Machine Learning for Classification of Hyperspectral Data

Kadiyala Abhiram, Rasagna C, Krishna

April 27, 2021

1 Introduction

Hyper-spectral data is very useful in the sense that we can get a continuous curve for reflectance instead of just a few discrete wavelengths. This type of data helps us identify the materials contained in the pixel. Initial methods attempt to create abundance maps of materials that give the proportion of the elements that give rise to the pixel value. These kind of **abundance maps** can be thought of as **soft classifications**. Some classical Machine Learning methods were used to classify the pixels in the pre-defined classes which is basically **hard classification**. Our interest primarily concentrates on the hard classification approaches. In this report, we will explore classical methods (both ML and non-ML approaches) and then, with emphasis, deep learning approaches. This area received renewed interest by the data science community due to the advent of deep learning. Section 2 broadly discusses the problem with some pre-processing and data acquisition methods. Sections 3 and 4 discuss the classical methods used. Section 5 discusses the methods we have implemented. Section 6 presents a summary of the results and methods used. Section 7 and References discuss the prior work in this area.

1.1 Problem Statement

Typically, remote sensing for hyperspectral data involves a **scene** on which we apply our methods. Using some ground work, we find the true labels and classes of elements in the scene. This data helps us find the ground control points or truth labels to train our models. While hyperspectral data has its own advantages, there is one crucial disadvantage which allows us not to apply the already accomplished computer vision approaches. That disadvantage is the low spatial resolution. This can be overcomed by using some techniques which we will talk about in this project. As far as the scene goes, we have a fixed image of size $x \times x$ pixels. Each pixel has a truth assigned to it i.e. there is some dominant material in the ground underneath the pixel. So we basically have x^2 pixels and x^2 labels that makes x^2 training examples. Each pixel has values in different bands. For normal data, usually we have around 8 bands. For multispectral data we have around 20-30 bands. Hyperspectral data typically has in the order of 1 or 2 hundred bands.

1.2 Dataset and Acquisition

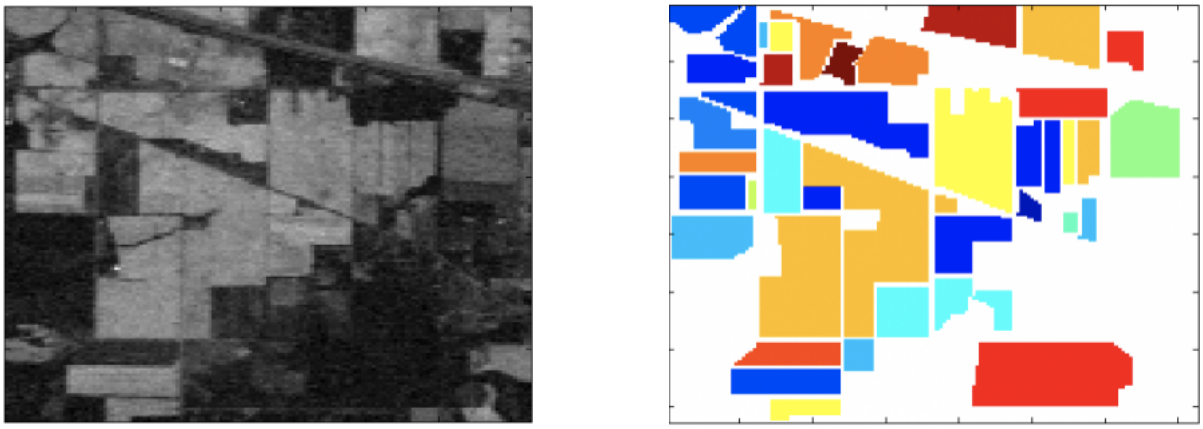
There are three prominent datasets that are available with hyperspectral data widely available. These are

- Pavia University Dataset
- Salinas Dataset

- Indian Pines

All of these datasets are cleaned and annotated with proper labels with the help of some on-ground investigation. We primarily focus on the Indian Pines dataset. These datasets are consolidated at `http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes`.

This scene was gathered by AVIRIS sensor over the Indian Pines test site in North-western Indiana and consists of 145×145 pixels and 224 spectral reflectance bands in the wavelength range 0.4–$2.5\ 10^{-6}$ meters. This scene is a subset of a larger one. The Indian Pines scene contains two-thirds agriculture, and one-third forest or other natural perennial vegetation. There are two major dual lane highways, a rail line, as well as some low density housing, other built structures, and smaller roads. Since the scene is taken in June some of the crops present, corn, soybeans, are in early stages of growth with less than 5% coverage. The ground truth available is designated into sixteen classes and is not all mutually exclusive. We have also reduced the number of bands to 200 by removing bands covering the region of water absorption. Some of the pixels without a definitive material are



(a) A Sample band image

(b) Ground Truths

Figure 1: Sample Images

removed from the data set. This is the basic cleaning step. The next step, PCA, helps us both remove redundancy (cleaning) and also helps us visualize the maximum variance bands as explained in our lecture. The PCA components are shown in the next figure-

1.3 Interpretation of the Dataset

As mentioned before, one pixel forms one training example. This is given as one training example to the ML classifiers. This is called as **per-pixel** approach as discussed in class. We can also have **object based** classification too. The figure below shows the per-pixel approach.

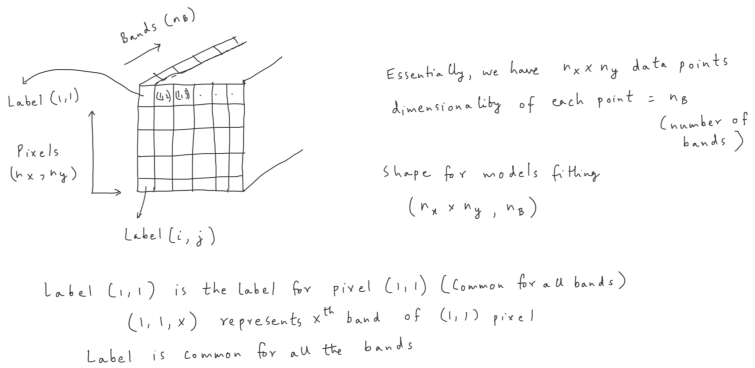


Figure 2: Per-Pixel Approach for Classification

This approach yields reasonable results. However, it is found that using the contextual spatial information yields better results. Basically, we use the neighbouring pixels for training this contextual information yields good results. The pictorial representation is shown below-

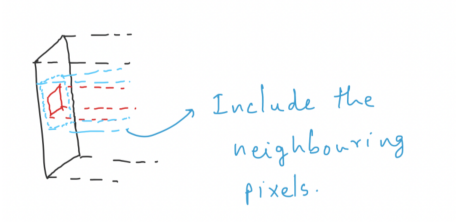


Figure 3: Using the Neighboring pixels for Classification

This is the style used in many of the recent deep learning approaches. The methods using these types of spatial information in addition to the spectral information are called *spatial-spectral* methods. We will later look at some of the spatial spectral methods later.

1.3.1 PCA

PCA is the process of finding the components which have the highest variance in data. We can compress the information in bands using only the first few components. However, we have to be careful here. There could be some important discriminatory information in the hyperspectral bands that might be rejected as noise. So there is an important tradeoff to consider here. The basic process of finding principal components is shown below-

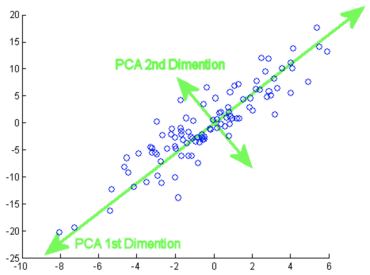


Figure 4: Examples of showing Principle Components of 2D data

We applied the standard PCA method to our data set. The results are shown in the following figures-

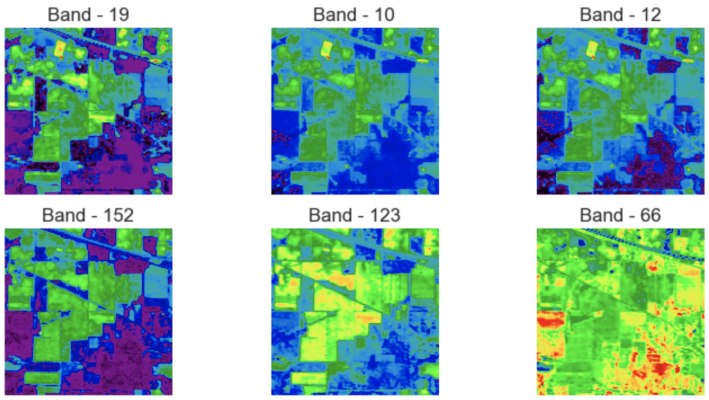


Figure 5: A few PCs

The variance vs the number of bands selected in shown in the following figure-

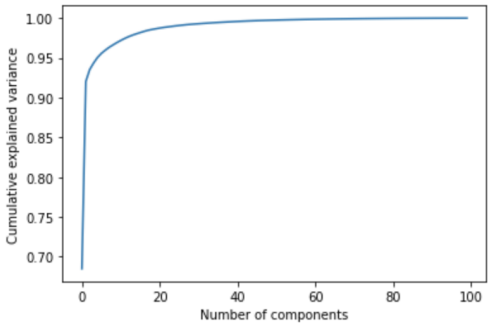


Figure 6: Variance vs no of bands selected

As an additional pre-processing step we could do unsupervised learning. K-means algorithm is a very popular unsupervised classification algorithm. For an example we show the result of work by [1]-

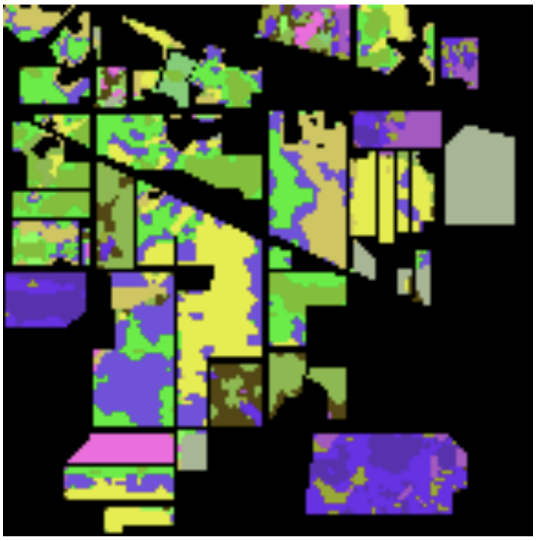


Figure 7: K-means Classification

We next discuss some of the methods used to solve the problem using standard methods.

1.4 Classical Methods in Literature That Don't use Machine Learning

There are two main classical approaches that we discuss are spectral unmixing and spectral angle mapper. Broadly, spectral unmixing assumes that the value of the pixel is a linear combination of reflectances of the materials in the mixture underneath the pixel. This gives rise to the name *unmixing*. Spectral angle mapper calculates the angle between the pixel and the vectors of materials available in a predefined library. The class with least angle is assigned. We will discuss it in a bit of detail in the next subsections.

1.4.1 Spectral Unmixing

Spectral Unmixing is the process unmixing a spectral signature to a set of original spectral signals of some set of prime elements(called as end members) [2][3]. Then a pixel at coordinates (i,j) can be linearly modeled as

$$\phi_{i,j} = \sum_{k=1}^{n} \lambda_k S_k$$

where $S_1, ..S_n$ represent the spectral signatures of end members and $\lambda_1, .., \lambda_n$ are the decomposition coefficients(abundance maps) which represent there proportional contributions to the pixel. If we know the S_k values then the linear equation can be solved to obtain the λ_k values i.e., the abundance maps.

In **Supervised Spectral Unmixing**, user provides the spectral signatures of the end members(S_k), so, given the knowledge about end members one can simply find the abundance maps by solving the linear equation. The problem with this technique is finding the correct S_k values as they are error prone because of the user interaction. Whereas in **Unsupervised Spectral Unmixing** we identify the end members and the mixture directly from $\phi_{i,j}$ values without any user interactions.

1.4.2 Spectral Angle Mapper

Spectral Angle Mapper(SAM) is a spectral classification that uses an n-dimensional angle to match pixels with reference spectra [4]. Spectral similarity between two spectra can be determined by calculating the angle between the two spectra. In SAM we measure the angles between the spectra of the pixels with all reference spectra and the class is assigned to the reference spectra with least angle. The angle between two spectra can be calculated as

$$\alpha = cos^{-1} \frac{\sum_{i=1}^{n} p_i r_i}{\sqrt{\sum_{i=1}^{n} p_i^2} \sqrt{\sum_{i=1}^{n} r_i^2}}$$

where α is angle between the two spectra, p is the spectra corresponding to the pixel, r is the reference spectra, and n represents the no. of bands. Using this formula angles with all the reference spectra can be calculated and the reference spectra with least angle is selected and the pixel is assigned to the corresponding class.

1.5 Classical ML Methods Implemented

Now that we have the structure for per pixel approach, we can now plug these into any classical machine learning models we have. We have implemented nearly every classical method available. We have looked at K-Nearest Neighbors algorithm, Logistic Regression, Support Vector Machines and Decision Tree Approaches. We will discuss each method briefly and produce the results. We used a tree of *size of 10*.

1.5.1 Decision Tree

Decision tree forms a hierarchy of comparisons which leads the algorithm to classify the sample to one of the classes. The depth of the tree is very important here. Deeper models might overfit the data and might not give the required results. Hence the depth must be chosen wisely.

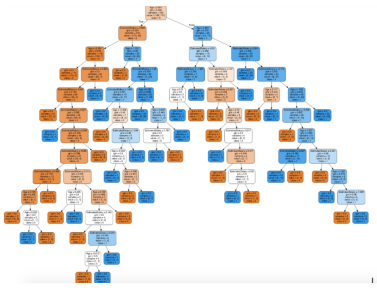
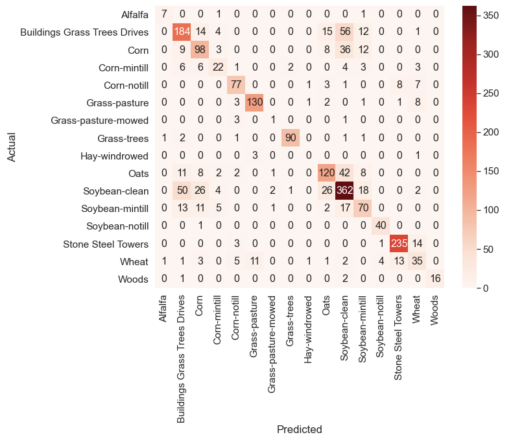


Figure 8: An example of a decision tree that is formed.

Our classifier had given decent results. The results are shown below- We can calculate κ from the confusion matrix. F-1 score, Accuracy, Precision and Recall all offer similar information.



(a) Confusion Matrix

	precision	recall	f1-score	support
Alfalfa	0.78	0.78	0.78	9
Corn-notill	0.66	0.64	0.65	286
Corn-mintill	0.59	0.59	0.59	166
Corn	0.54	0.47	0.50	47
Grass-pasture	0.81	0.79	0.80	97
Grass-trees	0.90	0.89	0.90	146
Grass-pasture-mowed	0.20	0.20	0.20	5
Hay-windrowed	0.97	0.94	0.95	96
Oats	0.00	0.00	0.00	4
Soybean-notill	0.68	0.62	0.65	194
Soybean-mintill	0.69	0.74	0.71	491
Soybean-clean	0.56	0.59	0.57	119
Wheat	0.89	0.98	0.93	41
Woods	0.91	0.93	0.92	253
Buildings Grass Trees Drives	0.49	0.45	0.47	77
Stone Steel Towers	1.00	0.84	0.91	19
accuracy			0.73	2050
macro avg	0.67	0.65	0.66	2050
weighted avg	0.72	0.73	0.72	2050

(b) Precision, F1, Recall Scores

Figure 9

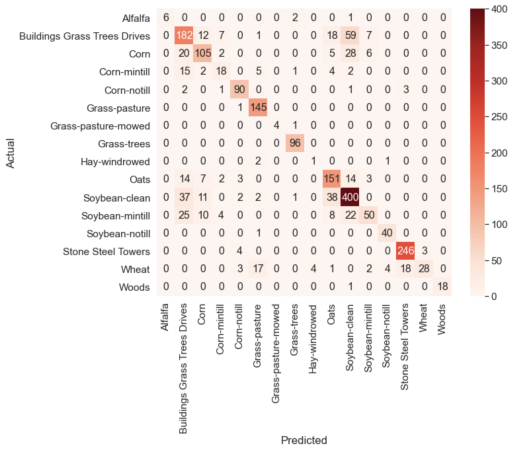
We then implemented K-Nearest Neighbors algorithm.

1.5.2 K-Nearest Neighbors Algorithm

We took 10 nearest neighbors and attributed it to the majority class. The Results are shown in *figure 10*.

	precision	recall	f1-score	support
Alfalfa	1.00	0.67	0.80	9
Corn-notill	0.62	0.64	0.63	286
Corn-mintill	0.71	0.63	0.67	166
Corn	0.53	0.38	0.44	47
Grass-pasture	0.87	0.93	0.90	97
Grass-trees	0.84	0.99	0.91	146
Grass-pasture-mowed	1.00	0.80	0.89	5
Hay-windrowed	0.95	1.00	0.97	96
Oats	0.20	0.25	0.22	4
Soybean-notill	0.67	0.78	0.72	194
Soybean-mintill	0.76	0.81	0.79	491
Soybean-clean	0.74	0.42	0.53	119
Wheat	0.89	0.98	0.93	41
Woods	0.92	0.97	0.95	253
Buildings Grass Trees Drives	0.90	0.36	0.52	77
Stone Steel Towers	1.00	0.95	0.97	19
accuracy			0.77	2050
macro avg	0.79	0.72	0.74	2050
weighted avg	0.77	0.77	0.76	2050

(a) Confusion Matrix



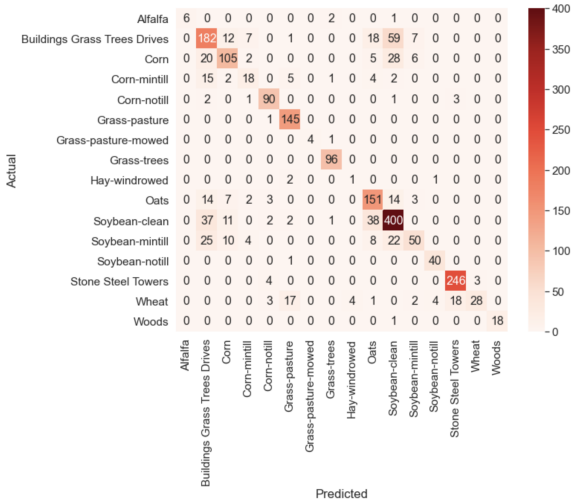
(b) Precision, F1, Recall Scores

Figure 10: KNN results

We then turn to one of the most popular methods which is SVM.

1.5.3 SVM

SVM is a very popular method in Classical circles. This is due to the fact that we can have many kernels which help in non linear classification. We implemented the liner, polynomial and RBF kernel. RBF kernel gives us the best answer. The results for RBF Kernel are shown below-



(a) Confusion Matrix

	precision	recall	f1-score	support
Alfalfa	1.00	0.89	0.94	9
Corn-notill	0.92	0.89	0.90	286
Corn-mintill	0.91	0.85	0.88	166
Corn	0.81	0.81	0.81	47
Grass-pasture	0.94	0.96	0.95	97
Grass-trees	0.99	0.97	0.98	146
Grass-pasture-mowed	1.00	0.80	0.89	5
Hay-windrowed	0.99	1.00	0.99	96
Oats	0.67	0.50	0.57	4
Soybean-notill	0.91	0.86	0.88	194
Soybean-mintill	0.87	0.94	0.90	491
Soybean-clean	0.91	0.92	0.92	119
Wheat	0.95	1.00	0.98	41
Woods	0.97	0.98	0.97	253
Buildings Grass Trees Drives	0.88	0.78	0.83	77
Stone Steel Towers	1.00	1.00	1.00	19
accuracy			0.92	2050
macro avg	0.92	0.88	0.90	2050
weighted avg	0.92	0.92	0.92	2050

(b) Precision, F1, Recall Scores

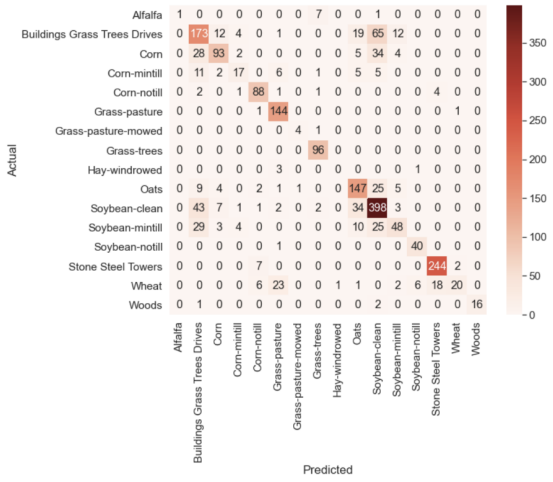
Figure 11: SVM RBF Kernel Results

1.5.4 Naive Bayes Method

Naive Bayes method is very similar to maximum likelihood estimation of the class. We show the results below. This classifier gives decent results due to the quadratic nature.and decent non linearity.

	precision	recall	f1-score	support
Alfalfa	1.00	0.11	0.20	9
Corn-notill	0.58	0.60	0.59	286
Corn-mintill	0.77	0.56	0.65	166
Corn	0.59	0.36	0.45	47
Grass-pasture	0.84	0.91	0.87	97
Grass-trees	0.79	0.99	0.88	146
Grass-pasture-mowed	0.80	0.80	0.80	5
Hay-windrowed	0.89	1.00	0.94	96
Oats	0.00	0.00	0.00	4
Soybean-notill	0.67	0.76	0.71	194
Soybean-mintill	0.72	0.81	0.76	491
Soybean-clean	0.65	0.40	0.50	119
Wheat	0.85	0.98	0.91	41
Woods	0.92	0.96	0.94	253
Buildings Grass Trees Drives	0.87	0.26	0.40	77
Stone Steel Towers	1.00	0.84	0.91	19
accuracy			0.75	2050
macro avg	0.75	0.65	0.66	2050
weighted avg	0.75	0.75	0.73	2050

(a) Confusion Matrix



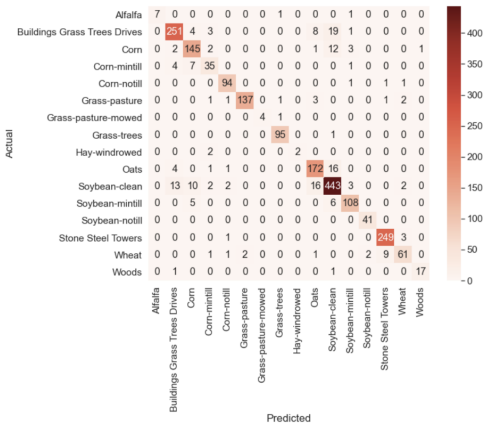
(b) Precision, F1, Recall Scores

1.5.5 Feed Forward Neural Network

We then looked at a shallow neural network. We used two hidden layers (1000 units, 400 units) in our neural network. The shallow network yields the following results and by far somewhat matches SVM.

	precision	recall	f1-score	support
Alfalfa	1.00	0.78	0.88	9
Corn-notill	0.91	0.88	0.89	286
Corn-mintill	0.85	0.87	0.86	166
Corn	0.74	0.74	0.74	47
Grass-pasture	0.94	0.97	0.95	97
Grass-trees	0.99	0.94	0.96	146
Grass-pasture-mowed	1.00	0.80	0.89	5
Hay-windrowed	0.97	0.99	0.98	96
Oats	1.00	0.50	0.67	4
Soybean-notill	0.86	0.89	0.87	194
Soybean-mintill	0.89	0.90	0.90	491
Soybean-clean	0.92	0.91	0.91	119
Wheat	0.95	1.00	0.98	41
Woods	0.96	0.98	0.97	253
ıildings Grass Trees Drives	0.88	0.79	0.84	77
Stone Steel Towers	0.94	0.89	0.92	19
accuracy			0.91	2050
macro avg	0.93	0.86	0.89	2050
weighted avg	0.91	0.91	0.91	2050

(a) Confusion Matrix



(b) Precision, F1, Recall Scores

Figure 13: Feed Forward Shallow Neural Network

This concludes our discussion on the classical ML based spectral methods.

1.6 Deep Learning Methods And Some Results Proposed

We have seen that the classical methods top out at around 90% accuracy. To improve this, we could go deeper in the neural network model we discussed above. But this leads to overfitting and actually decreases. Here, we need to include more information which will help us increase the accuracy of the classifiers. This is done by using the neighboring pixels to help discriminate with more accuracy. This now gives us a hyper cube which we have to use as our input to our classifier. The width of the neighborhood of the pixel we take is defined by the *patch size*. This is a very important hyper-parameter because a lower patch size might not help much because there's **not much new information** and a higher patch might **dilute** the information. All the recent methods use a patch of varying size. They do however differ in the patch size, Architecture and types of convolutions used. The first such paper is [5].

1.6.1 1D CNN Approach

[5] used a 1D convolutions and initially 20 feature maps. The feature maps are fed to the hidden layers and the output softmax layer. The architecture is shown below.

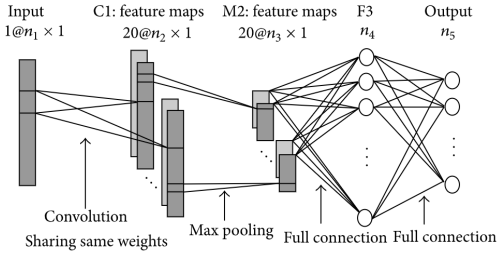
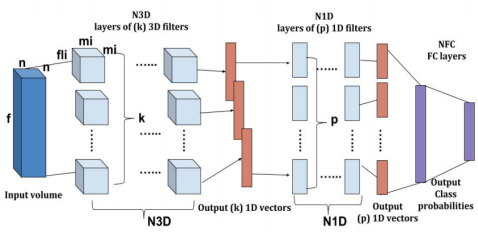


Figure 14: 1D: Architecture

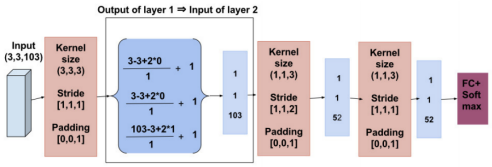
This approach has not yet used the patch method we should note. The next natural idea is to use spatio-spectral methods with the neighboring pixels. That is discussed in the next subsection.

1.6.2 3D CNN Approach: Hamida et al

This [6] is perhaps the most popular CNN spatio-spectral method out there. The architecture and the hyper-parameters are detailed in the figure. They have used a patch size of both 3×3 and 5×5. The results are also shown.



(a) Architecture Hyper-paramters



(b) 3D: CNN Architecture

Figure 15: 3D CNN proposed by Hamida et al.

PROCESSING TIME REQUIRED TO REACH 95% OF THE FINAL ACCURACY ON THE PAVIA UNIVERSITY DATA SET

Architecture	Iterations Nb	Processing Time (s)	Accuracy (early)	final Accuracy (100%)
8 layers 1×1	23733	313	85.8%	90.3%
8 layers 3×3	23500	95	88.3%	92.9%
8 layers 5×5	**23233**	**2535**	**92.4%**	**97.2%**
6 layers 5×5	23233	40000	89.8%	94.6%
4 layers 5×5	23233	3000	89.1%	93.8%

Figure 16: Results with multiple patches

We look at one more 3D CNN approach before moving on the state of the art model.

1.6.3 3D CNN Approach: Lee et al

Lee et al [7] used the following architecture-

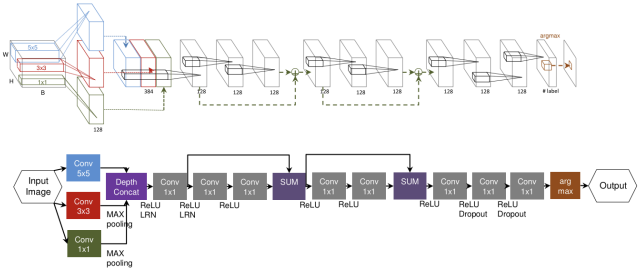


Figure 17: Modified Alex Net used by Lee et al

They used patches of size 3,5 and 7 and then concatenated them. The results are similar to [6] but somewhat inferior.

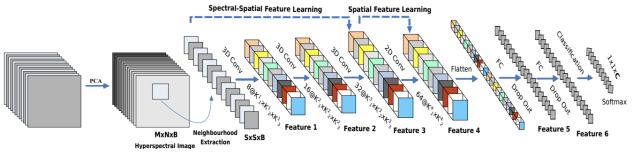
Similar 3D CNN methods were proposed in the literature. We will cite them in the references. We now turn to our own implementations .

2 Our Methods And Implementations

We have implemented the current state of the art model along with our own architecture which uses stripped down resnet model.

2.1 Current State of the Art Model

Current state of the world model is called Hybrid SN: They have used both 3D and 2D convolutions to create a hybrid. The model architecture is shown below-



(a) Architecture

Layer (type)	Output Shape	# Parameter
input_1 (InputLayer)	(25, 25, 30, 1)	0
conv3d_1 (Conv3D)	(23, 23, 24, 8)	512
conv3d_2 (Conv3D)	(21, 21, 20, 16)	5776
conv3d_3 (Conv3D)	(19, 19, 18, 32)	13856
reshape_1 (Reshape)	(19, 19, 576)	0
conv2d_1 (Conv2D)	(17, 17, 64)	331840
flatten_1 (Flatten)	(18496)	0
dense_1 (Dense)	(256)	4735232
dropout_1 (Dropout)	(256)	0
dense_2 (Dense)	(128)	32896
dropout_2 (Dropout)	(128)	0
dense_3 (Dense)	(16)	2064
Total Trainable Parameters: $5,122,176$		

(b) 3D-2D: Layers

Figure 18: Hybrid SN

We have implemented this model and we have achieved similar to the performance claimed in the paper (over 99% accuracy).

Methods	Indian Pines Dataset		
	OA	Kappa	AA
SVM	85.30 ± 2.8	83.10 ± 3.2	79.03 ± 2.7
2D-CNN	89.48 ± 0.2	87.96 ± 0.5	86.14 ± 0.8
3D-CNN	91.10 ± 0.4	89.98 ± 0.5	91.58 ± 0.2
M3D-CNN	95.32 ± 0.1	94.70 ± 0.2	96.41 ± 0.7
SSRN	99.19 ± 0.3	99.07 ± 0.3	98.93 ± 0.6
HybridSN	99.75 ± 0.1	99.71 ± 0.1	99.63 ± 0.2

Figure 19: Hybrid SN: Comparision(given by the authors)

We show our own achieved results using this model.

```
1.0898319073021412 Test loss (%)
99.66550469398499 Test accuracy (%)

99.61860915351764 Kappa accuracy (%)
99.66550522648083 Overall accuracy (%)
99.59556473688657 Average accuracy (%)

                                precision    recall  f1-score   support

                       Alfalfa       1.00      1.00      1.00        32
                    Corn-notill       0.99      0.99      0.99      1000
                    Corn-mintill       1.00      1.00      1.00       581
                            Corn       0.99      1.00      1.00       166
                  Grass-pasture       1.00      0.99      0.99       338
                    Grass-trees       1.00      1.00      1.00       511
            Grass-pasture-mowed       1.00      1.00      1.00        20
                    Hay-windrowed       1.00      1.00      1.00       335
                             Oats       1.00      1.00      1.00        14
                    Soybean-notill       1.00      1.00      1.00       680
                    Soybean-mintill       1.00      1.00      1.00      1719
                      Soybean-clean       1.00      0.99      1.00       415
                            Wheat       0.99      0.99      0.99       143
                            Woods       1.00      1.00      1.00       886
Buildings-Grass-Trees-Drives       1.00      0.99      0.99       270
          Stone-Steel-Towers       0.98      0.98      0.98        65

                      accuracy                           1.00      7175
                     macro avg       1.00      1.00      1.00      7175
                  weighted avg       1.00      1.00      1.00      7175
```

Figure 20: Classification Report and Kappa

There are some diadvantages of this model. This model is **shallow and has many parameters(5 Million Parameter)**. More expressive power can be achieved by using deeper network and fewer parameters. This leads to the idea of ResNets.

2.2 Our New Architecture Based on ResNets

ResNets can go very deep due to identity blocks and skip connection. This way we have a deep network and fewer parameters. We took the original ResNet (**Res**idual **Net**work) and only took the 1/2 layer i.e. we took an 18 layer (excluding input and output layers) network. We show the original ResNet architecture below-

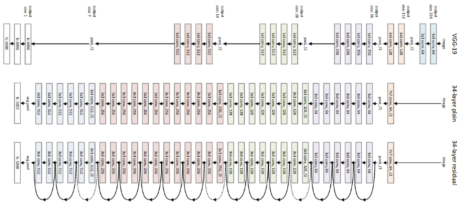


Figure 32: Original ResNet Proposed, We used a stripped down 18 layer version

Figure 21: ResNet Architecture

This model has far fewer parameters ($200k$ parameters) and trains a bit more easily. We do have similar expressive nature using more layers but less number of parameters. The classification report is shown below-

	precision	recall	f1-score	support
0	1.00	1.00	1.00	9
1	1.00	0.98	0.99	286
2	0.93	1.00	0.97	166
3	1.00	0.96	0.98	47
4	1.00	0.87	0.93	97
5	0.99	1.00	0.99	146
6	1.00	1.00	1.00	5
7	1.00	1.00	1.00	96
8	0.80	1.00	0.89	4
9	0.99	0.98	0.99	194
10	0.99	0.99	0.99	491
11	0.99	0.94	0.97	119
12	0.93	1.00	0.96	41
13	1.00	1.00	1.00	253
14	0.87	1.00	0.93	77
15	0.84	0.84	0.84	19
accuracy			0.98	2050
macro avg	0.96	0.97	0.96	2050
weighted avg	0.98	0.98	0.98	2050

Figure 22: Confusion Matrix

We found that the patch size of 21×21 is the optimal patch size for our model. There is a tradeoff here- We are trading of a little bit of accuracy for reducing the number of parameters largely. We summarize the results in the next page

3 Summary of Results and Discussion

Model	Performance
Decision Tree	73% (Our Implementation)
KNN (K=10)	77% (Our Implementation)
SVM (RBF kernel)	92% (Our Implementation)
Neural Network (Shallow)	91%(Our Implementation)
1D CNN	90.16%
2D CNN: Lee Kwon	94.6%
2D CNN: Hamida	99.4%
Hybrid SN: 3D-2D	99.7% (Our Implementation)
Modified ResNet(Ours) (18 Layers)	98%(Our Implementation)

We can see that the classical methods top out at 90%. To improve the efficiency, we used patches and deeper networks and then we discussed the results. We achieved close to full performance.

4 Acknowledgements

We would like to acknowledge the following sources-

- `https://github.com/samkreter/kmeans-clustering-with-spatial-bias`
- `http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes`
- `https://github.com/nshaud/DeepHyperX`

We also thank Prof RC Prasad sir for valuable inputs in phase 2.

References

[1] *Dimensionality Reduction of Hyperspectral Images for Classification*, Michael Wong

[2] *unmixing hyperspectral data, L. Parra et al.*

[3] *Deep Learning for Classification of Hyperspectral Data: A Comparative Review, Nicolas Audebert et al.*

[4] *The Spectral Image Processing System (SIPS): Interactive visualization and analysis of imaging spectrometer data, Kruse et al.*

[5] *Deep Convolutional Neural Networks for Hyperspectral Image Classification*

[6] *3-D Deep Learning Approach for Remote Sensing Image Classification, Hamida et al., TGRS 2018*

[7] *Going Deeper with Contextual CNN for Hyperspectral Image Classification*

[8] *Deep Feature Extraction and Classification of Hyperspectral Images Based on Convolutional Neural Networks*

[9] *HSI-CNN: A Novel Convolution Neural Network for Hyperspectral Image, Luo et al, ICPR 2018*

[10] *HybridSN: Exploring 3D-2D CNN Feature Hierarchy for Hyperspectral Image Classification*

[11] *Sklearn Documentation*