like ResNet-18 in remote sensing image classification, highlighting their robustness and efficiency in handling high-resolution satellite imagery like RSI-CB256 dataset.
2. The comparison between the custom CNN and ResNet-18 highlighted the effectiveness of pre-trained models in remote sensing applications.
3. Achieved significant accuracy and performance metrics, validating the effectiveness of the models used.
4. Pre-trained models like ResNet-18 are highly effective for remote sensing applications, demonstrating superior performance in classification tasks.
5. The custom CNN model, while effective, showed more variability in performance across different classes.
6. The custom CNN model has more scope of potential areas for further optimization and improvement.
7. Handling the large dataset and ensuring efficient training was a significant challenge due to the computational resources required.
8. Fine-tuning hyperparameters for optimal performance took multiple iterations and careful evaluation to achieve the best results.
9. These insights from this project can be utilized in various applications such as urban planning, environmental monitoring, and disaster management.

REFERENCES

- INFO6147 Deep Learning with Pytorch Course Materials and Jupyter Notebooks.
- https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns
- https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8
- https://billtcheng2013.medium.com/image-classification-with-transfer-learning-on-pytorch-2d718c85b58f
- ResNet-18:
- https://waww.geeksforgeeks.org/introduction-convolution-neural-network/
- https://medium.com/@sasirekharameshkumar/deep-learning-basics-part-8-convolutional-neural-network-cnn-4cff567bad46
- https://medium.com/@navarai/unveiling-the-diversity-a-comprehensive-guide-to-types-of-cnn-architectures-9d70da0b4521
- https://www.datacamp.com/courses/deep-learning-for-images-with-pytorch
- https://towardsdatascience.com/grid-search-for-model-tuning-3319b259367e
- https://medium.com/pythoneers/hyperparameter-tuning-in-data-science-ad1a03d830b9
- https://www.datacamp.com/tutorial/complete-guide-data-augmentation
- javatpoint.com/data-augmentation-in-machine-learning
- https://github.com/bentrevett/pytorch-image-classification
- https://github.com/PacktPublishing/Deep-Learning-for-Computer-Vision
- https://towardsdatascience.com/cnn-transfer-learning-fine-tuning-9f3e7c5806b2
- https://rumn.medium.com/part-1-ultimate-guide-to-fine-tuning-in-pytorch-pre-trained-model-and-its-configuration-8990194b71e
- https://medium.com/womenintechnology/data-preprocessing-steps-for-machine-learning-in-phyton-part-1-18009c6f1153
- https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8
- https://neptune.ai/blog/data-preprocessing-guide
- https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- https://medium.com/@zhonghong9998/evaluating-model-performance-a-comprehensive-guide-6f5e7c11409f
- https://towardsdatascience.com/good-approach-to-evaluate-your-machine-learning-model-e2e1fd6aa6bb
- https://github.com/anubhavparas/image-classification-using-cnn
- https://github.com/sinanw/cnn-image-classification
- https://github.com/jeffprosise/Deep-Learning/blob/master/Image%20Classification%20(CNN).ipynb
- https://github.com/rishivar/Resnet-18
- https://github.com/openvinotoolkit/open_model_zoo/blob/master/models/public/resnet-18-pytorch/
- https://github.com/Moddy2024/ResNet-18
- https://www.kaggle.com/datasets/mahmoudreda55/satellite-image-classification
- https://github.com/lehaifeng/RSI-CB
- https://huggingface.co/datasets/jonathan-roberts1/RSI-CB256
- https://www.kaggle.com/code/priyanksanghvitech/satellite-image-classification-using-cnn
- https://aitlas.readthedocs.io/en/latest/examples/multiclass_classification_example_rsi_cb256.html
- https://www.researchgate.net/figure/Confusion-matrices-for-the-RSI-CB256-dataset-a-validation-b-testing_fig2_355752357