SATELLITE IMAGE SEGMENTATION

Aiqi Zhang
Student# 1009253521
aiqiangela.zhang@mail.utoronto.ca

Emma Wang
Student# 1008956612
emmaxt.wang@mail.utoronto.ca

Kara Ning
Student# 1008877789
kara.ning@mail.utoronto.ca

Yijie Wang
Student# 1009020006
yijies.wang@mail.utoronto.ca

ABSTRACT

Satellite image analysis has a wide range of real-world applications, from tracking land cover evolution to identifying natural disasters. Automatic categorization and segmentation of land covers has become crucial across various fields. In the Satellite Image Segmentation project, the team built and trained an autoencoder to facilitate this process, taking an input satellite image and outputting a colour-coded image indicating the type of land types. This report details the model architecture, presents quantitative and qualitative results, and discusses the model's performance compared to the baseline model. —-Total Pages: 9

1 INTRODUCTION

The advancement of satellite imagery has opened new horizons in understanding and interpreting the Earth's surface, offering insights into the complex dynamics of landscapes across the globe. Automatic categorization and segmentation of land covers has become crucial across various fields, such as urban planning, agriculture, disaster management, and environmental monitoring. With the exponential growth of remote sensing data, there is also a growing need for efficient and accurate image segmentation techniques, which motivated this project[1].

Machine learning is an excellent approach for satellite image segmentation due to its proficiency in managing and interpreting complex datasets. It can effectively extract meaningful insights from high-dimensional data typical of satellite images, a task that would be challenging and time-consuming otherwise. Leveraging deep learning's automation capabilities, it effectively reduces the time to analyze thousands of satellite images for landscape analysis. This project leverages the vast source of satellite images through the development of a sophisticated neural network model to output the distribution of different landscape classes as a segmented image and calculate their respective percentages from given satellite images. This project is interesting because many satellite-related projects focus on single-label classification, where the model categorizes an image according to the predominant landscape. In our project, we aim to create a model that can output a masked image with annotation for land cover types.

The model is trained directly with satellite images and their corresponding labelled class proportions. The input of the model is a raw image taken from the satellite, while the output is the landscape class proportions from the image, as shown in Figure 1.

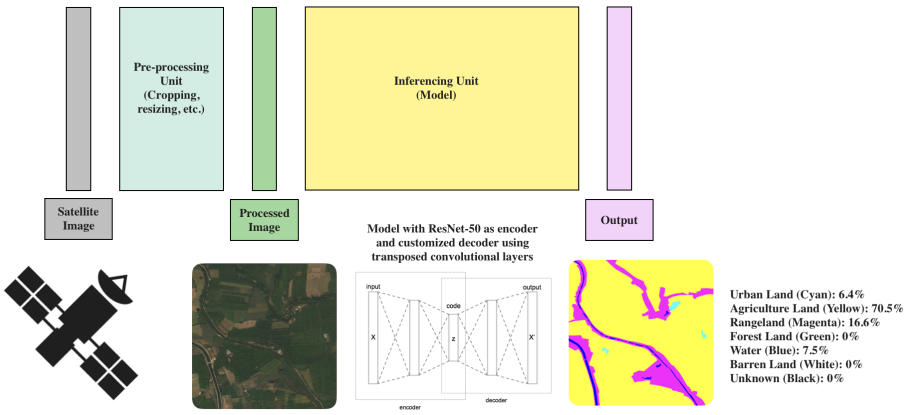

Figure 1: Overview of the project including the inputs and outputs

2 Background & Related Work

Through extensive research, we found many related literature using various deep learning techniques and optimization methods to analyze satellite images.

Many applications focus on single-label classification. Convolutional neural networks have been used for feature extraction with pre-trained models AlexNet, VGG19, GoogleNet and Resnet50 [2] [3]. The first paper classified each satellite image into one class, such as airport. It was found that Resnet50 has the highest accuracy and minimum loss value than other methods and successfully works on different datasets [2].

Figure 2: Single-label satellite image classification [2]

A paper on road segmentation in SAR images evaluated fully convolutional neural networks (FCNNs). It was found that although FCNNs natively lack efficiency for road segmentation, they are capable of good results if properly tuned [4]. Another similar work used two deep neural networks: a convolutional neural network for image classification followed by a residual network for image segmentation. The image classification was done through a VGG16 network while the image segmentation was done through a U-Net [5]. The third related work on optimization techniques for satellite segmentation applications analyzed different optimization algorithms, such as modified artificial bee colony (MABC) algorithm, ABC algorithm, particle swarm optimization (PSO) using different objective functions to find the optimized multilevel thresholds [6]. A work on multispectral data compared two methods of image classification, i.e. ML (Maximum Likelihood), a supervised method, and ISODATA (Iterative Self-Organizing Data Analysis Technique), an unsupervised

method [7]. ML classified the study area into 11 classes, which was chosen earlier, with accuracy 97% ($\kappa = 0.97$), while only eight can be clustered by ISODATA with accuracy 93% ($\kappa = 0.91$) [7].

3 Data Processing

3.1 Data Collection

The dataset was obtained from the Land Cover Classification Track in the DeepGlobe Challenge [8]. The dataset contains 803 satellite images in RGB, each with a 50cm pixel resolution and a size of $2448x2448$ pixels. Each image is paired with a mask image as land cover annotation. The mask is an RGB image with 7 classes of labels, with colour coding and distribution as follows:

```
{'Urban land': '0,255,255 - Man-made, built up areas with human artifacts (can ignore roads for now which is hard to label)',
 'Agriculture land': '255,255,0 - Farms, any planned (i.e. regular) plantation, cropland, orchards, vineyards, nurseries, and ornamental
horticultural areas; confined feeding operations.',
 'Rangeland': '255,0,255 - Any non-forest, non-farm, green land, grass.',
 'Forest land': '0,255,0 - Any land with x% tree crown density plus clearcuts.',
 'Water': '0,0,255 - Rivers, oceans, lakes, wetland, ponds.',
 'Barren land': '255,255,255 - Mountain, land, rock, dessert, beach, no vegetation',
 'Unknown': '0,0,0 - Clouds and others'}
```

Figure 3: Colour coding labels [2]

3.2 Data Augmentation

Because the data are large in size and have high resolution, all images are cropped into 16 smaller images. Images are resized to $256x256$ when training the model. In order to expand the training dataset even further and enhance model generalization capabilities, several data augmentation techniques were applied. The parameters for these processes are chosen randomly. The techniques are as follows:

1. Vertical and horizontal flips
2. 90-degree clockwise and counterclockwise rotations
3. Colour jittering to adjust brightness and contrast

Figure 4 illustrates a visual representation of the final cleaned sample. On the left side, the data is shown unaugmented, while on the right, the data is shown after the above-mentioned techniques have been applied. In addition, each RGB channel in the images are normalized by multiplying original value by 0.5.

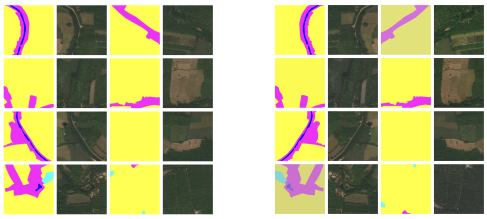

Figure 4: Original and augmented cropped data

As a result of a series of data cleansing and data augmentation processes, the final dataset contains a total of 19232 training images and 1616 images for validation and testing. Each image has a mask image as the label.

3.2.1 RGB Normalization

The RGB values are divided by 255 to normalize to be within the range of [0, 1], with the distribution shown in the histogram on the left. A linear transformation is applied to the values using the distribution's standard deviation and mean, thus, the RGB values were within the range [-3, 3].

The distribution is shown on the right. The effect of RGB normalization on the images is shown below. Having inputs centered around zero proved beneficial because it helped the network learn more efficiently, ensuring that the initial gradients are not too small or too large.

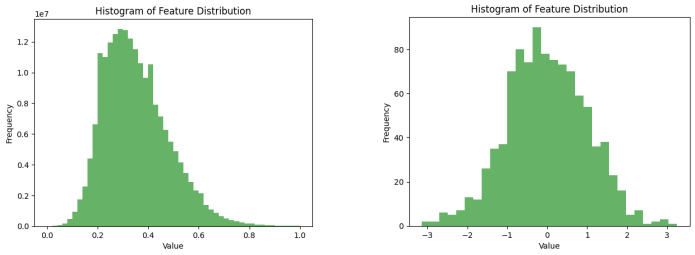

Figure 5: Original RGB Distribution (left) and Normalized RGB Distribution (right)

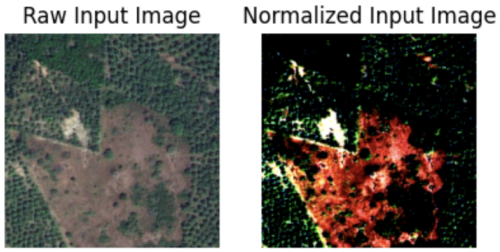

Figure 6: Original and normalized images [2]

4 Model Architecture

To construct our final model, we explored traditional strategies of encoder-decoder architecture and took inspiration from existing models, such as ResNet-50 and U-Net [9]. In this model, the encoder progressively downsamples the image to a more compact representation, and the decoder upsamples this representation to reconstruct the image, as shown in Fig. 10.

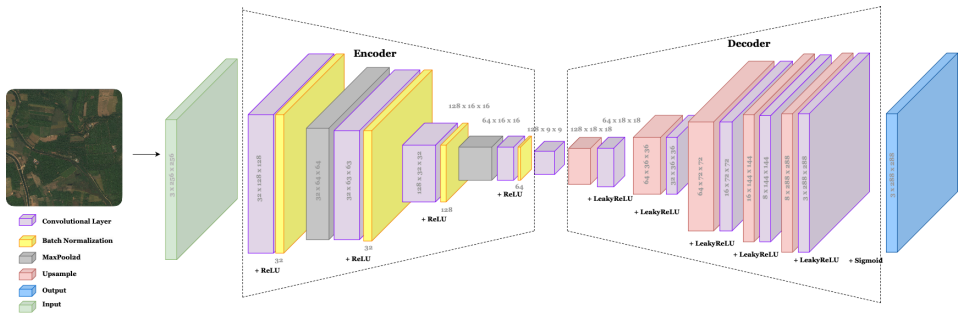

Figure 7: Model architecture of final model

The encoder portion consists of 5 convolutional layers. The ReLU activation function is used to introduce non-linearity into the network, allowing it to learn complex patterns. Normalizing the inputs to layers within the network helps stabilize the learning process and reduces internal covariate shifts, allowing higher learning rates and reducing the sensitivity to initialization. MaxPooling layers downsamples the data to reduce the spatial dimensions of the feature map.

The decoder consists of another 5 convolutional layers. LeakyReLU is used in the decoder to help with the flow of gradients during backpropagation and mitigate the vanishing gradient problem.

Upsampling is used here to reverse MaxPooling, reconstructing the spatial dimensions. Sigmoid is used as the final activation function to map the output between 0 and 1.

5 Baseline Model

The baseline model is a simple autoencoder with three convolutional layers in the encoder, and three transposed convolutional layers in the decoder. The LeakyReLU activation function is used. The hyperbolic tangent activation function is used as the final output activation function. The architecture is shown below:

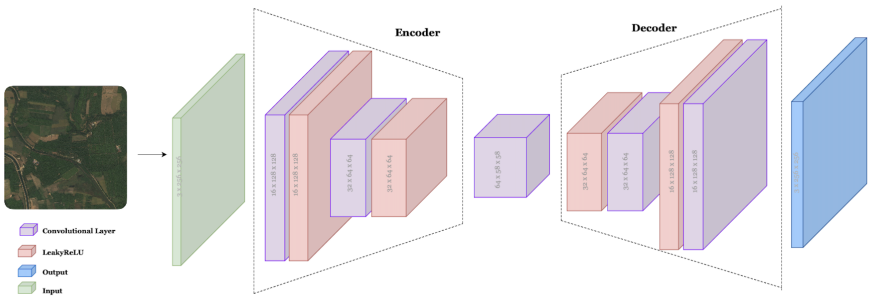

Figure 8: Model architecture of baseline model

The best parameter for the baseline model is with a 256 batch size, 0.0007 learning rate, and 80 epochs. The loss function used is the mean squared error (MSE) and the optimizer is Adam. The final training and validation accuracy are 0.4649 and 0.4970 with losses of 0.0152 and 0.0125. Both training and validation accuracy curves are gradually increasing and then stabilizing to reach a plateau at the final accuracy, which is slightly under 0.5. The results can be seen here:

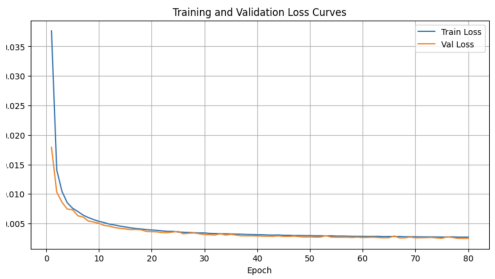

Figure 9: Baseline model loss

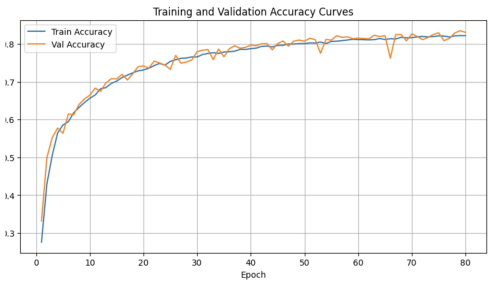

Figure 10: Baseline model accuracy

6 Final Model

6.1 Quantitative Result

The performance of the final model is assessed through a comprehensive set of quantitative measurements. In terms of accuracy metrics, the model evaluation process demonstrates strong performance at different stages, the numerical results are: 0.8792 for training, 0.8001 for validation, and 0.7999 for testing. Accuracy is calculated by comparing pixel values between the outputted image and the label using the mean squared error. The equation is given as follows:

```
# Calculate accuracy
total_pixels_test += inputs_test.numel()
correct_pixels_test += torch.sum(torch.abs(outputs_test - labels_test) < 0.05).item()
```

Figure 11: Loss is calculated by using the mean squared error function

The loss statistics for training, validation, and testing are 0.0225, 0.0714, and 0.0722, respectively.

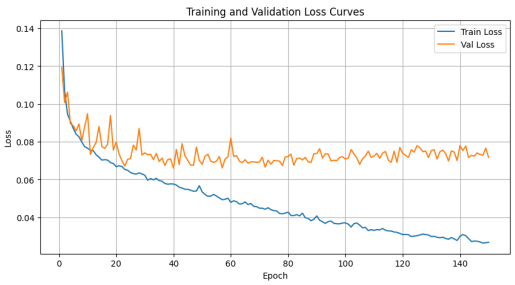

Figure 12: Final model loss

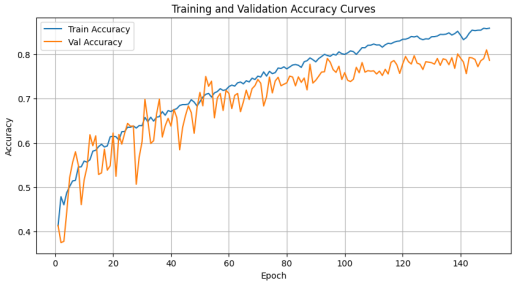

Figure 13: Final model accuracy

Remarkably, this model outshines all the trained hyper-parameter combinations. Furthermore, the F1 score is also computed to measure the precision and recall balance, which yields a result of 0.84. For our specific application, true positives are the number of pixels correctly identified as a certain feature, false positives are the pixels incorrectly identified as that feature, and false negatives are the pixels that are actually that feature but were not identified as such by the model. The F1 score is calculated for each feature/class of interest, and an overall F1 score is computed, through a weighted average based on the number of samples (pixels) for each class. A score of 0.84 is considered pretty good, validating model performance to accurately identify true positives and negatives.

In addition, the final model performance is also compared to the baseline model to indicate its improvements. The training accuracy surged from 0.4649 to 0.8792, while the testing accuracy rose from 0.4970 to 0.7999. In comparison with the initial baseline, this significant improvement highlights how much better the final model is, as it allows highly accurate satellite image segmentation to be achieved.

6.2 Qualitative Result

Using the original dataset, some sample outputs of the model are compared with the labeled images. Due to its close resemblance to the overall shapes and colors of the ground truth, we conclude that the model effectively captures and reconstructs the majority of image labels accurately.

Nonetheless, a closer examination reveals that the constructed image contains blurred boundaries, as opposed to the sharp separation seen in the labeled images. There are small areas where colors are out of sync with the ground truth and noise is scattered along the boundaries. This also highlights the limitations in the model's performance, as shown in Figure 12.

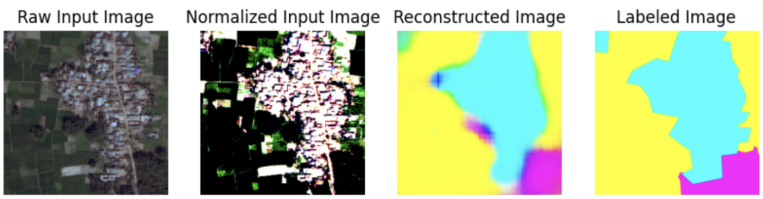

Figure 14: Sample Output From Final Model

Furthermore, the team has realized that the model is not sensitive to the 'unknown' class, which is labeled white. This is primarily due to the insufficiency of the collected training dataset. Additionally, the model fails to detect small features, such as the blue body of water in the top right corner of Figure 13. According to the quantitative results discussed previously, the 80% accuracy is primarily attributable to the large land cover areas in the reconstructed image. As a result of dataset limitations, the model is unable to handle small features and unfamiliar classes in the remaining 20%.

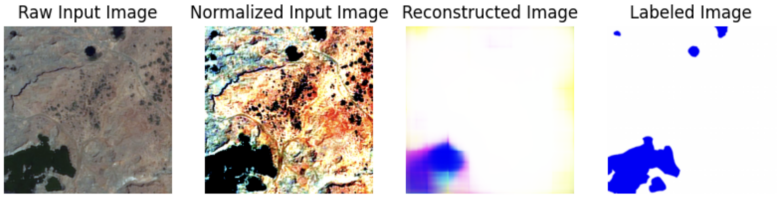

Figure 15: Sample output From Final Model

6.3 Model Evaluation

To proxy the model's performance in real-world application, we assessed the performance of the model on data it's never seen before. As such, we obtained satellite images from Google Maps, a popular web mapping platform used by people around the world, to simulate the model's performance in real applications. The classification is shown below:

Figure 16: Model Output for Testing Data

Figure 17: Model Output for Testing Data

As seen from the mask images, our model is able to identify the general landscapes of the satellite images. The mask shows a clear delineation of a particular area or feature based on the landscape classes the model has been trained on. For example, the agricultural lands have been identified and colour-coded to be yellow and urban areas have been indicated with the colour cyan. However, the reconstructed image contains blurred regions and fails to capture more nuanced details. These masks could be used to isolate or focus on a specific region for further study or action, thus fulfilling the purpose of our project.

7 DISCUSSION

Our model performed well based on our quantitative, qualitative results and testing demonstrations presented above. The outputted images from the final model is compared against that of the baseline model.

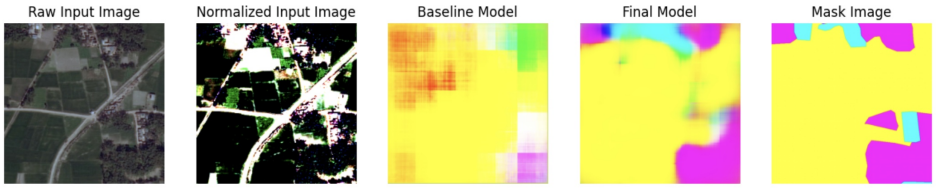

Figure 18: Output of Baseline and Final Model

Our model demonstrates the ability to distinguish and label different landscapes within an aerial photograph. For example, the delineation of different land types in the outputted image aligns with that on the label image. The colour segmentation matches with that of the label image and the border is much clearer than the baseline model.

Throughout this project, we found that many traditional strategies that should enhance our model accuracy theoretically, it didn't work well for us in reality. As such, we obtained surprising and interesting results after certain techniques have been implemented.

Apart from the baseline model, the performance of our other models are shown below:

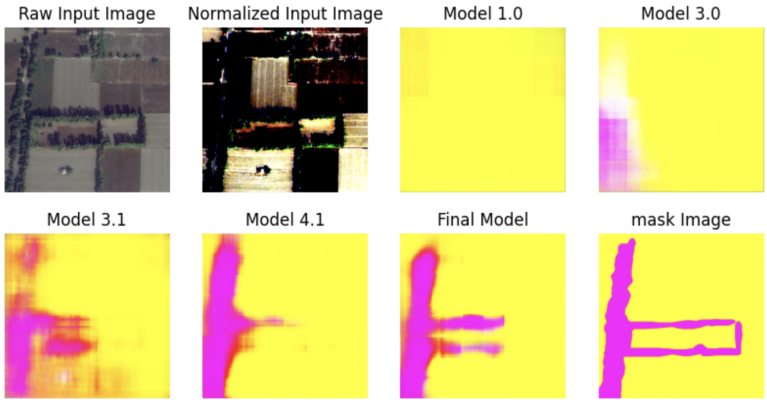

Figure 19: Output of Other Models and Final model

Although using transfer learning models such as ResNet would theoretically enhance the accuracy of reconstructed output. However, we found out that the transfer learning incorporates too many parameters which overwhelmed the computing resources and imposed challenges for the fine-tuning process. As such, the team customized their own architecture, taking inspiration from ResNet, to balance the tradeoff between performance and size of mode.

Instead of having the channel numbers gradually increase in the baseline model, the final model adjusted feature maps at each layer more irregularly, leaping between large and small channel numbers, which guaranteed the model's ability to extract key features at different levels of abstraction and mitigated the vanishing gradient problem. For pooling operations, max pooling is applied rather than average pooling, which is much more effective in extracting key information as it only extracts the most active feature in each region.

In the decoder layers, a big revolution is to replace transpose convolutional layers to Upsample function followed by a convolutional layer to reconstruct a mask image. The Upsample function reverses max pooling operation in encoder, specifically with nearest upsample reversing max pooling in encoder. Nearest upsampling increases the size of the feature map by duplicating the pixels in the input image, which preserves the sharp edges and fine details in an image. Then applying a convolutional layer with stride size of 1 filters the upsampled output, without changing the size of the output, to combine high-level and detailed features together in order to reconstruct masks close to ground truth.

8 Ethical Consideration

Privacy is one of the ethical issues, as the satellite images taken could track individual identities without consent. With the evolution and advancement of the camera and image-tracking instruments, individual figures are captured more clearly, raising concerns related to physical and locational privacies.

The model and training data have limitations as well. The training process for the model is based on an online dataset consisting of masked images that have been annotated and labelled by other individuals. As a result, the segmentation of the masked images is not 100% accurate due to the possibility of human bias. It is also important to note that the dataset obtained online, which is fairly small with only 803 images, only captured a small region in the United States.

Another ethical concern is related to the future applications of this model, as the land class segmentation output is biased and not completely accurate. The social-environmental decisions made could lead to errors. Since the model information will be uploaded to the Internet, privacy concerns arising from the repurposing and resharing of the information must also be noted.

9 Project Difficulty and Quality

It is a difficult project to perform since it is not a traditional cat-dog classification, but rather a multi-label and multi-class segmentation process. In this model, one of the most challenging steps is the computation of the loss function, since PyTorch does not provide a loss function that directly fulfills this task. The team calculates accuracy by comparing the output pixel to the correct pixel within the masked dataset. This approach is found after testing various methods, such as computing a probability prediction for the matching label using Binary Cross-Entropy Loss. Another difficulty is determining the model architecture. Considering that this project focuses on directly labelling information, CNN would not be a suitable model to use. Later, the team utilized other approaches, such as autoencoders, to directly capture the relationship between input and output data. Aside from this, the team also encountered challenges while training with Google Colab due to the limited RAM resources. As a result of the restricted usage, the team was only able to utilize a small portion of the dataset for the training process. Additional GPU resources were rented by the team to address this issue.

10 Conclusion

In conclusion, the autoencoder model constructed by the team achieves a notable accuracy of 0.8792 for training and 0.7999 for testing, representing a significant advancement over the baseline model. The model classifies and reconstructs large areas accurately, but exhibits limitations in handling small features and classes with insufficient datasets. Moving forward, the team plans to utilize a pre-trained model to enhance training accuracy and rates. Additionally, access to better GPU resources will also be crucial, as it is associated with several challenges that the team experienced. Furthermore, conducting extra and thorough background research and data analysis are also potential pathways to be pursued to enrich the model performance further.

Github Repository: https://github.com/yijie-04/APS360_Satellite_Image_Categorization

REFERENCES

[1] P. Malik, M. Chourasiya, R. Pandit, M. Bharaskar, and A. Medi, "Satellite image segmentation using neural networks: A comprehensive review," *International Journal of Enhanced Research in Educational Development*, vol. Vol. 11, pp. 2320–8708, 11 2023.

[2] M. Kadhim and M. Abed, *Convolutional Neural Network for Satellite Image Classification*, pp. 165–178. 01 2020.

[3] A. K. Rai, N. Mandal, A. Singh, and K. K. Singh, "Landsat 8 oli satellite image classification using convolutional neural network," *Procedia Computer Science*, vol. 167, pp. 987–993, 2020. International Conference on Computational Intelligence and Data Science.

[4] C. Henry, S. M. Azimi, and N. Merkle, "Road segmentation in sar satellite images with deep fully convolutional neural networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 12, pp. 1867–1871, 2018.

[5] R. G. do Nascimento and F. Viana, *Satellite Image Classification and Segmentation with Transfer Learning*.

[6] B. N. Pandey, A. K. shrivastava, and A. Rana, "A literature survey of optimization techniques for satellite image segmentation," in *2018 International Conference on Advanced Computation and Telecommunication (ICACAT)*, pp. 1–5, 2018.

[7] A. Ahmad and S. Quegan, "Comparative analysis of supervised and unsupervised classification on multispectral data," *Applied Mathematical Sciences*, vol. 7, pp. 3681 – 3694, 2013.

[8] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar, "Deepglobe 2018: A challenge to parse the earth through satellite images," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[9] R. G. do Nascimento and F. Viana, *Satellite Image Classification and Segmentation with Transfer Learning*.