Evaluating Preprocessing Alternatives for Ore-Waste Classification of Hyperspectral Images

by

Alexis Bruneau

A thesis submitted in conformity with the requirements
for the degree of Master of Engineering

Department of Mechanical and Industrial Engineering
University of Toronto

Evaluating Preprocessing Alternatives for Ore-Waste Classification of Hyperspectral Images

Alexis Bruneau
Master of Engineering
Department of Mechanical and Industrial Engineering
University of Toronto
2023

Abstract

Hyperspectral imaging holds potential for mineral mapping, offering unprecedented spectral resolution at the cost of overwhelming data volumes that strain computational capacities in image segmentation tasks. Addressing these complexities, this study undertakes a comprehensive evaluation of diverse preprocessing techniques and segmentation models. Unlike prior studies, which predominantly focused on standard, small-scale datasets such as Indian Pines or Pavia, this research distinguishes itself by applying established preprocessing methodologies to a significantly extensively larger dataset tailored for mineral applications. The study found a high degree of correlation among the spectral bands across the image, making Principal Component Analysis (PCA) an especially effective technique. By retaining five Principal Components, PCA managed to capture 99.8% of the dataset's variance, condensing it to less than 2% of its initial size. This optimization yielded a five-fold reduction in computational time and an increase of 4.9% in model accuracy in contrast to the baseline of unprocessed data. On the segmentation front, various segmentation models were explored, with U-Net++ topping the list in terms of accuracy, and FPN in computational speed. The study presents a list of combination of preprocessing and segmentation models, presenting their trade-offs with computational speed and accuracy when performing image segmentation between Ore and Waste rocks. This study lays a foundational groundwork for targeted future research in large-scale hyperspectral imaging within the realm of mineral exploration

Contents

0.1 Introduction

In mineral exploration, accurate differentiation between ore and waste materials is important to the economic success of mining operations. Using an efficient method to acquire the data and using a precise and automated decision-making system is crucial for the mining industry [4]. Among the tools available, hyperspectral remote sensing technology compares well with traditional remote sensing techniques for mineral mapping [21].

Hyperspectral images go beyond the capabilities of conventional RGB images. They offer a spectrum for every pixel, spanning numerous contiguous spectral bands. This richness in detail underscores their utility in mineral mapping [15]. Yet, with this depth of information comes the challenge of significantly larger file sizes when compared to their RGB counterparts. This escalation in data volume can quickly saturate available memory resources, particularly on systems with limited RAM or GPU capacities. Large data sizes will lengthen training duration's, hindering the development and optimization of models. Moreover, within these expansive hyperspectral images, certain data might prove redundant, which may not necessarily enhance a model's accuracy by keeping all the information [16]. It underscores the critical importance of effective prepossessing, to reduce computational overhead without compromising accuracy.

With the ascent of advanced computational techniques and the advances in deep learning capabilities, the field of hyperspectral image (HSI) classification has undergone transformative progress. Deep learning has established itself as a tool for HSI classification due to its capacity to process complex data. A challenge in HSI classification is the acquisition of these extensive labeled samples, given the time-consuming task of manual labeling. To surpass this, recent research pivots towards formulating deep learning models that can deliver accurate HSI classification even with a sparse set of labeled samples [13].

In line with these advancements, the optimization of preprocessing techniques emerges as a pivotal factor to further harness the full potential of deep learning for image segmentation. This paper delves into various preprocessing techniques applied to hyperspectral images, assessing their impacts on both computational time and model performance. Building on the foundation laid by [1], which used the Spectral Angle Mapper (SAM) and ENVI deep learning model for classification between Ore and waste, this research takes a different approach. Instead of employing SAM and ENVI for classification, SAM output is used as the reference label and centralizes preprocessing and model development within Python. This shift sidesteps the need for domain-specific expertise in the original tools. Utilizing the dataset from [1], the outcomes are compared to gauge the efficacy of different processing methods and segmentation models. The overarching aim is to highlight methods that reduce processing speed while conserving model's accuracy, thereby increasing the practical utility of hyperspectral imaging in mineral exploration and mining.

Several studies have delved into preprocessing techniques and band selection to boost model performance or trim computational duration. Often, these focus on popular, albeit smaller-scale, hyperspectral datasets like the Indian Pines and Pavian University, or they venture into disparate applications [20], [11]. In contrast, this work centers on the nuanced context of mineral rock segmentation, with a dataset considerably larger than these commonly referenced datasets.

0.2 Background and Methods

0.2.1 Overview of SAM & ENVI-Net5

Hyperspectral imaging generates data-rich images that capture information across hundreds of spectral bands. Such vast volumes of data may need sophisticated tools to get insightful interpretations. The Spectral Angle Mapper (SAM) and ENVI-Net5 are two pivotal tools in hyperspectral image analysis.

SAM provides a spectral classification method that assigns distinct classifications to each pixel in a hyperspectral image by comparing pixel spectra with a reference spectrum. Based on the principle that materials can be discerned by their spectral shape, SAM ensures each pixel's spectrum is appropriately classified.

Conversely, ENVI-Net5 is the deep learning module within the ENVI software suite, designed for pixel-wise classification of images. ENVI-Net5 employs a U-Net-inspired architecture, making comprehensive pixel-based classification feasible. Although the ENVI suite streamlines the implementation process by removing the programming aspect, its optimal application often demands substantive domain expertise.

0.2.2 Previous Work

Building upon previous research, this work adopts the dataset presented in [1]. This dataset consists of hyperspectral imagery of drill cores extracted from a silver ore deposit. As depicted in Figure 1, these cores, spanning approximately 16 meters in total, were initially cataloged by a site geologist. All images account for 64,944,400 pixels, structured as 10,560 by 6150, across 256 spectral bands. The first row as seen in Figure 1, represents cores labeled as waste, while the subsequent two rows denote those classified as Ore. In the originating study, each pixel was classified using tools like Spectral Angle Mapper (SAM), with subsequent classification performed using ENVI-Net 5 which attained an accuracy of 95%. The size of the raw spectral data amounts to 65 GB.

This methodology adapts and extends this work. It uses the data captured by the hyperspectral images and uses the classification of SAM as a reference label for the pixel classifications generated by SAM for model training.

0.2.3 Computational Infrastructure: Mist & Niagara

In this study, the pipeline harnessed the capabilities of two distinguished high-performance computing systems: the Niagara Supercomputer and the Mist GPU Cluster.

The Niagara Supercomputer, managed by SciNet at the University of Toronto, is tailored for large-scale computational tasks. With 80,640 cores and spanning 2,016 nodes, every node is equipped with 202 GB of RAM. Such abundant memory provisions position Niagara as an ideal platform for data-heavy processes, making it the chosen system for preprocessing the dataset.

On the other hand, the Mist GPU Cluster shines in GPU-centric tasks. Comprising 54 nodes, and each equipped with 4 NVIDIA V100 GPUs, Mist is purpose-built for operations that thrive on GPU acceleration. This makes it particularly apt for deep learning model training, an important component in this analysis.

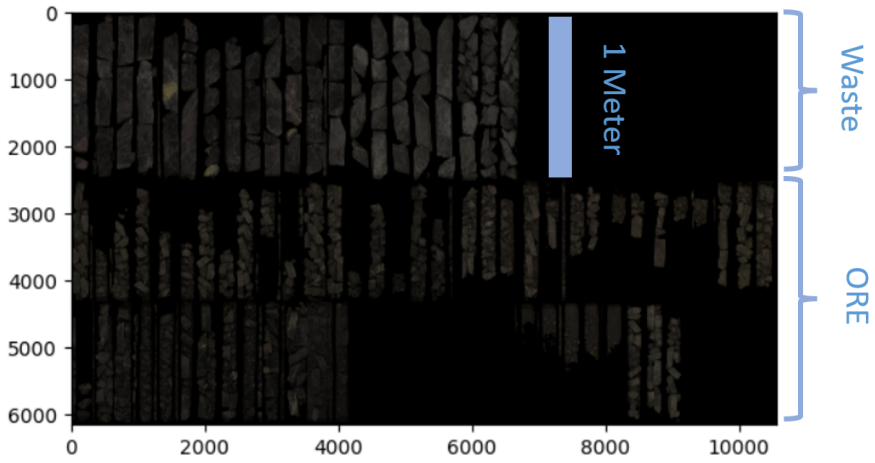


Figure 1: Full Mosaic Image of core sample

0.2.4 Pipeline setup

This analysis started with getting a subset of the Mosaic Image. This strategic selection was required due to memory constraints associated with MIST during extensive model training sessions on the full mosaic. The selected subset spans the rows between 1500 and 5500 and the columns between 0 and 4000 from Figure 1. This chosen subset served as the raw data for all experimental trials. Within this subset, a total of 16 million pixels were identified by SAM, consisting of 2,239,989 Waste, 4,038,938 Ore Pixels, and 9,721,073 Background Pixels, and used as reference labels.

Each test starts with Niagara Supercomputer being leveraged for the prepossessing of the data, with the processed outputs being stored for subsequent operations. Shifting to the training stage, Mist GPU Cluster takes precedence. Within the preprocessed data subset, non-overlapping tiles gets created from the processed data of dimensions 256 by 256. These tiles, treated as distinct images, are allocated into training, validation, and testing splits. Using these tiles a training model for image segmentation gets run, which will be discussed in further detail here 0.2.6. Model checkpointing was used to only save the model when the validation loss decreased. After 100 epochs, the model inference is executed on the test set to get performance metrics. This pipeline tested multiple preprocessing methods and modeling permutations, the detailed outcomes of which will be elaborated in the following section.

0.2.5 Preprocessing Methods

Hyperspectral imaging captures detailed spectral responses across numerous bands, bringing forth challenges of spectral redundancy leading to high computational costs. An approach in tackling these challenges is the preprocessing of hyperspectral data. Numerous studies have been investigated, from band selection techniques aiming to preserve spectral information while mitigating redundancy [20], to different studies that focused on incorporating prepossessing steps followed by deep learning models [11]. For instance, methods encompassing ranking, searching, clustering have been explored for band selection, while domain-specific preprocessing techniques, like EMSC and PCA, are frequently employed to enhance data integrity and model performance. In this study,

drawing insights from these prior works and a book that foccused on hyperspectral image processing [23], preprocessing techniques were selected and tailored to align with the dataset and objectives.

PCA (Principal Component Analysis): PCA is a prevalent technique in hyperspectral imaging for dimensionality reduction. Its objective is to maintain the salient features of the data while minimizing redundancy [23]. PCA achieves this by identifying the principal components (PCs) of the data, which are orthogonal vectors that encapsulate the maximum variance. The first principal component embodies the greatest variance, followed by the second, and so on. These PCs are formulated using linear combinations of the initial data features.

In the context of this study, PCA preprocessing function offers the adaptability to dictate the number of principal components to retain. For instance, setting the parameter to preserve only 2 PCs will reshape the transformed data's dimensions to (width, height, 2), with the final dimension indicating the two retained PCs. This configuration, when applied, will be represented as PCA2 in Table 2.

SNV (Standard Normal Variate): SNV is a common preprocessing technique in hyperspectral analysis, specifically designed to correct scatter effects in spectral data. Scatter effects can arise due to differences in sample surface roughness, particle size, or packing density. These factors can distort the underlying spectral signals of the mineral samples and reduce the consistency of measurements across different samples [3].

SNV works by centering and scaling the spectra. For each spectrum, it subtracts the mean of that spectrum and then divides it by its standard deviation. The results make in sort that the processed spectrum has a mean of zero and a standard deviation of one. Such normalization may help in emphasizing the variations arising from mineralogical differences while minimizing the effects of external scatter-related inconsistencies.

Eliminating Searching-Based (ESB) Band Selection:

The Eliminating Searching-Based (ESB) approach starts with the full set of bands. Through iterative processes, it removes bands until it gets a desired subset. This methodology is referred to in the following hyperspectral band selection survey [20].

The essence of ESB lies in its iterative approach. Starting with the complete set of bands, the method evaluates the correlation of each band with a target variable. With every cycle, the band that has the highest absolute correlation is retained. This procedure is repeated, with each iteration by removing one band at a time, until the specified number of bands is retained.

In this study, the employed function harnesses second-order correlation statistics, specifically the Pearson correlation coefficient, for band determination—resembling the methodology delineated in [9]. However, rather than combining multiple-order statistics, the algorithm only uses second-order correlation statistics, iteratively selecting the band with the highest absolute correlation to the reference labels.

MSC (Multiplicative Scatter Correction): MSC is a popular preprocessing technique often employed in hyperspectral imaging to correct for scatter effects and variations in the physical properties of samples. MSC is designed to enhance the consistency of data by adjusting for multiplicative (scatter-related) and additive (baseline) spectral variations [25].

The essence of MSC is its linear transformation approach, which reduces scattering effects and ensures that spectra primarily reflect actual mineralogical variations. In the applied function, the data undergoes a transformation using the StandardScaler. This adjustment ensures each feature in

the spectrum is centered and standardized, thereby minimizing the scatter-induced disparities.

RankDissimilarity: This method aims to select bands that exhibit the most distinct characteristics from one another. At its core, the function quantifies the distance between bands, identifying those that are most emblematic of the given hyperspectral data. This approach is similar to a method that is presented in [19] that uses exemplar component analysis.

This function uses the assessment of pairwise band distances, combined with an evaluation of their local density in the spectral space. The function assigns an Exemplar Score (ES) to each band, determined by both its local density and its distance from other, denser bands. Bands with large ES scores emerge as significant representatives and are selected for the subset.

K-mean band selection: This method uses k-means clustering to group hyperspectral image bands based on their statistical characteristics. This approach is derived from [2]. The function computes an assortment of statistics for each band: interquartile range, geometric mean, median absolute deviation, median, correlation coefficient (CC), covariance, and mode. By normalizing these statistics, the method creates a feature space where each band is represented. K-means clustering is then applied to partition the bands into subsets. From each cluster, the band with the highest interquartile range is selected as the representative.

SearchBasedUnsupervisedSkewness: This method selects hyperspectral bands that stand out based on their inherent statistical properties, particularly skewness and kurtosis, combined with the mutual information via the Jeffrie-Matusite (JM) distance. This method is based on the approach presented in [6] which emphasizes unsupervised ranking based on higher-order statistics.

This method first assesses the skewness and kurtosis for each band, removing those that surpass a defined threshold. Subsequently, bands are removed based on the JM distance. The 3 numbers shown after SearchSkweness in Table 2 refer to the threshold defined for all the mentioned criteria.

SparseRegressionBasedMethod: This method adapts the sparse regression-based for band selection presented in [10]. This method converts the band selection challenge into a sparse linear regression problem, using the capabilities of training samples paired with their respective class labels. Hence, the fewer coefficients that are actively showing show the importance of each band.

This method employs the Lasso regression method, a type of sparse regression. It treats each class label as a distinct entity using the One-vs-All strategy. Upon training the Lasso model for each class, it imposes a sparsity constraint producing coefficients reflecting the significance of each band concerning that class. Following the training, the bands are ranked based on the magnitude of their coefficients, and the ones that have high significance are kept.

0.2.6 Segmentation Methods

Hyperspectral images are dense with information. While the preprocessing methods discussed above work to refine and optimize the data, the segmentation models focus on the actual classification of pixels of the output of the preprocessed data. This study focuses on multiple segmentation architectures, with particular emphasis on U-Net due to its precedence in [1].

Segmentation Library:

For the deployment and experimentation within this study, the Segmentation_models library was used [12]. Considering the diverse shapes of processed images, modifications were made, especially to the models' first layer, ensuring it reflects the number of bands present in the preprocessed data.

Core Architectures: U-Net and UNet++

U-Net and UNet++, originally designed to address challenges in biomedical imaging, have emerged as foundational pillars in the field of image segmentation. Their remarkable capabilities have not only found recognition in biomedical imaging but also across diverse domains, extending to areas like hyperspectral imaging [18].

U-Net: The original U-Net architecture is designed around an encoder-decoder structure. The encoder captures the context of the input image, progressively downsampling the image through convolutional and pooling layers. The decoder, conversely, semantically projects the contextual features learned by the encoder to pixel space, enabling precise localization. Additionally, skip connections between the encoder and decoder layers ensure that fine-grained details lost during downsampling are reintroduced, enhancing the segmentation's accuracy.

UNet++: A nuanced iteration of the U-Net, UNet++ introduces a series of nested, skip pathways, which aim to address the semantic gap between the feature maps in the encoder and decoder. These nested connections promote feature reuse and enable the model to capture more detailed and varied representations at different depths, thereby refining the segmentation outputs. The architectural visualizations for both models can be seen in Figure 2.

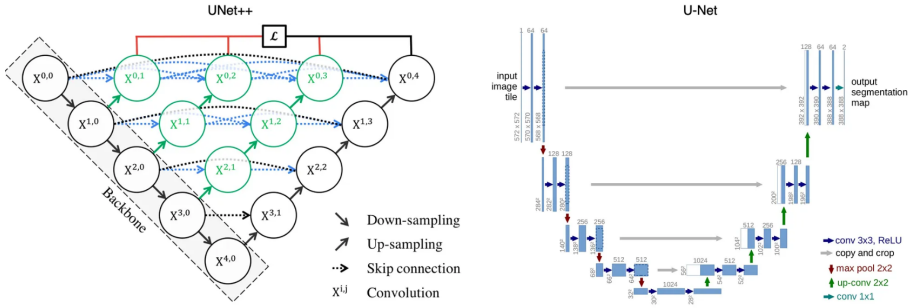


Figure 2: U-net & UNet++ Architecture [14]

Extended Architectures: For a well-rounded assessment of segmentation potentialities, an array of other architectures were integrated after determining potential processing techniques. Each offers a unique approach and strengths in image segmentation:

MAnet (Multi-Attention-Network): Crafted for fine-resolution remote sensing imagery, MAnet employs multiple attention mechanisms, accentuating the nuanced details that are paramount in such images.

Linknet: Designed with a swift and resource-efficient segmentation in mind, Linknet melds the encoder-decoder paradigm with additional shortcuts. These ensure that essential features persist through the network's layers.

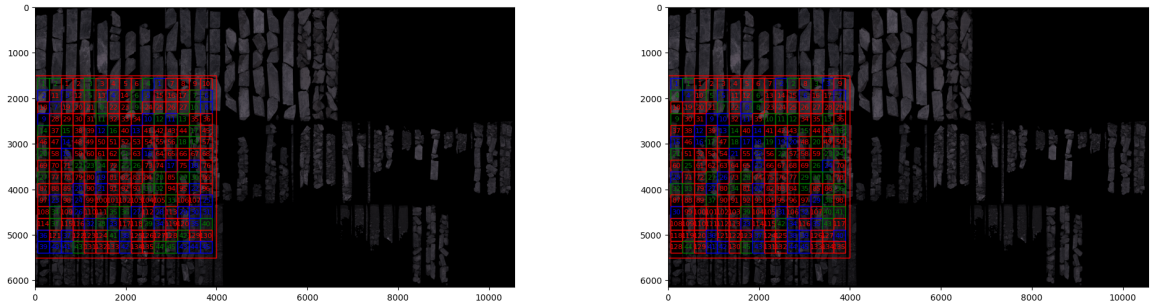
FPN (Feature Pyramid Network): FPN fuses high-level semantics and low-level features using a pyramid of skip connections. This configuration bolsters object detection across diverse scales.

PSPNet (Pyramid Scene Parsing Network): With its pyramid parsing, PSPNet enhances the receptive field, grasping comprehensive context information which yields a more meticulous segmentation output.

PAN (Pyramid Attention Network): Integrating both ascendant and descendant attention mechanisms, PAN adeptly seizes multi-scale features, enhancing the overall segmentation precision.

Model Modification

Along with the primary structural modifications, other modifications were made to factor in the variance of each model. Central to this was the incorporation of a unique random seed in data partitioning, facilitating consistent training, validation, and testing set splits across multiple runs for the same methodology using a job array. The data distribution for the tiles was segmented as 60% for training, 20% for validation, and 20% for testing, as illustrated in Figure 3. In this representation, red tiles indicate training, green for validation, and blue for testing. This particular setup was essential in observing the variance in results when applying the same preprocessing approach with different random splits. In tandem, the capabilities of PyTorch's DataLoader constructs were leveraged. These ensured the data was not only batched efficiently but also shuffled, guaranteeing a broad and varied learning spectrum throughout the training epochs.



(a) Random Split Example 1

(b) Random Split Example 1

Figure 3: Random Train/Val/Test Split Examples

For the training regimen, the Adam optimizer directed the model's parameter updates, while the DiceLoss multiclass served as the criterion for gauging model performance.

The computational environment posed offline constraints. To address this, pre-trained weights were loaded for the resnet34 backbone from an offline repository, thus eliminating online dependencies. This method ensured initialization with patterns from prior datasets and seamless deployment in settings without internet access.

Throughout the training phase, regular performance checks on the validation set helped in determining the best model iteration, with checkpoints saved only for models that exhibited improved validation loss. This rigorous checkpointing ensured that the final model represented the most optimal state across all epochs.

0.3 Results

Building upon the methodologies detailed in the previous section, the results start by analyzing the segmentation model's performance without applying any preprocessing techniques to the raw data. Primarily, the U-Net model was evaluated using this raw data to discern the potential influence of preprocessing on model accuracy. The outcomes of this evaluation, executed on both the Mist and Niagara platforms for ten separate jobs, are presented in Table 1 and serve as benchmarks for subsequent comparisons regarding the model's accuracy.

Within the table 1, several metrics elucidate the model's performance. "Avg Acc" delineates the model's average accuracy as a percentage, culminating in results from the ten jobs. For each distinct jobs, a varying selections of tiles were selected for training, validation, and testing as shown in Figure 3. "Min Acc" and "Max Acc" capture the spectrum of observed accuracies, spotlighting the minimum and maximum accuracies respectively. The median accuracy is captured under "Med Acc". Providing a perspective on the results' dispersion, the "RSD" metric conveys the relative standard deviation of accuracy across all the jobs. "Inf Time" represents the time, measured in seconds, that the model necessitates for inference on the test tiles.

Method	Avg Acc	Min Acc	Max Acc	Med Acc	RSD	Avg Train Time	Inf Time
U-Net Niagara	87.4	74.4	93.7	89.1	5.9	1366	2.044
U-Net Mist	87.2	80.6	91.8	88.5	4.44	897	2.073

Table 1: Model Performance on Raw Data

Having established the benchmark performance on raw data, the natural progression is to investigate how preprocessing can refine this raw data, potentially enhancing the segmentation model's performance. Preprocessing not only affects the model's accuracy but also impacts the computational resources, both in terms of processing time and storage requirements. The subsequent section sheds light on the time and storage implications of various preprocessing techniques and their effectiveness in preparing the data for segmentation.

0.3.1 Preprocessing Results

Table 2 extensively examines the preprocessing techniques tested. The "Time (s)" column details the time taken for preprocessing the raw numpy images in seconds. Additionally, the column Input Size\Output Size (%) offers insight into how the preprocessing techniques alter the dataset's size relative to the original raw data. Finally, the column "Output (GB)" represents the size in GB of the processed data.

Table 2: Preprocessing Summary

Methods	Time (s)	Input Size\Output Size (%)	Output Size (GB)
PCA1	**63**	**0.39**	**0.06**
PCA2	64	0.78	0.12
PCA3	70	1.17	0.18
PCA4	71	1.56	0.24
PCA5	81	1.95	0.30
PCA10	100	3.91	0.60
PCA20	140	7.81	1.2
PCA25	181	**9.77**	**1.5**
MSC_PCA1	120	**0.39**	**0.06**
MSC_PCA2	123	0.78	0.12
MSC_PCA3	124	1.17	0.18
MSC_PCA4	154	1.56	0.24
MSC_PCA5	134	1.95	0.30
SNV_PCA1	72	**0.39**	**0.06**
SNV_PCA2	71	0.78	0.12
SNV_PCA3	76	1.17	0.18
SNV_PCA4	79	1.56	0.24
SNV_PCA5	79	1.95	0.30
EMSC1_PCA5	25512	1.95	0.30
EMSC2_PCA5	43327	1.95	0.30
ESB1	115	**0.39**	**0.06**
ESB2	219	0.78	0.12
ESB3	326	1.17	0.18
ESB4	410	1.56	0.24
ESB5	501	1.95	0.30
Kmean5	**67150**	1.95	0.30
RankDissimilarity5	8871	1.05	0.30
SparseRegression1	811	**0.39**	**0.06**
SparseRegression2	811	0.78	0.12
SparseRegression3	820	1.17	0.18
SparseRegression4	824	1.56	0.24
SearchSkweness(1,1,1)	236	0.78	0.12
SearchSkweness(2,2,2)	188	0.78	0.12
SearchSkweness(3,3,3)	195	0.78	0.12
SearchSkweness(4,4,4)	235	**0.39**	**0.06**
SearchSkweness(5,5,5)	187	**0.39**	**0.06**

Variance Explained by PCA: Following the exploration of preprocessing techniques as captured in Table 2, it's evident that PCA frequently operated in tandem with many of these methodologies.

Given this frequent pairing, determining the optimal number of Principal Components (PCs) to

retain for tests became an important consideration.

In the data visualized in Figure 4, it can be seen that a significant portion of the variance can be captured by just a few principal components, suggesting a high level of redundancy in the dataset. This was seen in a similar study presented in [7], where a hyperspectral image containing 204 bands was compressed to represent 99% of its variance using 3 PCs. In this study, a single PC explains 98.8% of the total variance. With just 2 PCs, this coverage exceeds 99%. Only 8 PCs are required to represent 99.9% of the variance. Given these findings, tests were designed to focus on 1 to 5 PCs. Yet, in scenarios where only PCA preprocessing was applied, further tests using 10, 20, and 25 PCs were conducted to assess if slightly broader information might enhance the model's efficacy.

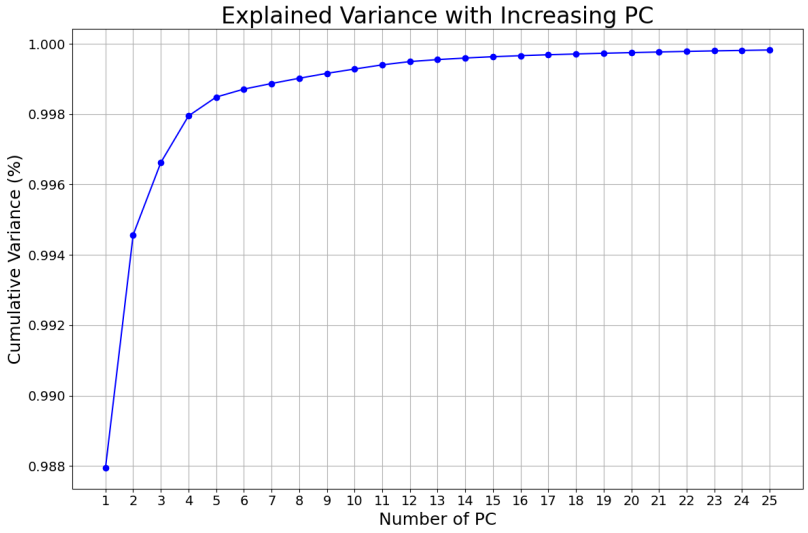
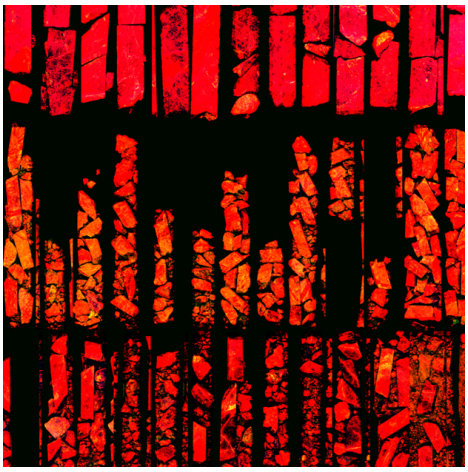


Figure 4: Cumulative Variance by PC used

In light of the substantial variance explained by a few key principal components, it's important to explore the visual content captured by these components. Figure 5 showcases the first Principal Component as a monochromatic image, revealing its remarkable resemblance to the original image. This visual similarity highlights the significant explanatory power held by the first PC. Moreover, Figure 5 combines the first three principal components into an RGB representation. A subtle shift in color between the first and second rows of the core image, which may visually indicate the differences between the two classes.

(a) 1 PC Monochromatic View



(b) 3 PCs RGB View

Figure 5: Visual Representation of Principal Components in Original Dataset

0.3.2 Performance on Processed Data

After applying various preprocessing techniques to the data, the immediate goal became evaluating their influence on the U-Net model's performance. Each preprocessing method was paired with U-Net and assessed over 10 jobs. The results of these evaluations are captured in Table 3. The same performance metrics as those in Table 1 were maintained for clarity and continuity in analysis. Upon identifying preprocessing techniques that optimally balance accuracy with processing time, subsequent sections of this report further investigate these "winning methods" by employing a broader range of segmentation models.

Table 3: Preprocessing Methods with U-Net performance summary

Methods	Avg Acc	Min Acc	Max Acc	Med Acc	RSD	Avg Train Time	Inf Time
PCA1	78.9	66.7	88.4	80.5	9.3	79	0.0648
PCA2	87.9	83.1	92.5	88.3	3.4	82	0.0686
PCA3	90.9	87.6	94.3	91.2	2.4	84	0.0709
PCA4	91.8	88.9	95.1	91.9	2.1	85	0.0723
PCA5	92.1	89.4	95.1	92	2.1	88	0.0747
PCA10	92.3	**89.5**	95.3	92.5	**2.0**	95	0.0965
PCA20	92.3	88.3	**95.8**	92.4	2.4	108	0.1229
PCA25	**92.5**	88.9	95.6	**92.9**	2.3	**119**	0.153
MSC_PCA1	78.1	70.1	85.9	76.7	7.9	81	0.0651
MSC_PCA2	87.9	82.7	92.1	88.3	3.5	82	0.0705
MSC_PCA3	90.1	83.5	94.1	91.2	3.4	85	0.0729
MSC_PCA4	91.6	88.2	94.5	91.6	2.3	86	0.0738
MSC_PCA5	91.8	88.9	95.4	91.2	2.4	87	0.0753
SNV_PCA1	82.3	70.6	87.8	84.3	6.1	80	0.0648
SNV_PCA2	87.9	82.5	91.8	88.6	3.5	83	0.0673
SNV_PCA3	90.8	87.7	94.1	90.1	2.4	85	0.0719
SNV_PCA4	91.5	87.6	94.8	92	2.4	85	0.0728
SNV_PCA5	92.1	88.4	95.3	92.5	2.3	87	0.0743
EMSC1_PCA5	89.8	86.1	93.8	90.1	2.7	85	0.0747
EMSC_PCA5	85.7	79.0	90.1	86.9	4.4	86	0.0747
ESB1	79.3	61.9	87.4	80.2	9.4	79	0.0653
ESB2	79.7	67.7	85.8	82.3	7.6	82	0.069
ESB3	76.5	65.6	90.4	74.4	10.6	83	0.0716
ESB4	78.7	72.9	83.4	78.9	3.9	83	0.0731
ESB5	74.7	62	86.8	74.7	**10.9**	83	0.0749
Kmean5	86.0	81.4	90.3	85.8	3.8	83	0.0752
RankDissimilarity5	83.5	62.9	91	86.4	9.6	84	0.0752
SparseRegression1	86.5	82	90.7	87.5	3.7	84	0.0714
SparseRegression2	87.0	80.5	88.6	87.2	3.1	85	0.0766
SparseRegression3	88.9	83.1	92.8	90.1	3.5	90	0.0881
SparseRegression4	88.9	82.7	93.7	89.2	3.6	93	0.0952
SearchSkweness1_1_1	83.4	76.6	88.3	85	5.3	80	0.0689
SearchSkweness2_2_2	84.2	76.5	88.9	86.3	5.1	80	0.0679
SearchSkweness3_3_3	83.8	77.1	87.7	85.7	5.0	80	0.0679
SearchSkweness4_4_4	**70.2**	**61.1**	**82.4**	**67.9**	10.4	**78**	0.0659
SearchSkweness5_5_5	72.9	61.9	85.4	71.8	9.6	79	0.0669

0.3.3 Methodological Consideration

Data Leakage in Preprocessing: In this study, preprocessing was uniformly applied to the full data subset before partitioning it into training, validation, and testing sets. This methodology

simplifies file management, a notable benefit given that different computational resources—Niagara for preprocessing and Mist for model training—were utilized. However, it also opens the door to potential data leakage concerns. To analyze the impact of data leakage, a focused experiment was carried out using the U-Net model with PCA5. In this setup, principal component vectors were calculated exclusively from training tiles, and subsequently applied to the validation and testing sets. The exercise yielded a 0.3% decline in average accuracy against the same methodology when data leakage was not considered, as shown in Table 4. Given the minor impact, the study proceeded with the original preprocessing strategy, assuming minimal effects on performance metrics due to data leakage.

Table 4: Performance Metrics Under Data Leakage Consideration

Methods	Avg Acc	Min Acc	Max Acc	Med Acc	RSD
PCA5	91.8	87.5	95.4	92.4	2.5

Variance Investigation: In the evaluation process, understanding the impact of preprocessing techniques is vital, accompanied by the need to ensure the reliability of the model's performance metrics due to the unpredictability introduced by random tile selections. The core intent of this investigation was to identify a job count where the model demonstrates consistent and representative performance. The focus was on determining the point at which adding more jobs results in minimal reductions in variance. Beyond this threshold, increasing the number of jobs would not significantly enhance the validity of the findings but would instead consume more computational resources. To determine this, tests where preprocessing was completed within an hour and expanded the evaluation to include performance over 5, 10, and 20 jobs. As seen in Table 3, all methods underwent testing for 5, 10, and 20 jobs, with the exceptions of combinations involving EMSC, Kmean, or RankDissimilarity. Based on the findings presented in Figure 6, progressing with 10 jobs for subsequent evaluations is justified.

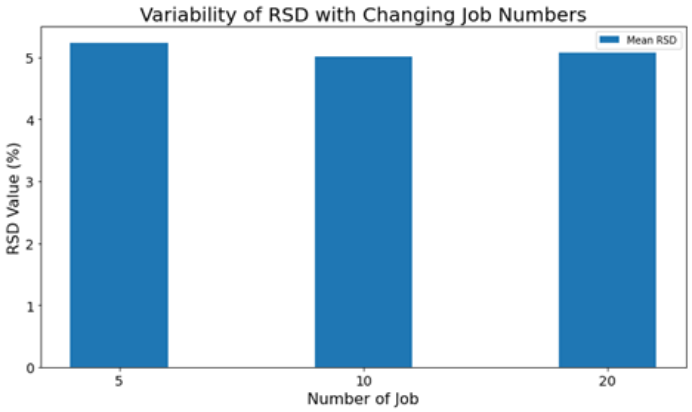


Figure 6: Mean RSD for different number of jobs

Optimal Preprocessing Methods: A Time-Accuracy Tradeoff

To provide a more comprehensive understanding of these findings, a scatter plot is employed in Figure 7. This plot depicts the average accuracy in % across 10 jobs on the y-axis and the combined time for preprocessing and training on the x-axis in seconds. For clarity, not all preprocessing

methods are represented in the scatter plot. Specifically, any method that took longer than 260 seconds or had an average accuracy below 75% has been omitted. Numbers displayed in the center of each scatter point represent the number of channels in the processed data. Points situated in the top left of the plot represent methods with the highest accuracy and lowest preprocessing time, effectively highlighting PCA and the combination of SNV & PCA as the most balanced and efficient methods.

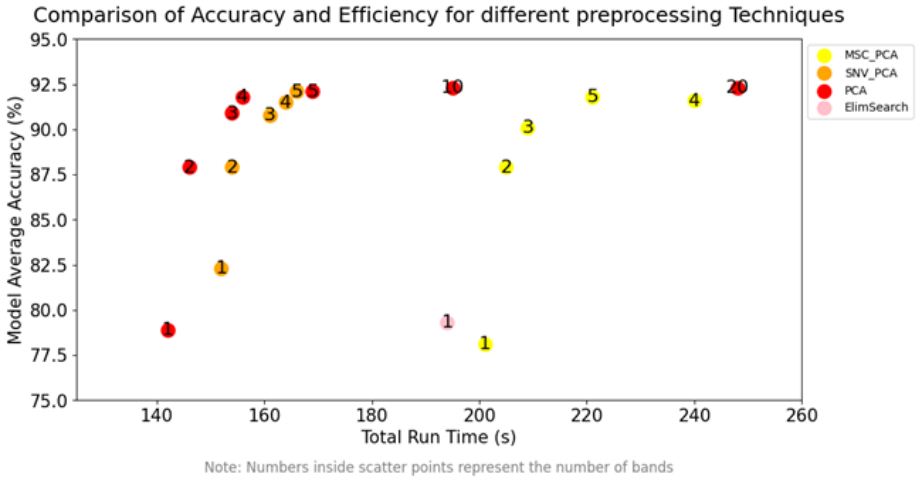


Figure 7: Compraison of Accuracy and Efficiency for different preprocessing method

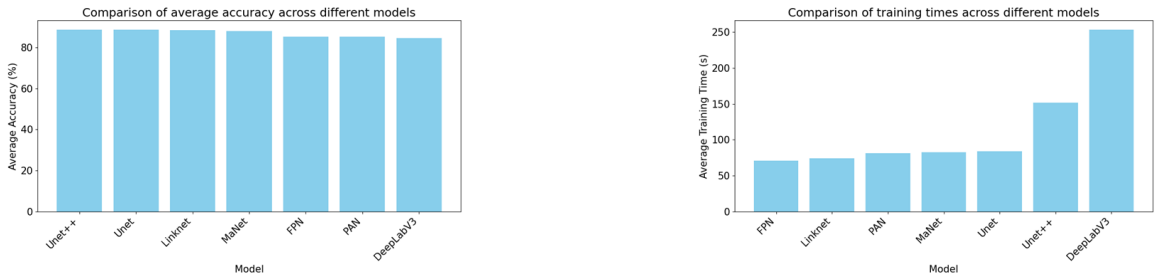
0.3.4 Expansion to Various Models

Building on the previous sections, this investigation shifts its focus to a broader exploration of segmentation models. With the insights on optimal preprocessing techniques and job counts, this study aims to further leverage and assess these "winning methods" across a more varied range of segmentation models.

From Figure 7, two preprocessing methods particularly stand out: PCA and the combination of SNV followed by PCA. Their appeal was rooted in the balance they struck between preprocessing time and model performance, pointing to their potential applicability across diverse model architectures.

With this understanding in hand, the research direction involved pairing these preprocessing methods with an array of segmentation models, thereby broadening the scope beyond the confines of the foundational U-Net model. The models tested encompassed Unet++, FPN, MaNet, Linknet, PAN, and DeepLabV3. The purpose was to verify if any methods resulted in better results compared to the test made with U-Net presented in Table 3.

Figure 8 illustrates the following: only one segmentation model obtained a higher accuracy the U-Net model. Specifically, Unet++ demonstrated the highest average accuracy with a result of 88.74% while the lowest result obtained was from DeepLabV3 with an average accuracy of 84.56 %. While the choice of segmentation model did yield variances in accuracy, the magnitude of these differences remained relatively narrow with an accuracy difference of 4.16% between the best and worst segmentation model.



(a) Comparison of average accuracy

(b) Comparison of training time

Figure 8: Segmentation Models Results

Concerning the training durations, four of the segmentation models proved to be more time-efficient compared to U-Net which had an average training time of 83.8 seconds. Specifically, the faster models were FPN, Linknet, MaNet, and PAN, with the following respective training time: 70.7, 73.9, 82.5, and 81.4 seconds. On the other hand, both Unet++ and DeepLabV3 required longer training durations. Unet++ average training time was 151.5 seconds whereas DeepLabV3 had the longest training time with 253.5 seconds.

Figure 9 illustrates the trade-offs between model accuracy and training time. Points positioned closer to the top-left corner represent models that achieve a desirable balance of high accuracy and low training time.

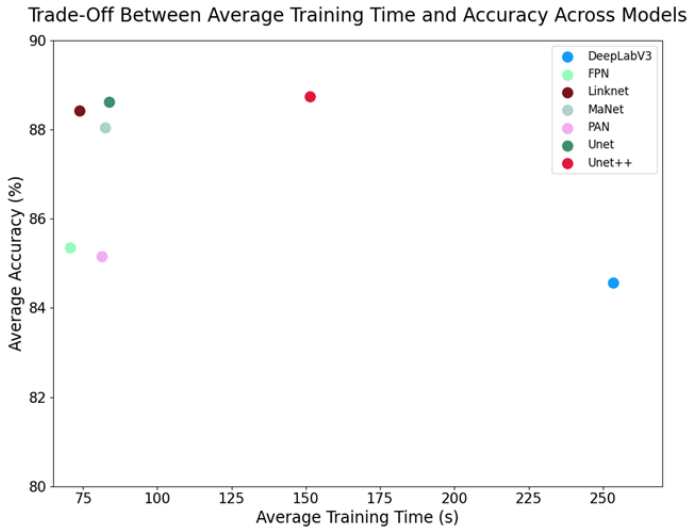


Figure 9: Comparison of Accuracy and Efficiency for different Segmentation Models

Insights from Training & Validation: Epoch Choice Explored:

The selection of the number of training epochs is a pivotal factor affecting the performance of deep learning models. Overtraining with excessive epochs can result in overfitting and wasted computational resources. Conversely, an insufficient number of epochs may lead to undertraining, where the model fails to capture meaningful patterns from the data.

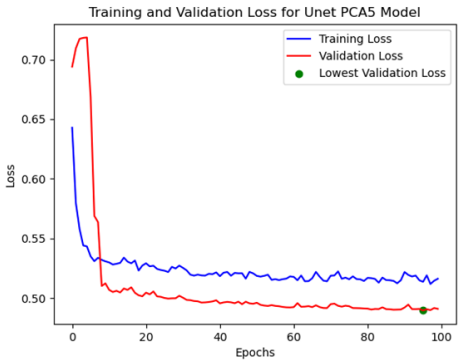
In this study, the choice of 100 epochs was informed by empirical observations and carefully reasoned analysis, rather than being arbitrary. Model checkpointing was employed based on validation

loss to sidestep the issues of both overfitting and undertraining.

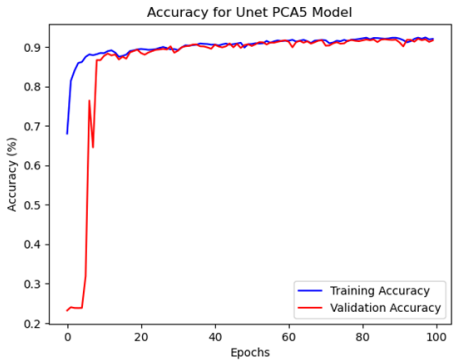
Preliminary experiments across varying epoch counts were conducted to gauge the performance at different checkpoints. This was done with the objective of providing sufficient training time for the models to learn relevant features without unnecessarily consuming computational resources. This analysis revealed that most models reached their last saved state before the 100-epoch limit, creating a reasonable buffer.

For instance, the lowest validation loss for LinkNet_PCA5 occurred at the 92nd epoch, and for Unet_PCA5, it was at the 87th epoch. These observations substantiate the choice of a 100-epoch threshold. Even if the models had continued to train beyond 100 epochs, the marginal decline in validation loss and minimal improvement in accuracy suggest the gains would be insignificant as shown in Figure 10.

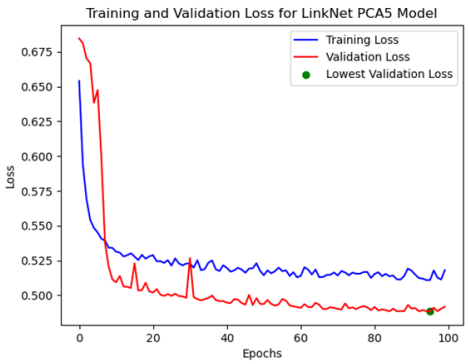
While extending the training beyond 100 epochs could potentially result in marginal gains in accuracy and further reductions in validation loss, the observations suggest these improvements would likely be minimal. Given this consistent pattern, it was reasoned that extending the epochs further would not significantly alter the comparative performance between the various methods explored in this study. Consequently, the benchmark of 100 epochs was chosen as it offered an optimal balance between computational efficiency and model performance.



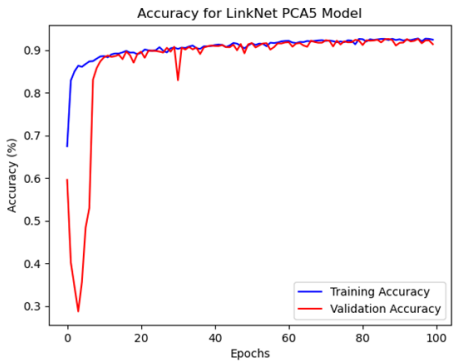
(a) Unet PCA5 Loss Plot



(b) Unet PCA5 Accuracy Plot



(c) LinkNet PCA5 Loss Plot



(d) LinkNet PCA5 Accuracy Plot

Figure 10: Optimal Epoch Choice: Training & Loss Plot for Unet & LinkNet PCA5 Models

0.3.5 Model Performance: A Qualitative Perspective

After the exploration of quantitative performance metrics for various segmentation models, attention shifts to a qualitative evaluation. Figure 11 illustrates SAM's output, which serves as the reference label during the training phase of the model. In this image, different colors signify different classes: black pixels represent the background, blue pixels are designated as Ore and green pixels correspond to Waste.

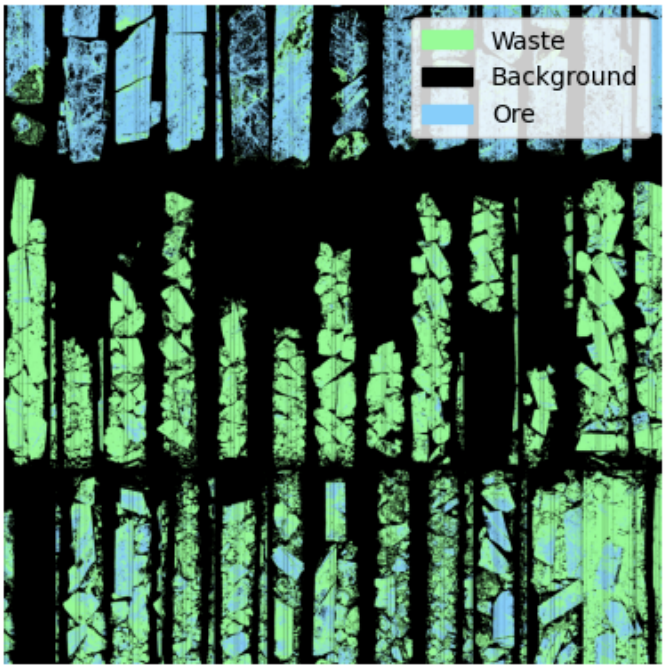


Figure 11: Reference Label Generated by SSAM for the Data Subset

The following predictions show the results when using U-Net & PCA5 as it demonstrated a good balance between accuracy and computational efficiency. Figure 12 illustrates the model's segmentation performance on three test files, each characterized by a unique material composition: one predominantly Ore, another Waste, and a third containing a blend of both. The images reveal a substantial alignment between the model's predictions and the SAM-based reference labels, reiterating the model's previously quantified high accuracy.

It is worth noting that while SAM's output generally aligns well with the original hyperspectral image, it can be seen that minor deviation from the original image. Interestingly, these small deviations or "noises" in the SAM output do not seem to adversely impact the model's predictions, suggesting inherent robustness in the Unet_PCA5 model against such discrepancies.

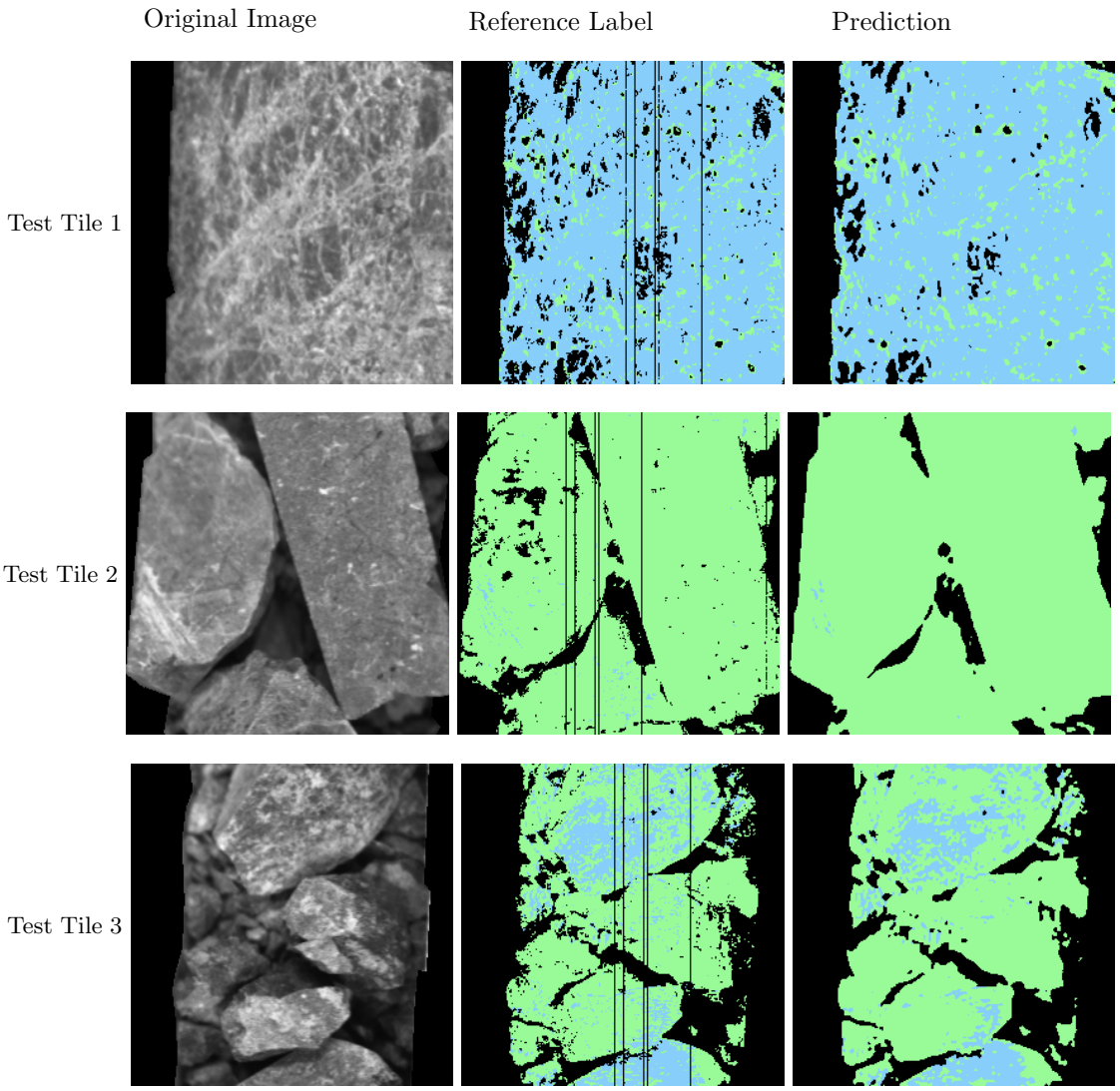


Figure 12: Segmentation Model Results

0.4 Discussion

0.4.1 A Close Look at Preprocessing Results

Transitioning to the Discussion section, it's crucial to take a detailed look at how different preprocessing techniques performed. These methods significantly impact both the overall accuracy of the model and the time needed for training.

Initial results show that using U-Net with raw data yielded an average accuracy of 87.4% and a total training time of 897 seconds. However, the fastest preprocessing method completed the same task in just 142 seconds, reducing the total run time by more than a factor of 6. However, this speed advantage did result in a decrease in average accuracy, which stood at 78.9%.

Contrastingly, the PCA25 preprocessing method emerged as the most accurate, registering an average accuracy of 92.6% and cutting the total training time to 300 seconds. This not only trims down the time needed almost by a factor of 3 but also boosts the model's accuracy by 5.2%. It should be noted that the gains in accuracy taper off beyond PCA3. Between PCA3 and PCA25, the average accuracy increased by only 1.6%, and an even smaller increase of 0.4% occurred between PCA5 and PCA25. These minor gains in accuracy are consistent with the observations from Figure 4, which indicates that the early principal components capture most of the dataset's variance.

In terms of computational efficiency, the PCA algorithm shines, particularly when Eigendecomposition or singular value decomposition (SVD) methods are employed. This is inherent in the scikit-learn implementation used for this study. Additionally, only a nominal increase in processing time was noted when more principal components were retained. This can be attributed to the algorithmic design of PCA: once the principal components are calculated, adding additional bands simply involves accessing pre-computed vectors. Consequently, the computational time only marginally increases with the inclusion of more bands.

Following closely behind PCA is the SNV_PCA method, which showed only a slight increase in preprocessing time. The method involves performing a computationally inexpensive operation—subtracting the mean and dividing by the standard deviation across each spectral band—prior to applying PCA. This accounts for the modest increment in preprocessing time observed in the SNV_PCA approach.

The SearchSkweness function performs more intricate statistical calculations like skewness and kurtosis across multiple bands. Secondly, it employs conditional selections based on these computed moments, adding another layer of computational effort. Thirdly, the Jeffries-Matusita (JM) distance calculation between bands involves matrix inversion and determinant calculations which may explain why it took longer for its processing time.

The ESB has another layer of computational complexity. One potential reason for its long runtime may be its iterative nature. It calculates correlation coefficients for each remaining band with the target variable in each iteration until a certain number of bands are selected. Since each iteration involves a non-trivial computational cost due to correlation calculation, and bands are eliminated one at a time, the cumulative time requirement grows. Furthermore, eliminating the best-performing band from the band pool at each step necessitates updating the correlation coefficients, making the algorithm inherently iterative and computationally intensive.

The Sparse Regression-Based method may have a larger computational time attributed to the

complexity of the Lasso regression it utilizes. The algorithm runs Lasso regression independently for each unique class in the training labels. This not only entails fitting the Lasso model multiple times but also involves calculating the Lasso coefficients for every spectral band per class.

The RankDissimilarity method showed higher computational time. This may be attributed due to its nested looping through pixels and pairwise band comparisons. This not only entails calculating Euclidean distances for each pixel-band pair but also further operations to compute local densities and exemplar scores. These numerous iterations scale with the cube of the number of bands and linearly with the number of pixels, making the method significantly time-intensive

When compared to other methods of hyperspectral data preprocessing, EMSC, and K-means Band Selection—had by far the longest preprocessing times.

The EMSC method involved several computationally intensive steps, including a nested loop through each pixel and each spectral band, polynomial curve fitting, and multiple arithmetic operations. Each of these steps can be costly in terms of computational time, especially when dealing with high-dimensional hyperspectral data with a large number of pixels and bands. The use of the curve_fit function to fit a polynomial trend to each spectrum further adds to the complexity and runtime, as it inherently involves iterative optimization techniques to minimize the residual sum of squares.

On the other hand, K-means Band Selection method also shows high computational complexity. Primarily, it involves computing a series of statistical properties for each band, including Interquartile Range (IQR), correlation coefficients, covariances, and geometric means, among others. These statistical calculations themselves are computationally expensive, and when coupled with K-means clustering, the time complexity escalates even further. K-means clustering involves iterative assignment and reassignment of cluster centroids, and in this case is performed on multiple statistical attributes for each band, adding another layer of complexity. Consequently, both methods exhibit a longer runtime compared to the other band selection algorithms.

0.4.2 A Close Look at the Segmentation Results

This section transitions from the evaluation of preprocessing methods to explore the performance of various segmentation models, which were applied subsequent to the most effective preprocessing techniques.

As depicted in Figure 9,FPN, PAN, MaNet, LinkNet, and U-Net offer similar training times and performance metrics. FPN, LinkNet, U-Net, and UNet++ emerge as ideal choices for the Pareto Front, balancing average accuracy and computational efficiency. Notably, FPN recorded the fastest training time, whereas UNet++ had the highest average accuracy.

Contrasting these findings with a previous study, which evaluated U-Net, LinkNet, and FPN on satellite image segmentation, FPN's superior training speed in the current study is interesting. The previous study indicated almost identical training times for U-Net and LinkNet, with FPN lagging slightly [8]. The divergence may be attributed to the nature of the processed, larger-scale images in the present study.

U-Net and its variant UNet++ are often go-to choices for semantic segmentation, making UNet++'s high accuracy less surprising. UNet++ enhances U-Net's performance by incorporating a nested skip pathway, leading to increased model complexity and, consequently, slightly higher accuracy at the expense of longer training time [22].

DeepLabV3 is noteworthy for its prolonged training time and reduced accuracy in the context of this specific study. One potential explanation for this outcome is the model's utilization of atrous (dilated) convolutions, engineered to enlarge the receptive field and thereby capture a greater breadth of contextual information within images. While such an approach has proven advantageous for intricate segmentation tasks requiring detailed contextual understanding, it also imposes substantial computational and memory burdens. These overheads could be the contributing factors to the observed extended training durations and compromised accuracy on the dataset at hand. [24].

0.4.3 Limitations & Future Directions

The Case for Individual Rock Analysis: One valuable avenue for future research involves analyzing images of individual rocks, each paired with corresponding SAM labels instead of using a Mosaic Image. Adopting this approach could mitigate the risk of models learning spatial features specific to the experimental arrangement, such as the placement of Waste rocks in the first row and Ore rocks in the two bottom rows of the mosaic. This is significant, as models trained on individual rocks could offer more realistic and generalizable performance in real-world mineral operations. Additionally, this method offers multiple advantages including the ease of parallelization. Furthermore, this methodology enables a more concentrated effort on rock classification, reducing the computational overhead associated with analyzing unnecessary background pixels.

Exploring Various Backbone Architectures: Building on the topic of model architecture, another direction for future research lies in the exploration of different backbone architectures. As demonstrated in [8], the choice of backbone can significantly influence model performance. While this study utilized ResNet-34 as the backbone for all segmentation models, investigating alternative architectures could provide valuable insights and potentially enhance the robustness and efficacy of segmentation models. This diversification of backbones could add an extra layer of nuance to the findings, opening up opportunities for performance optimization in subsequent research.

Metrics & Loss Functions:

Although this study employed Accuracy as the primary metric and Dice Loss as the loss function, it's worth noting that the dataset's class distribution was not perfectly balanced, consisting of 4,038,938 Ore Pixels, 9,721,073 Background Pixels, and 2,239,989 Waste Pixels. Given this imbalance, Accuracy might provide a less nuanced representation of model performance, particularly when evaluating the model's effectiveness in distinguishing between ore and waste.

In light of these considerations, future research could explore a more diversified suite of metrics and loss functions to obtain a well-rounded view of model performance. For instance, precision, recall, and F1-score could offer a more class-specific understanding, while Intersection over Union (IoU) and Mean IoU could provide insights into the segmentation's quality, especially given the class imbalance.

This comprehensive approach to metrics would serve to deepen our understanding of how well the model performs across a range of criteria, facilitating more informed decisions in real-world mineral operations.

Advancements in Hyperspectral Image Preprocessing:

While this study incorporated preprocessing techniques commonly employed in hyperspectral imaging literature, it did not exhaustively cover the wide array of available preprocessing methods. An intriguing direction for future work would be to explore a more diverse set of preprocessing tech-

niques and assess their impact on both computational speed and model performance. By extending the preprocessing toolbox, future studies could potentially discover methods that enhance both the efficiency and the efficacy of rock classification models, thereby providing a more comprehensive understanding of how preprocessing affects outcomes in mineral operations.

Limitations of Using SAM as a Reference Label This study relies on the Spectral Angle Mapper (SAM) as the reference for pixel-based labeling in hyperspectral images. SAM is widely adopted in the field, yet it comes with its own set of limitations that should be acknowledged [5, 17]. For instance, SAM may not provide a fully accurate representation of the complex spectral characteristics of each pixel. It focuses on the angle between spectral vectors, which can be insensitive to variations brightness levels. In certain scenarios, this could result in misclassifications, especially when materials with similar spectral angles but different actual compositions are present.

Further complicating matters are potential environmental factors that can influence the spectral data, such as variations in lighting conditions or surface roughness. These external variables can create discrepancies between the SAM-based labels and the true material properties of each pixel.

0.5 Conclusion

Building upon previous research that utilized SAM and the ENVI deep learning model for mineral rock classification [1], this study explores a different avenue. Instead of solely aiming for the highest model accuracy, the focus here is on identifying methods that strike a balanced relationship between computational efficiency and segmentation accuracy. While the prior study employed SAM to create reference labels and used ENVI-Net5's U-Net architecture for segmentation, this study introduced a flexible, Python-based pipeline. This new approach provides a methodologically diverse framework for the seamless integration of various preprocessing techniques and segmentation models.

The emphasis on a Python-centric pipeline is not merely a technical pivot but an important step forward for both academic research and practical applications in mineral operations. Hyperspectral images, despite their richness in capturing spectral variations, pose considerable computational challenges. They come with data dimensions that can quickly overwhelm computing systems with limited resources. Large data volumes not only extend training durations but may also incorporate redundant data that doesn't necessarily improve model accuracy. Therefore, the ability to tailor preprocessing methods to negotiate the tradeoff between computational time and model accuracy becomes crucial. This study serves as a case in point, demonstrating how customizable pipelines can be leveraged to explore these important trade-offs, consequently elevating the practical utility of hyperspectral imaging in mineral exploration and mining. Navigating the complex landscape of preprocessing and segmentation, this study primarily draws upon methods explored in prior literature [20, 11, 23]. Yet it distinguishes itself by scaling these techniques to a dataset of significantly larger magnitude, specifically oriented towards mineral applications. Where previous studies often focused on smaller-scale hyperspectral datasets like Pavia and Indian Pines, or diverged into disparate applications.

For preprocessing, a diverse array of techniques were tested, including but not limited to PCA with varying numbers of principal components kept, SNV followed by PCA, EMSC & PCA, Extended Spectral Bands (ESB), K-means clustering, Rank Dissimilarity, Sparse Regression, and Search Skewness. This expansive set of preprocessing methods aimed to distinguish methods that offered good trade-off between computational efficiency and model performance.In terms of segmentation models, the study leveraged the flexibility of the Segmentation_Models library [12] to experiment with a range of architectures: U-Net, UNet++, MAnet, Linknet, FPN, PSPNet, and PAN. This multiplicity not only widens the comparative lens but also contributes to a more nuanced understanding of the advantages and limitations inherent to each model when applied to the large and complex datasets typical in mineral exploration.

Among the assortment of preprocessing methods explored, PCA emerged as a particularly effective technique. The utility of PCA was underscored by its ability to compress the dataset to less than 2% of its original size while capturing 99.8% of the data variance when keeping 5 PCs. This level of compression yielded tangible benefits in both computational speed and model performance. For instance, using a GPU-only approach on the Mist compute system, processing the raw data took a total runtime of 897 seconds. In contrast, when PCA was employed for preprocessing on the Niagara CPU-based system, followed by model training on the Mist GPU, the total runtime was dramatically reduced to 169 seconds. This marks more than a five-fold decrease in computational time. Notably, this gain in efficiency did not come at the cost of model accuracy; in fact, the model's

performance improved by an additional 4.9%.

Turning the attention to the segmentation models, it is worth noting that the range of accuracies across the various architectures was relatively narrow. The best-performing model, UNet++, achieved an average accuracy of 88.74%, while the least effective, DeepLabV3, yielded 84.56%—a modest difference of only 4.16%. Regarding computational time, the most efficient model was the FPN with a training time of 70.7 seconds, whereas DeepLabV3 took an average of 253.5 seconds. Most models exhibited similar computational requirements, with the exception of UNet++ and DeepLabV3, which deviated from them. Ultimately, in this analysis the choice of segmentation model depends on the specific trade-offs one is willing to make between computational time and accuracy. In that context, FPN, LinkNet, U-Net, and UNet++ emerged as models that reside on the Pareto frontier, striking an optimal balance between these two critical factors.

In summary, this study was designed to explore preprocessing methods and various segmentation models specifically in the context of large-scale hyperspectral images for mineral exploration. It provides insights into different methods that achieves a balance between accuracy and efficiency. As such, it stands as a cornerstone for future studies seeking to further optimize image segmentation models in the complex domain of mineral exploration and large hyperspectral imaging.

Bibliography

[1] Mehdi Abdolmaleki, Mariano Consens, and Kamran Esmaeili. "Ore-waste discrimination using supervised and unsupervised classification of hyperspectral images". In: *Remote Sensing* 14.24 (2022), p. 6386.

[2] Muhammad Ahmad et al. "A new statistical approach for band clustering and band selection using K-means clustering". In: *Int. J. Eng. Technol* 3.6 (2011), pp. 606–614.

[3] José Manuel Amigo and Carolina Santos. "Preprocessing of hyperspectral and multispectral images". In: *Data handling in science and technology*. Vol. 32. Elsevier, 2019, pp. 37–53.

[4] Jörg Benndorf and Mike WN Buxton. "Sensor-based real-time resource model reconciliation for improved mine production control–a conceptual framework". In: *Mining Technology* 125.1 (2016), pp. 54–64.

[5] Sujata Chakravarty et al. "Hyperspectral Image Classification using Spectral Angle Mapper". In: *2021 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE)*. 2021, pp. 87–90. DOI: 10.1109/WIECON-ECE54711.2021.9829585.

[6] Chein-I Chang et al. "A joint band prioritization and band-decorrelation approach to band selection for hyperspectral image classification". In: *IEEE Transactions on Geoscience and Remote Sensing* 37.6 (1999), pp. 2631–2641. DOI: 10.1109/36.803411.

[7] Jessica Farrugia et al. "Principal component analysis of hyperspectral data for early detection of mould in cheeselets". In: *Current Research in Food Science* 4 (2021), pp. 18–27.

[8] Yunya Gao et al. "Comparing the robustness of U-Net, LinkNet, and FPN towards label noise for refugee dwelling extraction from satellite imagery". In: *2022 IEEE Global Humanitarian Technology Conference (GHTC)*. 2022, pp. 88–94. DOI: 10.1109/GHTC55712.2022.9911036.

[9] Xiurui Geng et al. "Joint skewness and its application in unsupervised band selection for small target detection". In: *Scientific reports* 5.1 (2015), p. 9915.

[10] Zhouxiao Guo et al. "Semi-supervised hyperspectral band selection via sparse linear regression and hypergraph models". In: *2013 IEEE International Geoscience and Remote Sensing Symposium - IGARSS*. 2013, pp. 1474–1477. DOI: 10.1109/IGARSS.2013.6723064.

[11] Runar Helin et al. "On the possible benefits of deep learning for spectral preprocessing". In: *Journal of Chemometrics* 36.2 (2022), e3374.

[12] Pavel Iakubovskii. *Segmentation Models*. https://github.com/qubvel/segmentation_models. 2019.

[13] Sen Jia et al. "A survey: Deep learning for hyperspectral image classification with few labeled samples". In: *Neurocomputing* 448 (2021), pp. 179–204.

[14] Hong Jing. *Biomedical Image Segmentation: UNet++*. https://towardsdatascience.com/biomedical-image-segmentation-unet-991d075a3a4b. 2019.

[15] Diana Krupnik and Shuhab Khan. "Close-range, ground-based hyperspectral imaging for mining applications at various scales: Review and case studies". In: *Earth-science reviews* 198 (2019), p. 102952.

[16] Brajesh Kumar et al. "Feature extraction for hyperspectral image classification: A review". In: *International Journal of Remote Sensing* 41.16 (2020), pp. 6248–6287.

[17] Erzsébet Merényi et al. "Classification of hyperspectral imagery with neural networks: comparison to conventional tools". In: *EURASIP Journal on Advances in Signal Processing* 2014.1 (2014), pp. 1–19.

[18] Daifeng Peng, Yongjun Zhang, and Haiyan Guan. "End-to-end change detection for high resolution satellite images using improved UNet++". In: *Remote Sensing* 11.11 (2019), p. 1382.

[19] Alex Rodriguez and Alessandro Laio. "Clustering by fast search and find of density peaks". In: *science* 344.6191 (2014), pp. 1492–1496.

[20] Weiwei Sun and Qian Du. "Hyperspectral Band Selection: A Review". In: *IEEE Geoscience and Remote Sensing Magazine* 7.2 (2019), pp. 118–139. DOI: 10.1109/MGRS.2019.2911100.

[21] Zhang Ting-ting and Liu Fei. "Application of hyperspectral remote sensing in mineral identification and mapping". In: *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*. 2012, pp. 103–106. DOI: 10.1109/ICCSNT.2012.6525900.

[22] Hongsheng Wang et al. "Deep-learning-based workflow for boundary and small target segmentation in digital rock images using UNet++ and IK-EBM". In: *Journal of Petroleum Science and Engineering* 215 (2022), p. 110596.

[23] Liguo Wang and Chunhui Zhao. *Hyperspectral image processing*. Springer, 2016.

[24] Tianyi Wu et al. "Tree-structured kronecker convolutional network for semantic segmentation". In: *2019 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE. 2019, pp. 940–945.

[25] Lu Xu et al. "Ensemble preprocessing of near-infrared (NIR) spectra for multivariate calibration". In: *Analytica chimica acta* 616.2 (2008), pp. 138–143.