Machine Learning for Classification of Hyperspectral Data

Abhiram, Rasagna, Krishna

IIIT Hyderabad

April 29, 2021

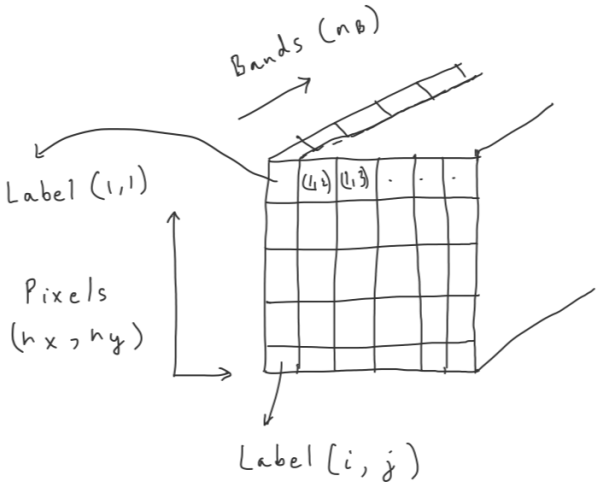
What is Hyperspectral Data

- Hyperspectral sensors measure the intensity of radiant flux over a particular given wavelength. (Measured in Watts per Meter Steradian)
- This gives us the opportunity to have sensors with high spectral resolution capture spectral reflectance over a wide range.
- Spectral reflectance properties give us the opportunity to potentially detect the surfaces from which the light has bounced off.
- This can be helpful for detecting different kinds of vegetation and some artificial elements like Steel/Stone

Dataset Structure

- The dataset is acquired over a particular scene setting (The area over which the data was captured is the same)
- There are multiple wavelengths over which the data is captured. The number of bands is typically around 100-200 bands.
- Here, each pixel is separately annotated for the material that is present underneath. The pictorial representation will be shown in the next slide.

Dataset Structure(Cont'd)



Essentially, we have $n_x \times n_y$ data points

dimensionality of each point = n_B (number of bands)

Shape for models fitting

$(n_x \times n_y, n_B)$

Label (1,1) is the label for pixel (1,1) (Common for all bands)

(1, 1, x) represents x^{th} band of (1,1) pixel

Label is common for all the bands

Figure 1: Visualization of the data

Dataset Structure

Now that we have established the structure of the data, we can use all the traditional methods of Machine Learning to get some preliminary base line results. Before we describe those methods, We will first give a brief introduction to the data set we are using for this purpose.

Indian Pines Dataset

- Indian Pines is a dataset captured using the AVIRIS sensor.
- The fixed scene covers agricultural areas in North Western Indiana.
- The picture/scene is of size of 145px $\times$ 145px and there are 224 spectral bands. The water absorption bands are removed and after some trivial processing, we have around 200 bands at our disposal. There are 16 classes (Rare classes are excluded and the pixels are removed from the dataset)
- The classes and their description are shown in the next slide.

Indian Pines Dataset: Classes

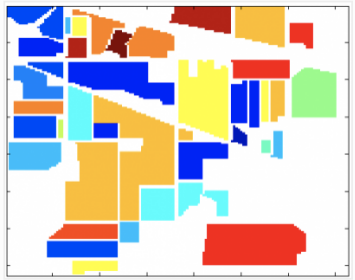
Groundtruth classes for the Indian Pines scene and their respective samples number

#	Class	Samples
1	Alfalfa	46
2	Corn-notill	1428
3	Corn-mintill	830
4	Corn	237
5	Grass-pasture	483
6	Grass-trees	730
7	Grass-pasture-mowed	28
8	Hay-windrowed	478
9	Oats	20
10	Soybean-notill	972
11	Soybean-mintill	2455
12	Soybean-clean	593
13	Wheat	205
14	Woods	1265
15	Buildings-Grass-Trees-Drives	386
16	Stone-Steel-Towers	93

Figure 2: Indian Pines Classes

Indian Pines Dataset: Scene and Ground Truth

Sample band of Indian Pines dataset



Groundtruth of Indian Pines dataset

Figure 3: Indian Pines Scene/Ground Truth [1]

Classical Methods

- The standard/classical approaches share some preprocessing techniques like **Band Selection**, **Spectral Normalization** etc. These techniques are somewhat known to ML community too.
- The classical approaches are divided into two types- 1. Spectral Classification, 2. Spatial-Spectral Classification.
- *Unmixing* is a classical method which models a pixel value to be a linear combinations of various sources and then uses that to predict/classify the material underneath using the *spectral reflectance* curves.
- Spatio-Spectral methods leverage the spatial dependencies for classification of the data. One example of this would be *Pre-segmentation*.

Results

We have mainly concentrated on the spectral Machine Learning Classical approaches. We aim to use advanced deep learning models (RNN, CNN, Resnets) both in the spectral and spacio-spectral domains in the final project presentations. We have used the following models to fit our data-

- PCA (Preprocessing)
- Band Selection (Removing the water absorption bands)
- Decision Tress
- KNN (K-Nearest Neighbors Classification)
- SVM (Linear kernel, Polynomial Kernel, RBF kernel)
- Feed Forward Neural Networks

Results (1) PCA

PCA is the method of projecting the data onto the orthogonal vectors with the maximum variance. The bands in the hyperspectral data have very high correlation and can be reduced by using PCA. The PCA results for the Indian Pines dataset are shown below. The variance flattens at around 100 bands and is a reasonable dimension to reduce into.

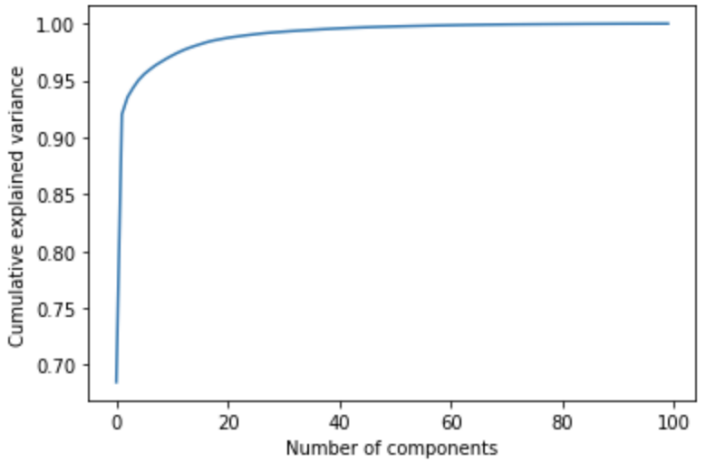


Figure 4: Variance w.r.t number of PC included

Results (2) PCA Components

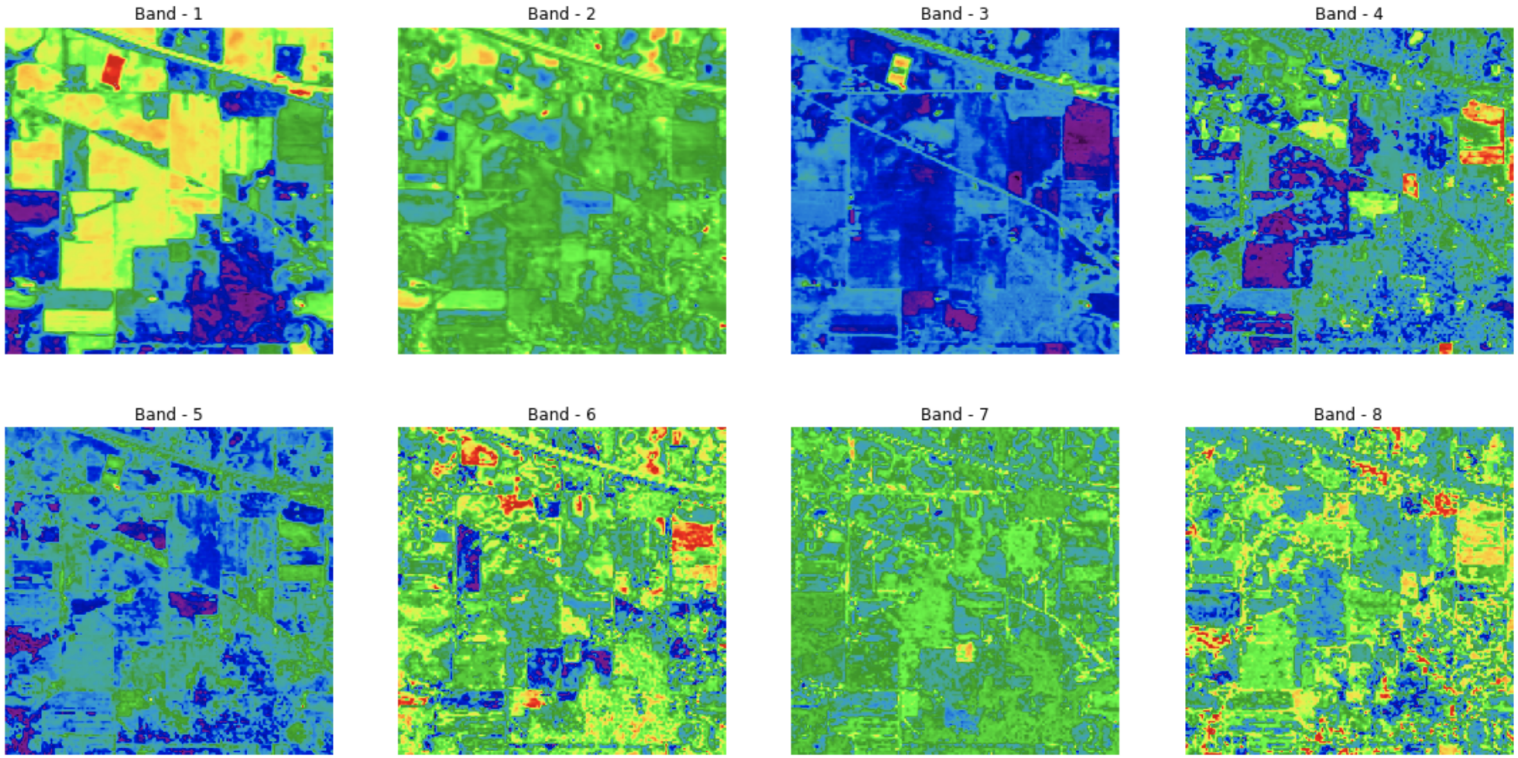


Figure 5: Pricipal Components Visualization

Results (3) Decision Trees

We have used Decision Trees of depth 10 to achieve reasonable results without overfitting. The confusion matrix for the classification is shown below-

	precision	recall	f1-score	support
Alfalfa	0.73	0.89	0.80	9
Corn-notill	0.66	0.65	0.65	286
Corn-mintill	0.59	0.58	0.58	166
Corn	0.47	0.40	0.44	47
Grass-pasture	0.78	0.82	0.80	97
Grass-trees	0.91	0.88	0.90	146
Grass-pasture-mowed	0.00	0.00	0.00	5
Hay-windrowed	0.98	0.95	0.96	96
Oats	0.20	0.25	0.22	4
Soybean-notill	0.67	0.63	0.65	194
Soybean-mintill	0.69	0.74	0.71	491
Soybean-clean	0.56	0.55	0.56	119
Wheat	0.89	0.98	0.93	41
Woods	0.89	0.92	0.90	253
Buildings Grass Trees Drives	0.43	0.35	0.39	77
Stone Steel Towers	1.00	0.84	0.91	19
accuracy			0.72	2050
macro avg	0.65	0.65	0.65	2050
weighted avg	0.72	0.72	0.72	2050

Figure 6: Confusion Matrix

Results (4) Decision Trees

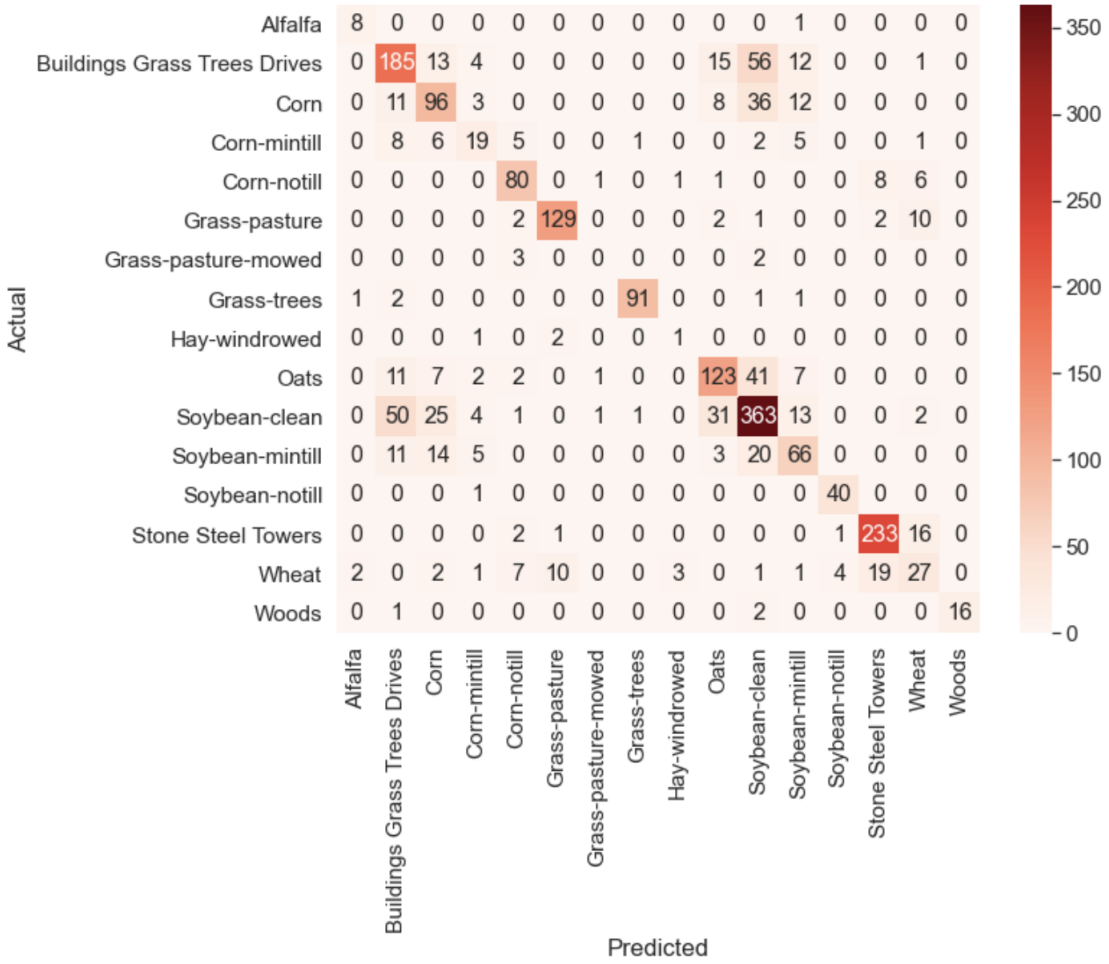


Figure 7: Predictions Stats

Results (5) KNN with K=10

We look at 10 nearest neighbors to make a decision in the euclidean space.

	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

Figure 8: KNN confusion Matrix

Results (6) KNN with K=10

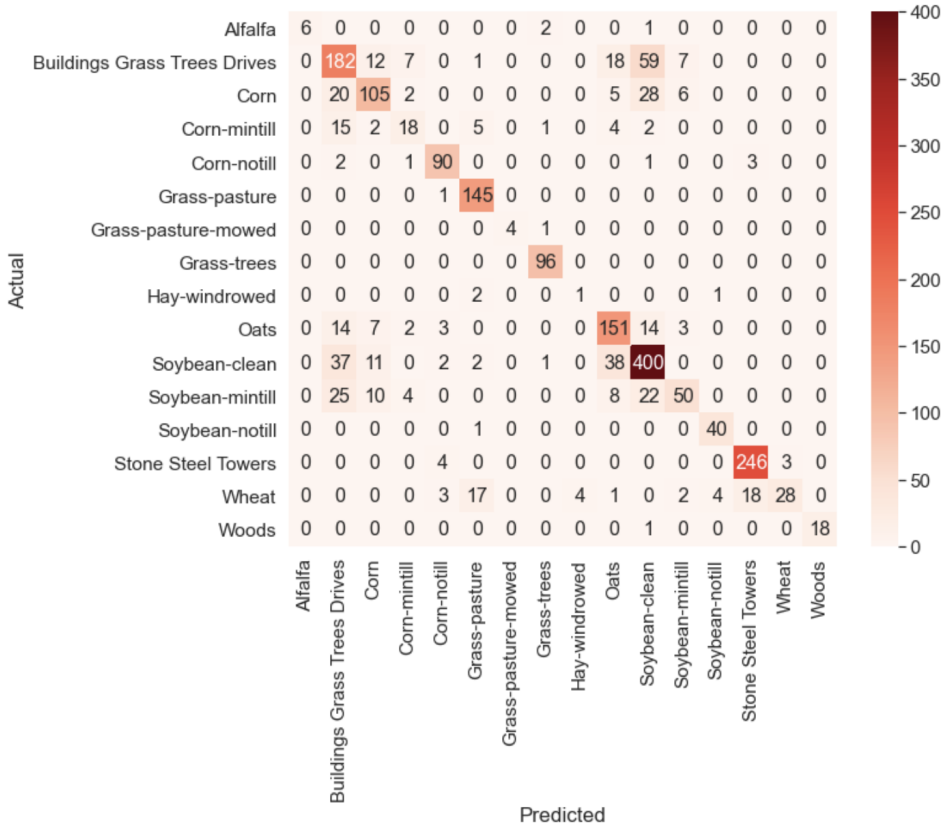


Figure 9: Prediction Stats

Results (7) Naive Bayes Classification

We can use MAP/ML rules to classify the samples too. The results are shown below-

	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

Figure 10: Confusion Matrix for Naive Bayes

Results (8) Naive Bayes Classification

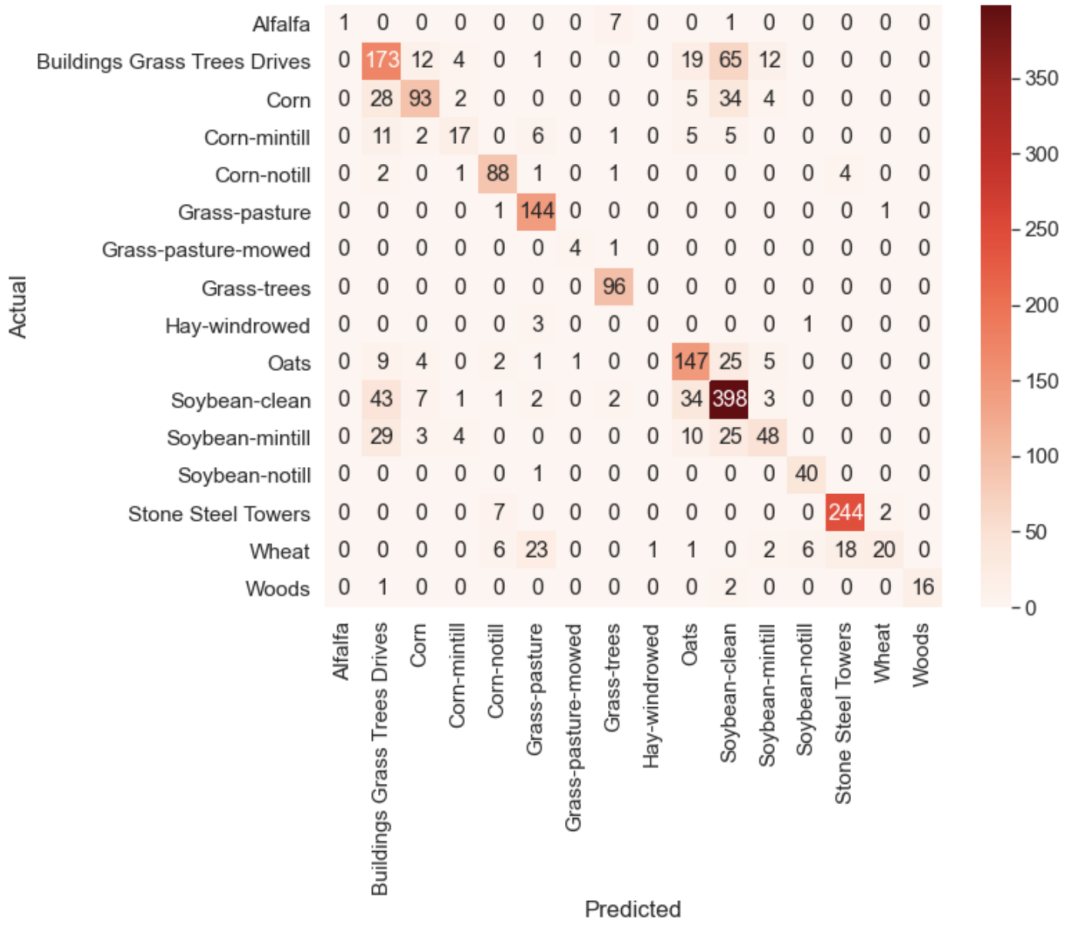


Figure 11: Prediction Stats

Results (9) SVM

We have fit SVM with three kernels- Linear, RBF and Polynomial Kernels.
Linear kernel did not converge easily and the results were not satisfactory. RBF and Polynomial kernel yield similar results. We show the results for RBF SVM.

	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

Figure 12: SVM Confusion Matrix

Results (10) SVM

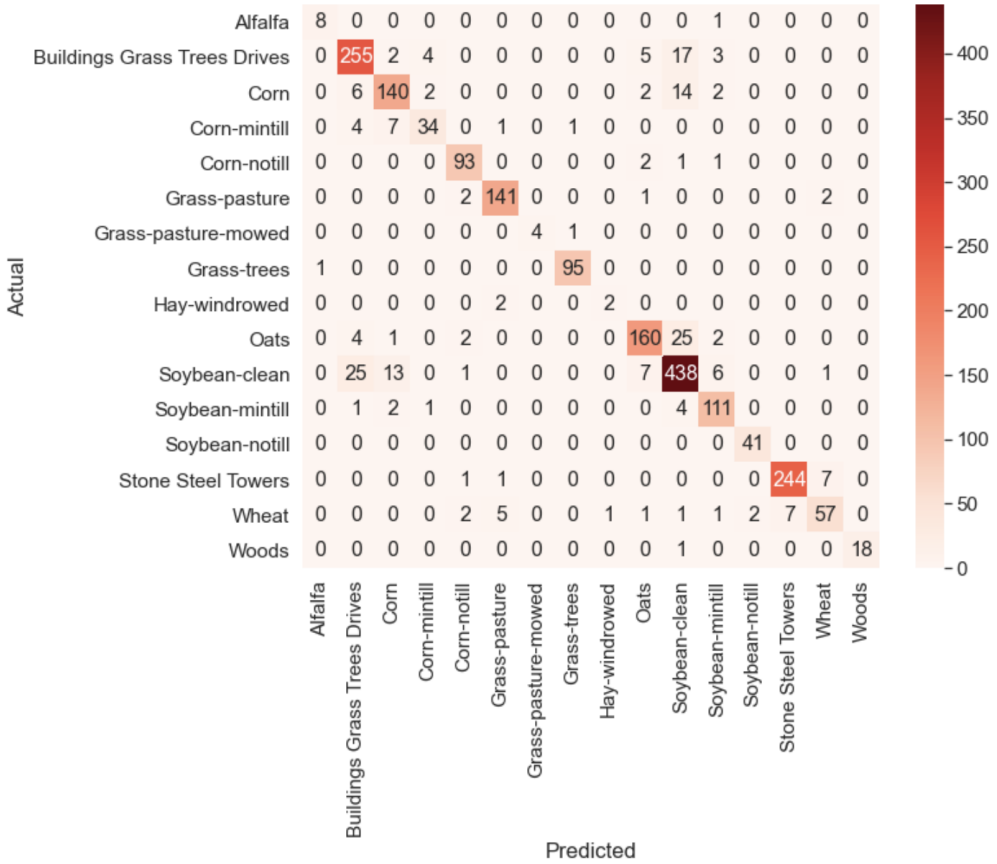


Figure 13: Prediction Stats

Results (10) Feed Forward NN

We use a shallow (1000,400) network which easily matches the SVM results and provides a promising avenue for further exploration

	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
Buildings 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

Figure 14: SVM Confusion Matrix

Results (11) Feed Forward NN

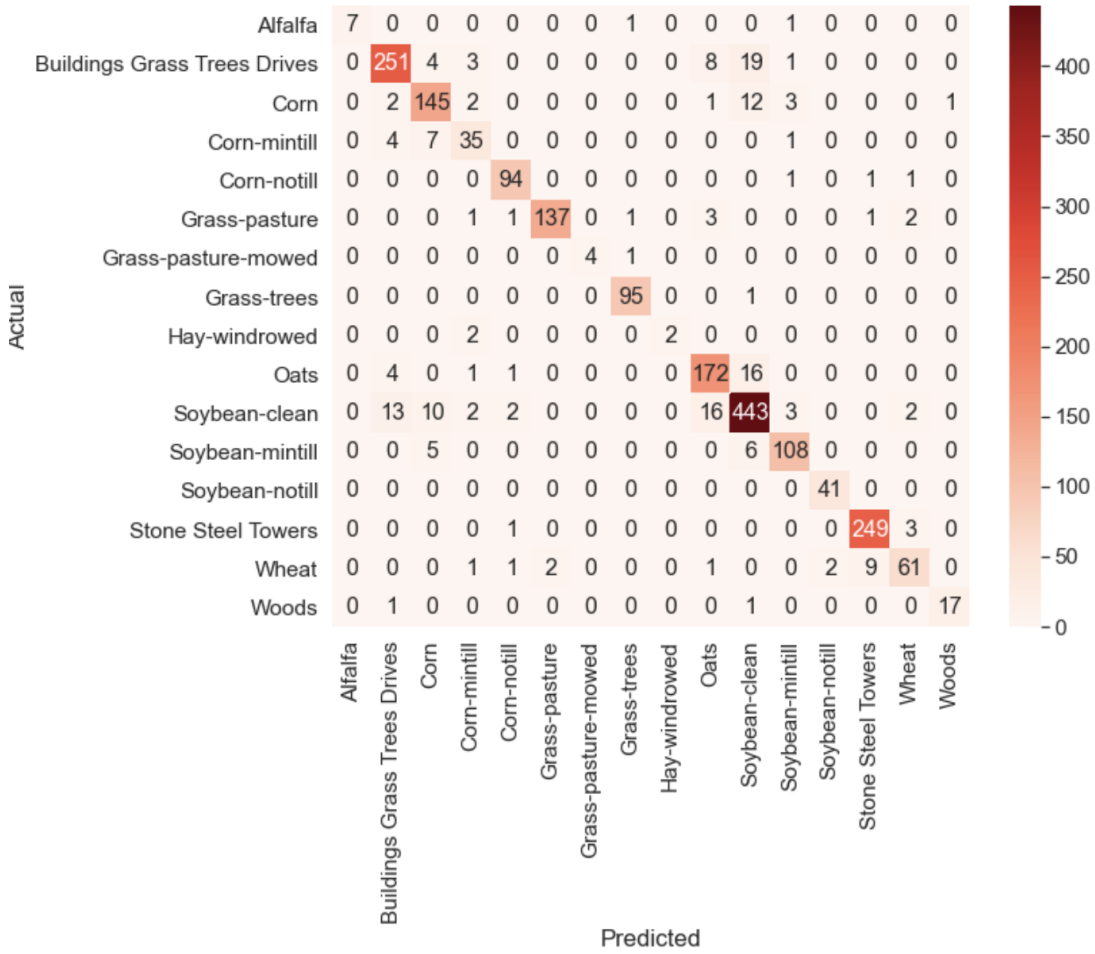


Figure 15: Prediction Stats

Conclusion

We have first formulated the problem for the case of spectral classification. Then we briefly visualized the dataset and then we moved on to the classification of existing methods.
We have seen that the NN and SVM go neck to neck in terms of performance. This forms the basis for further work in this area.

Future Work

- Implement and explore methods that exploit the spatial nature of the data. (RNN)
- Use deeper and more apt networks like CNNs to achieve better results.
- Compare results with the classical baseline.
- Extend the models to datasets like Pavia University and datasets.

How do we improve the accuracy?

- Up until now we could achieve around 90-92% accuracy for classical machine learning methods.
- We can improve it by using more information to discriminate between classes.
- The additional information is basically given by the neighbouring pixels.
- This can be seen intuitively too. While looking at images we often use neighbouring pixels to make a guess on the pixel in question.
- We can implement this in the following way. Let's say that we have a hyperspectral cube of size 145*145*100- cont'd

Earlier we flattened this data into 21025*100 cubes with each sample being of shape 1*100. Now, we take the neighbouring pixels with the pixel in question as the central pixel as one training example. For example, we can take a cube of size 5*5*100 as one training example here. The idea is shown in the figure-

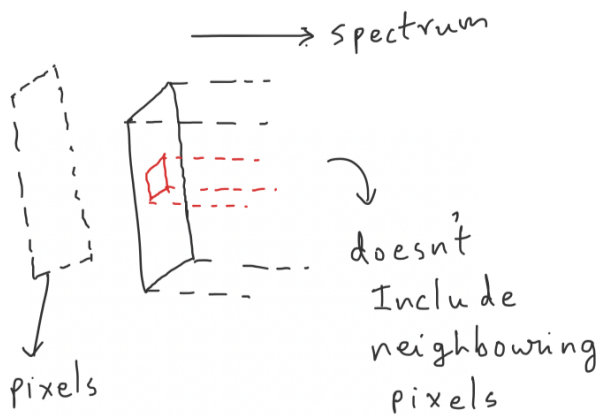


Figure 16: Our initial model of training example

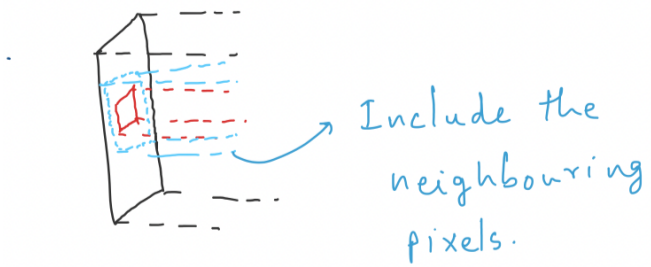


Figure 17: New Training Patch

we can see that the spatial information is also included. This adds to our hyperparameters set. We have to decide on the patch size.

Setting the Patch Size

- Having a **large patch** size might dilute the true essence of the pixel and hence not desirable.
- Having a **small patch** size is also not desirable because enough spatial information is not covered.
- The optimal patch size depends on the spatial frequency of variations. In cities were there change in almost every pixel, it might not be advisable to take a very large patch whereas that might be helpful in a rural setting.
- The resolution of the satellite also plays an important role in this.
- In theory, we can calculate $R(x, x + p)$ (Spatial Correlation, p stands for patch size) for various values of p and choose the patch size which will give good correlation.

Continued

- A more simpler approach would be to run a few experiments based on experience and intuition to decide on the patch size.
- We will show some results for patch sizes later.
- This now allows us to use CNN architectures to train the neural networks.
- CNN (Convolutional Neural Networks) approach is a filter based approach which we will explain in brief. Now that we have the spatial information too, This is called a **Spatial-Spectral** method.

- The basic idea of CNN is to use filters in the first few layers with trainable parameters to extract useful features from the image. The useful features are then fed to the ANN layers which we have used before. This has a dual advantage.
- Common elements- 1. Convolution layer (This layer is the filter area).
- 2. Pooling
- 3. Fully Connected layer (ANN)
- We will explain them briefly.

As discussed earlier convolutional layer is just a filter applied on the 3 dimensional cube. There will be filters which traverse across the image as shown in the figure below-

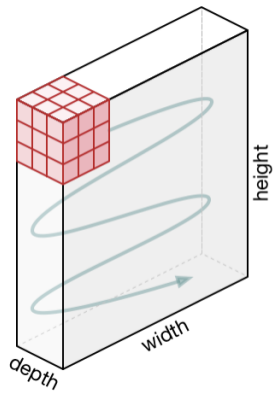


Figure 18: The filter Movement. How far it moves in a given step is given by the **stride**

As an example we can check the following convolution-

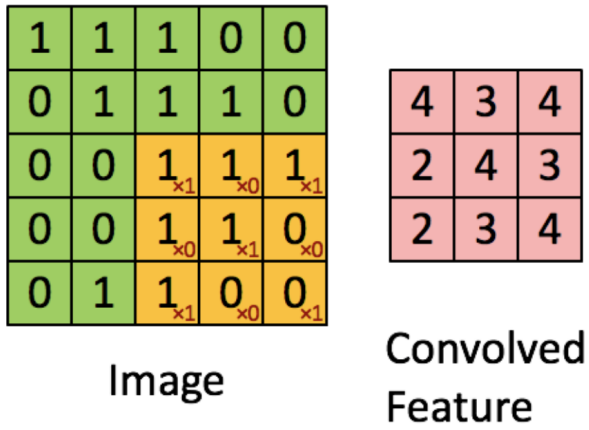
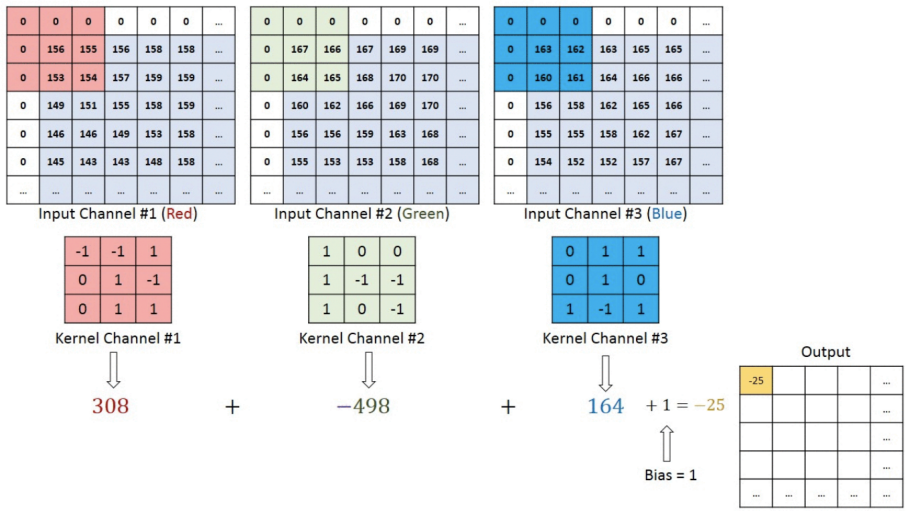


Figure 19: Convolution Output

Convolution Output depends on-

- Stride (How far we would like our filter to move across the image?)
- Padding (Whether we choose to pad the edge pixels or not?)

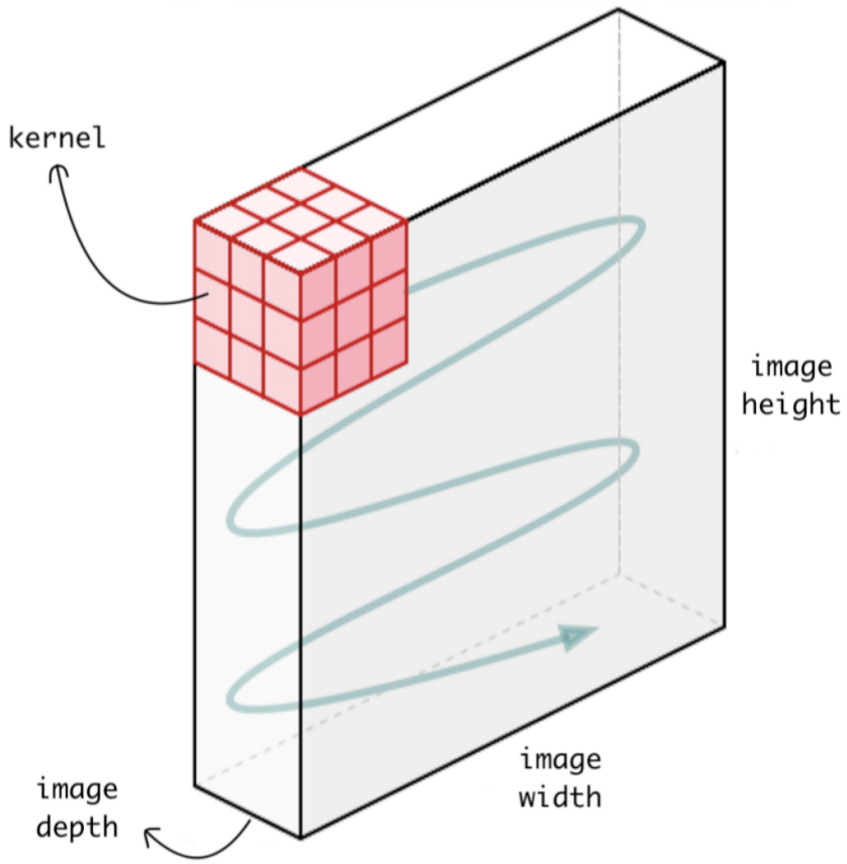
Based on this we have the size of the output from the filter. If there are multiple channels like in the case of RGB and Hyperspectral cases, we add the channel outputs to get an aggregated picture. An example of this is shown in the figure below-



Convolution operation on a MxNx3 image matrix with a 3×3×3 Kernel

Aggregation of the Bands of the convolution of an RGB image. Same can be done for the hyperspectral data.

Convolutional Layers in 2 dimensions. The kernel moves in two directions (2 degrees of freedom). The representation is shown below-



Conv3D

We can similarly look at what happens in the case of 3d Kernels. Here, there is an opportunity to get more fine grained features which might help in the classification of edge cases. The visualization of 3DConv is shown in the next figure-

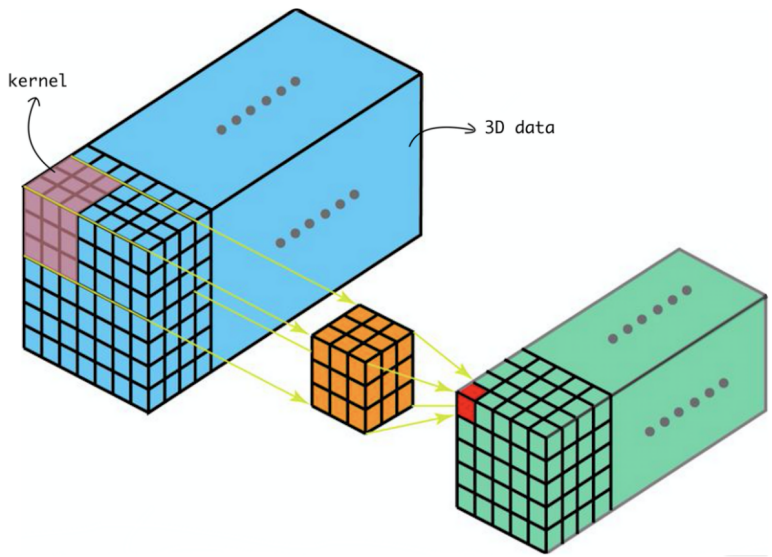


Figure 21: Conv3D visualization

Pooling helps reduce the spatial dimension of the data. Common types are max pooling and average pooling. Max pooling is generally more widely used. An example is shown in the figure below-

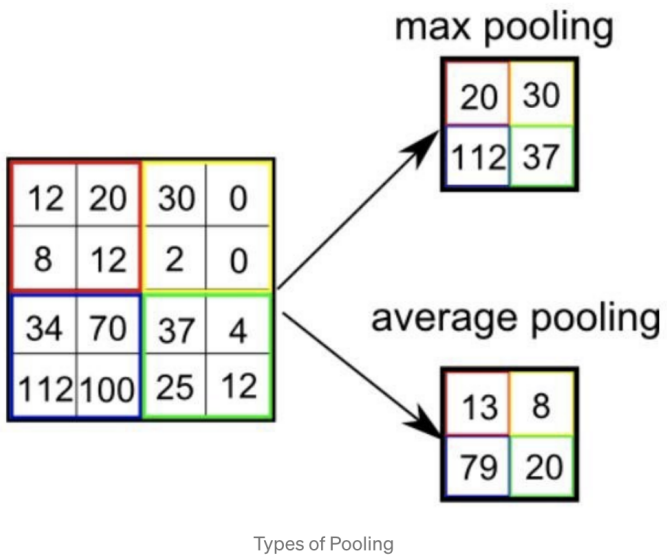


Figure 22: Pooling is used to reduce the spatial dimension

ANNs are standard NN appended to the pooled layer. At the end we have a softmax layer for multi-class classification.
A CNN architecture describes the layers and the number of units in each layer along with the padding and strides for each convolutional layer and pooling layer.
With this knowledge we can now describe the models which have been proposed over the years.

Hamida et al proposed the following architecture (The patch size used is both 3 and 5)-

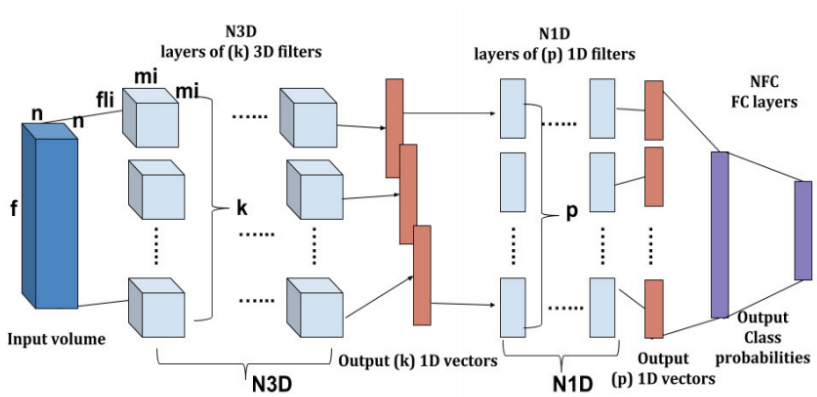


Figure 23: Overview of the architecture

They have experimented with patch sizes 3×3 (too small), 5×5 and 7×7(too large, dilutes the result). They found out that 5×5 is the optimal patch size as we have seen earlier. They used 3D convolutions instead of 2D convolutions which increase the expressivity of the model.
The figure below summarizes their experiments and results in Pavia University scene-

Architecture	Iterat-ions Nb	Processing Time (s)	Accuracy (early)	final Accu-racy (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 24: Experiments and Their Results

Lee and Kwon took a more novel approach, they have take spatial patches of size 3×3, 5×5 and 7×7 as the input and used a modified alex net to arrive at an architecture. The idea is shown below-

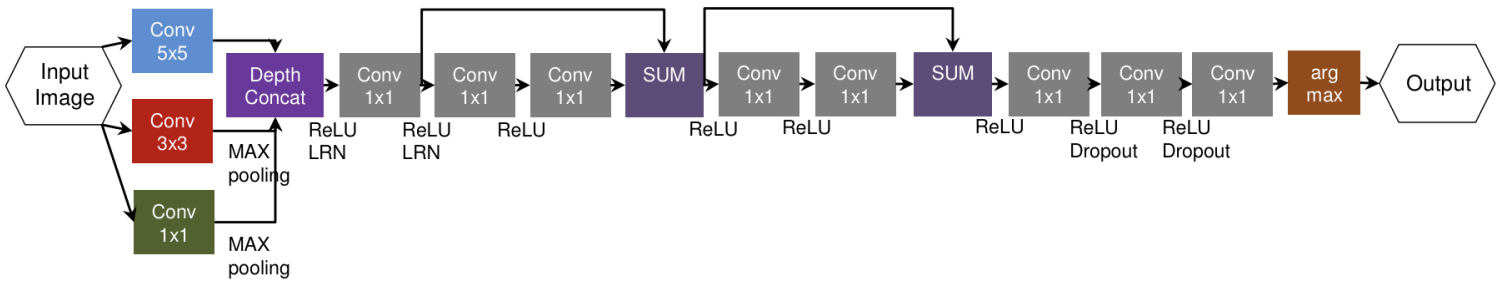


Figure 25: Modified Alex Net used by Lee and Kwon

The results obtained are summarized below-

Method	Indian Pines
Two-layer NN [2]	86.49
RBF-SVM [2]	87.60
Three-layer NN [1], [2]	87.93
LeNet-5 [2], [15]	88.27
Shallower CNN [2]	90.16
D-DBN [6]	91.03 ± 0.12
The proposed network	**93.61 ± 0.56 (94.24)**

Figure 26: Results Obtained by Lee and Kwon

Other approaches have a similar idea of taking a patch and then using a 3d convolutional networks for inference. All yield a good accuracy. The results were summarized in the paper "Deep Learning for Classification of Hyperspectral Data: A Comparative Review" and shown below-

Dataset	Indian Pines	
Model	Random	Disjoint
Nearest-neighbor	75.63	67.27
1D CNN [42] (original)	90.16	-
1D CNN (ours)	89.34	82.99
RNN [43] (original)	85.7	-
RNN (ours)	79.70	62.23
2D+1D CNN [63] (original)	-	-
2D+1D CNN (ours)	-	-
3D CNN [68] (original)	99.07	-
3D CNN (ours)	96.87	75.47

Figure 27: Results of implementations of recent approaches

Continued

We now turn our attention to the current State of The art method. We have implemented this on our own to get a grip on the implementation. All other approaches can be implemented very easily since they all belong to the same class and variations of each other.

Current SOTA: Hybrid SN

The current state of the art method combines 3d and 2d convolutions. The architecture is shown below. The main difference between this and other approaches is the patch size used. The patch size is 25 which as we discussed is high. The network is big enough to discount pixels that don't count so it actually performs better.

SOTA: Architecture

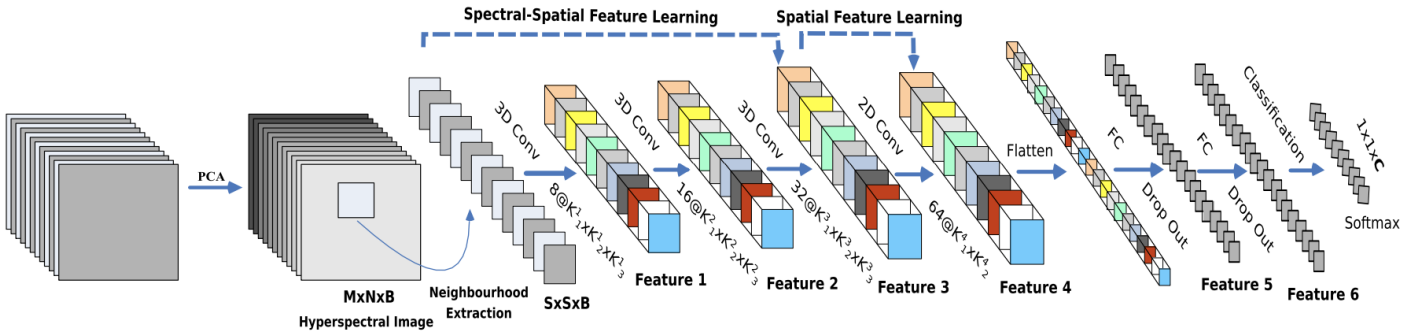


Figure 28: Hybrid SSN architecture

Summary of the Literature in this area

Initial models proposed used 1D and 2D CNN which yielded aroundd 88-90% accuracy.
The recent models used 3D convolutions and the fully connected layers.
The SOTA has a hybrid (combination) of 3D and 2D convolutions and fully connected layers to achieve the top performance.
In total we surveyed around 6-7 papers and this is what we found as a result of that.

Our Implementations

We have implemented the current Hybrid Model along with our own architecture based on Resnet which we will describe in a moment.

Implementation Details SOTA

- The implementation uses 3D convolutions which we have used earlier.
- We have used a patch size of 25 along.
- We used a 70/30 data split.
- The library used to implement is Keras. The model summary is shown in the next slide.
- 20 epochs used to train. Around 0.5 million parameters.

Model Summary

Layer (type)	Output Shape	Param #
input_7 (InputLayer)	[(None, 25, 25, 30, 1)]	0
conv3d_10 (Conv3D)	(None, 23, 23, 24, 8)	512
conv3d_11 (Conv3D)	(None, 21, 21, 20, 16)	5776
conv3d_12 (Conv3D)	(None, 19, 19, 18, 32)	13856
reshape_5 (Reshape)	(None, 19, 19, 576)	0
conv2d_5 (Conv2D)	(None, 17, 17, 64)	331840
flatten_5 (Flatten)	(None, 18496)	0
dense_12 (Dense)	(None, 256)	4735232
dropout_7 (Dropout)	(None, 256)	0
dense_13 (Dense)	(None, 128)	32896
dropout_8 (Dropout)	(None, 128)	0
dense_14 (Dense)	(None, 16)	2064

Total params: 5,122,176
Trainable params: 5,122,176
Non-trainable params: 0

Figure 29: Model Summary

Classification Reports

```
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

[[  32    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0]
 [   0  994    0    0    0    0    0    0    0    1    4    0    0    1
    0    0]
 [   0    0  581    0    0    0    0    0    0    0    0    0    0    0
    0    0]
 [   0    0    0  166    0    0    0    0    0    0    0    0    0    0
    0    0]
 [   0    0    0    0  335    0    0    0    0    2    0    0    1    0
    0    0]
 [   0    1    0    0    0  510    0    0    0    0    0    0    0    0
    0    0]
 [   0    0    0    0    0    0   20    0    0    0    0    0    0    0
    0    0]
 [   0    0    0    0    0    0    0  335    0    0    0    0    0    0
    0    0]
 [   0    0    0    0    0    0    0    0   14    0    0    0    0    0
    0    0]
 [   0    0    0    0    0    0    0    0    0  680    0    0    0    0
    0    0]
 [   0    2    0    0    0    1    0    0    0    0 1715    0    0    0
    1    0]
 [   0    1    0    1    0    0    0    0    0    0    0  412    0    0
    0    1]
 [   0    0    0    0    1    0    0    0    0    0    0    0  142    0
    0    0]
 [   0    0    0    0    0    0    0    0    0    0    1    0    0  885
    0    0]
 [   0    4    0    0    0    0    0    0    0    0    0    0    0    0
  266    0]
```

Results(1): SOTA

We have achieved around 99+% accuracy on our implementation of the SOTA architecture. The predicted labels are shown below-

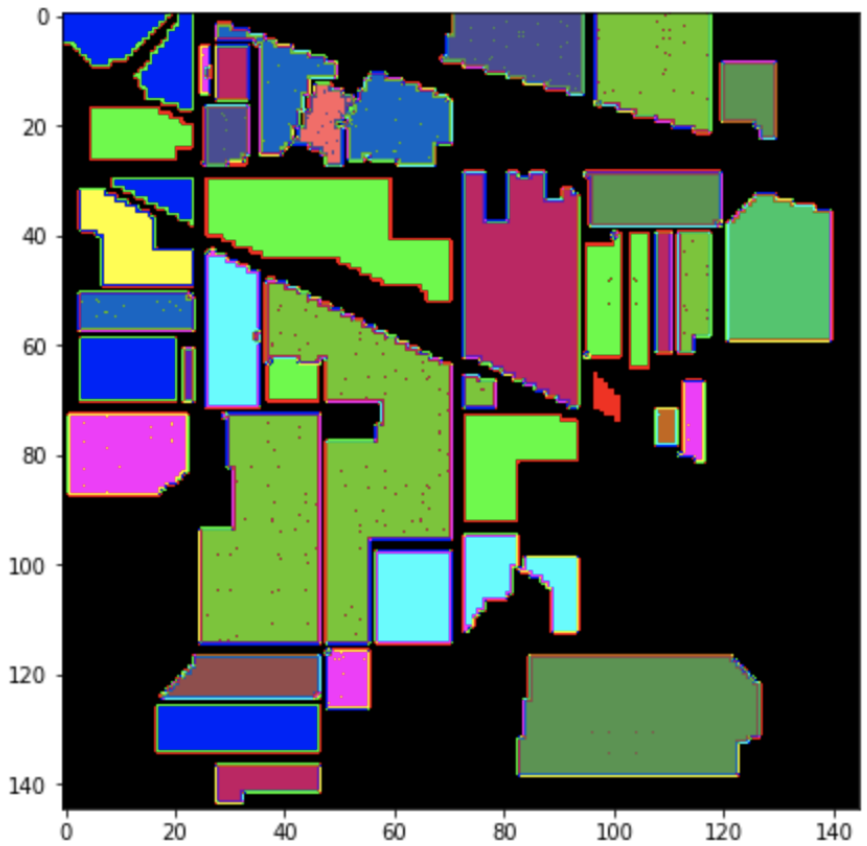


Figure 31: Predicted Labels

New Architecture Proposed

We now propose an architecture which helps us go very deep without losing accuracy. We propose the use of Resnets in this particular case. Resnets have identity blocks and skip connections which allow us to go very deep.
How is this useful? It is very easy to see that deeper networks need lesser number of parameters than shallow networks to have the **same expressive power**.
This forms the basis of our investigation. We used Resnets with depths around 16 layers to train the neural network.
It has comparable performance to the SOTA model but has fewer parameters and hence easier to train.
A model of our Resnet is shown in the next slide-

ResNet Model

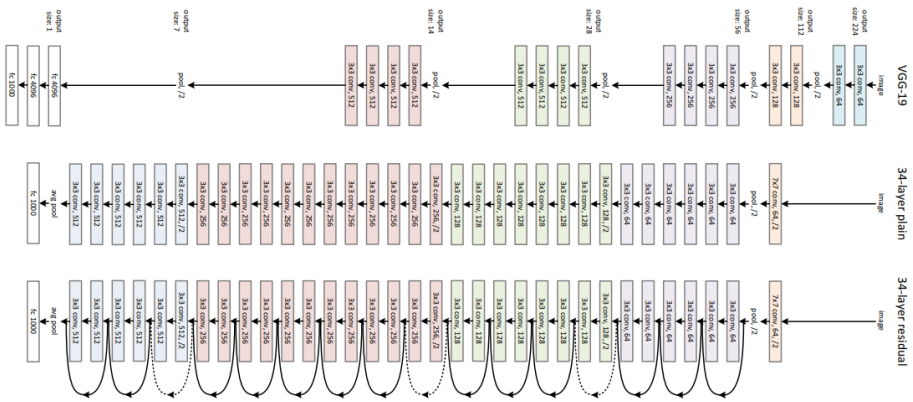


Figure 32: Original ResNet Proposed, We used a stripped down 18 layer version

Results of Our Experiments

The model has 16 Convolutional Layers along with an input and a fully connected output layer.
We used the same standardized hyper parameter values.
After extensive experimentation, we found that the patch size of 21×21 is the **optimal patch size** for our model.
The classification output and results are shown in the following slides.

Results: Classification Report

	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 33: Classification Report

Comparision: Ground and Predictions

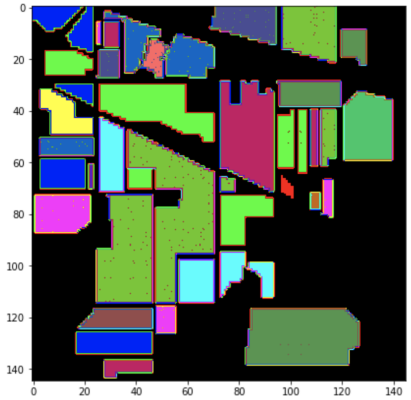


Figure 34:

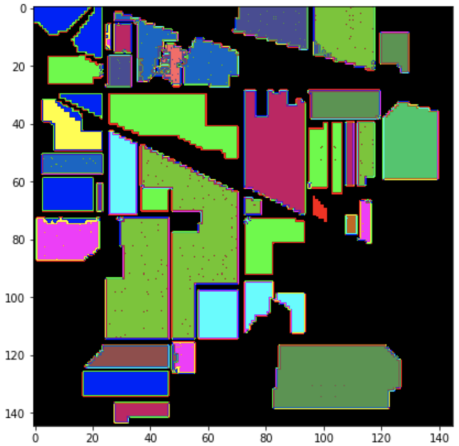


Figure 35: Predicted Labels

Advantages of our approach

- We have unlocked the potential to go deeper given more training data.
- Our model has approximately 0.2 Million parameters whereas the state of the art model has 5 Million parameters. This can be attributed to the deep layer expressiveness that ResNets bring to the table.
- We have, similar to the SOTA model, incorporated both the **spatial and spectral features** in turn increasing the accuracy.

Limitations and Future Improvements of Our Model

- We have successfully implemented our idea but we have only used 2D Convolutions in our model. This means that we leave some combined Spatio-Spectral Features which might help in differentiating the edge samples and further increase accuracy.
- This points towards a future improvement which is to use ResNets with 3D convolutions. This could help us overtake the SOTA model because we can go much deeper.
- While we had comparable performance when matched against the SOTA, we didn't completely match the performance ($1-2\%$ difference). This can be understood as the tradeoff for having much lower number of parameters.

- *Hyperspectral Remote Sensing Scenes*. Grupo de Inteligencia Computacional (GIC) Buscar
- *Scikit Learn Multi class Classification*. `https://scikit-learn.org/stable/modules/multiclass.html`
- *Deep Learning for Classification of Hyperspectral Data: A Comparative Review*. Audebert et al.
- `https://towardsdatascience.com/hyperspectral-image-analysis-classification-c41f69ac447f`
- Deep Convolutional Neural Networks for Hyperspectral Image Classification, Hu et al., Journal of Sensors 2015
- Hyperspectral CNN for Image Classification Band Selection, with Application to Face Recognition, Sharma et al, technical report 2018
- 3-D Deep Learning Approach for Remote Sensing Image Classification, Hamida et al., TGRS 2018

- Contextual Deep CNN Based Hyperspectral Classification, Lee and Kwon, IGARSS 2016
- HybridSN: Exploring 3-D–2-D CNN Feature Hierarchy for Hyperspectral Image Classification