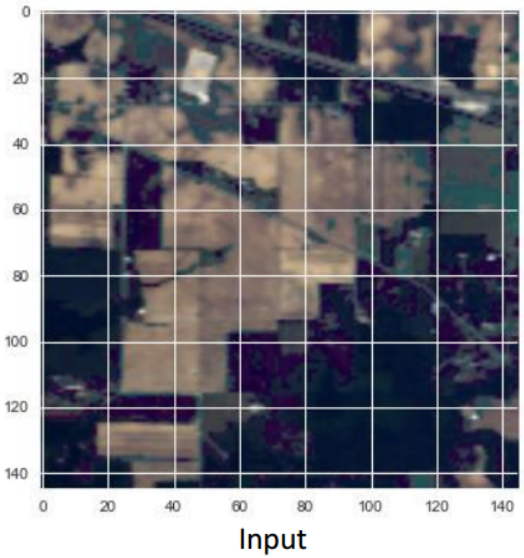
Hyperspectral

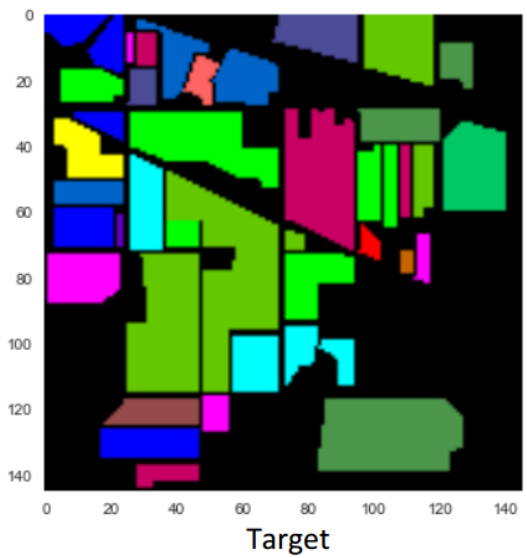
Deep Learning for Land-cover Classification in Hyperspectral Images.

Deep Learning for Land-cover Classification in Hyperspectral Images

PS: Check out our latest work in which we carefully design a novel deep neural network architecture that simultaneously addresses the three biggest challenges of Hyperspectral Image Classification - large dimensionality, spatial variability of spectral signatures and scarcity of labeled data. It also beats the state-of-the-art performance on three popular datasets and converges significantly faster. Our code is available here.

Hyperspectral images are images captured in multiple bands of the electromagnetic spectrum. This project is focussed at the development of Deep Learned Artificial Neural Networks for robust landcover classification in hyperspectral images. Land-cover classification is the task of assigning to every pixel, a class label that represents the type of land-cover present in the location of the pixel. It is an image segmentation/scene labeling task. The following diagram describes the task.





This website describes our explorations with the performance of Multi-Layer Perceptrons and Convolutional Neural Networks at the task of Land-cover Classification in Hyperspectral Images. Currently we perform pixel-wise classification.

Dataset

We have performed our experiments on the Indian Pines Dataset. The following are the particulars of the dataset:

- Source: AVIRIS sensor
- Region: Indian Pines test site over north-western Indiana
- Time of the year: June
- Wavelength range: 0.4 – 2.5 micron
- Number of spectral bands: 220
- Size of image: 145x145 pixel
- Number of land-cover classes: 16

Input data format

Each pixel is described by an `NxN` patch centered at the pixel. `N` denotes the size of spatial context used for making the inference about a given pixel.

The input data was divided into training set (75%) and a test set (25%).

Hardware used

The neural networks were trained on a machine with dual Intel Xeon E5-2630 v2 CPUs, 32 GB RAM and NVIDIA Tesla K-20C GPU.

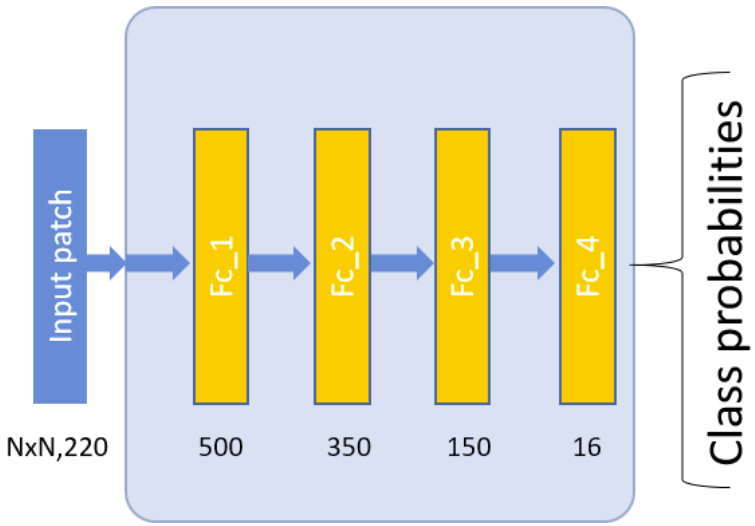
Multi-Layer Perceptron

Multi-Layer Perceptron (MLP) is an artificial neural network with one or more hidden layers of neurons. MLP is capable of modelling highly non-linear functions between the input and output and forms the basis of Deep-learning Neural Network (DNN) models.

Architecture of Multi-Layer Perceptron used

input- [affine - relu] x 3 - affine - softmax

(Schematic representation below)



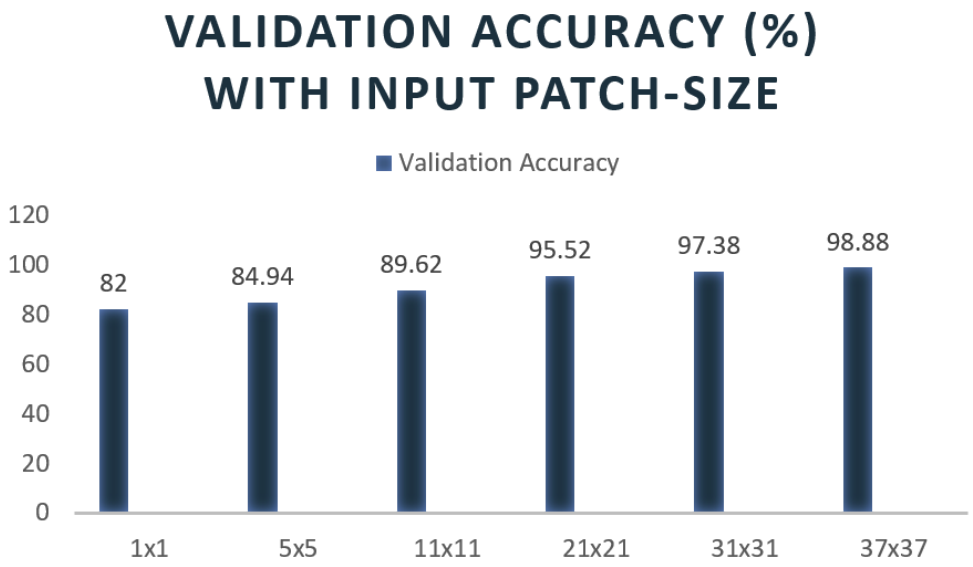
N denotes the size of the input patch.

Specifics of the learning algorithm

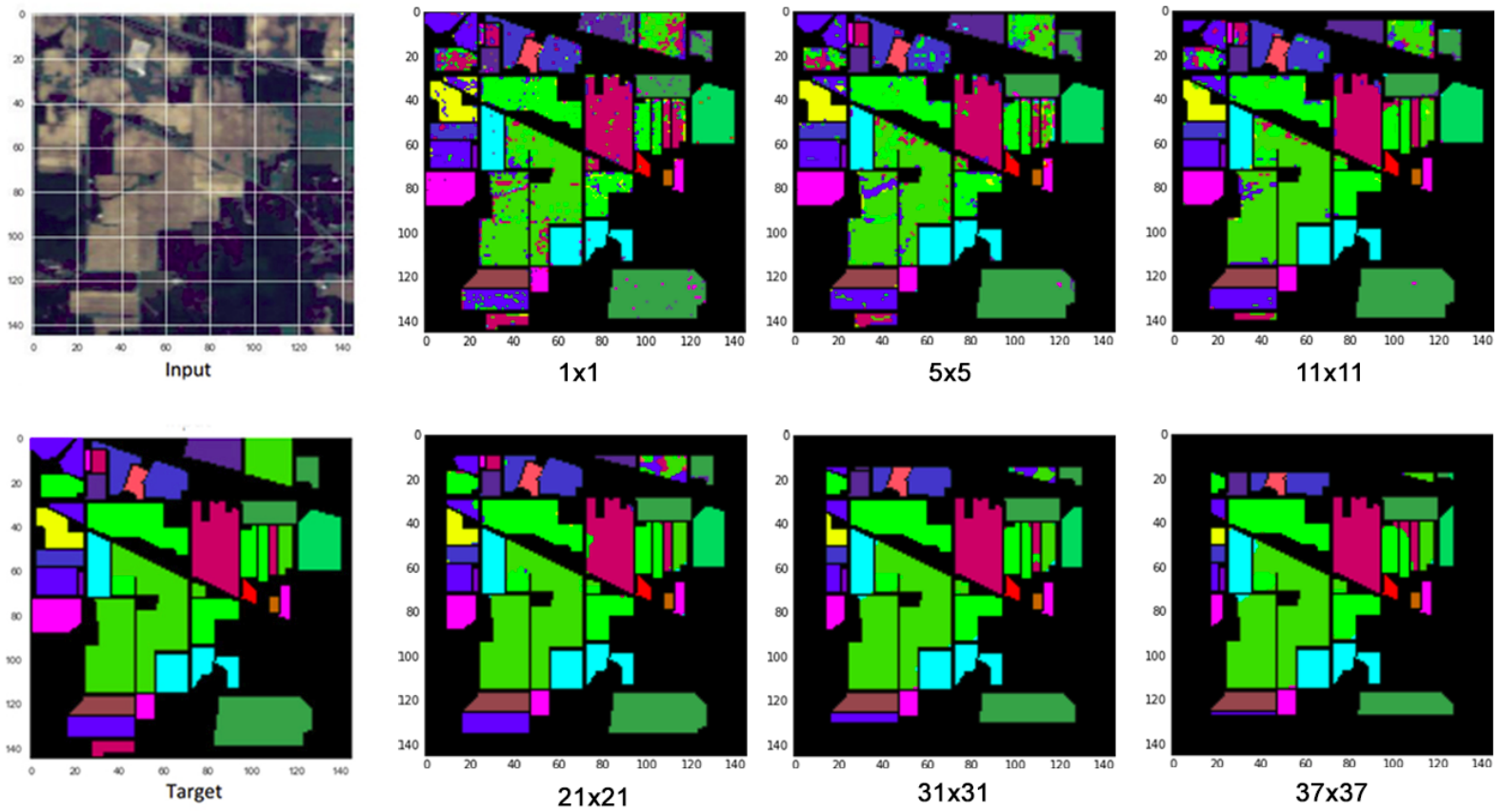
The following are the details of the learning algorithm used:

- Parameter update algorithm used: Adagrad
  - Batch size: 200
  - Learning rate: 0.01
- Number of steps: until best validation performance

Performance



Decoding generated for different input patch sizes:



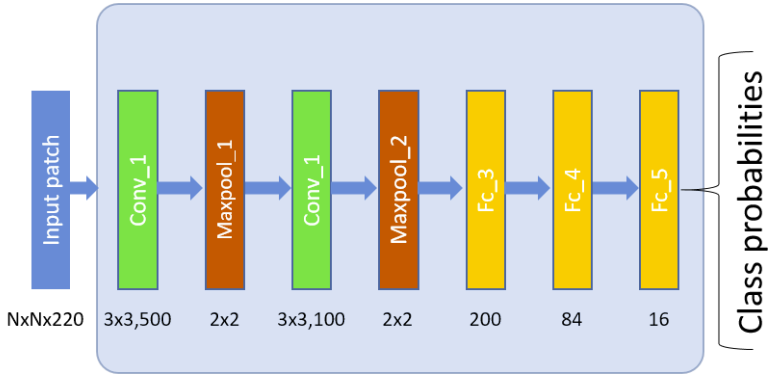
Convolutional Neural Network

(CNN or ConvNet) are a special category of artificial neural networks designed for processing data with a gridlike structure. The ConvNet architecture is based on sparse interactions and parameter sharing and is highly effective for efficient learning of spatial invariances in images. There are four kinds of layers in a typical ConvNet architecture: convolutional (conv), pooling (pool), fullyconnected (affine) and rectifying linear unit (ReLU). Each convolutional layer transforms one set of feature maps into another set of feature maps by convolution with a set of filters.

Architecture of Convolutional Neural Network used

input- [conv - relu - maxpool] x 2 - [affine - relu] x 2 - affine - softmax

(Schematic representation below)



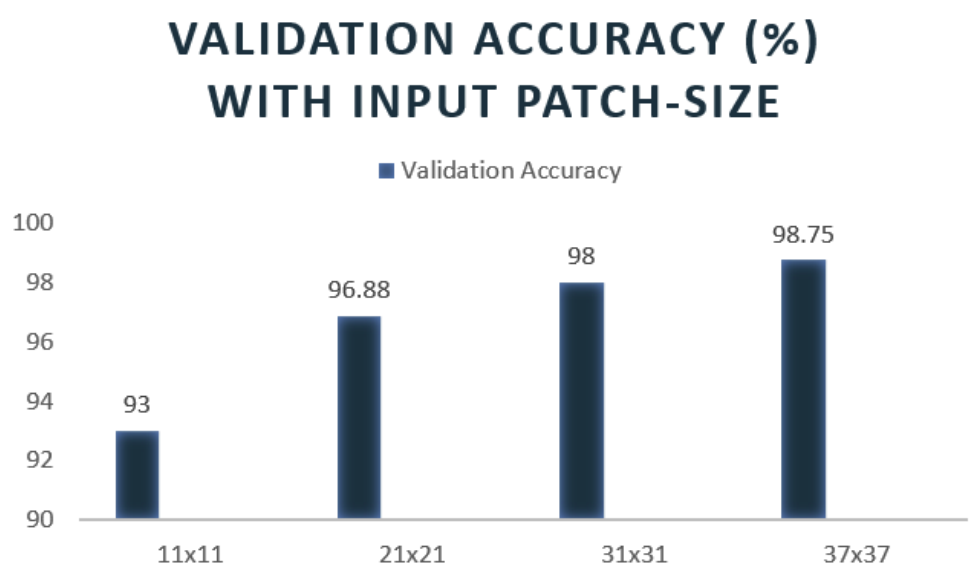
N denotes the size of the input patch.

Specifics of the learning algorithm

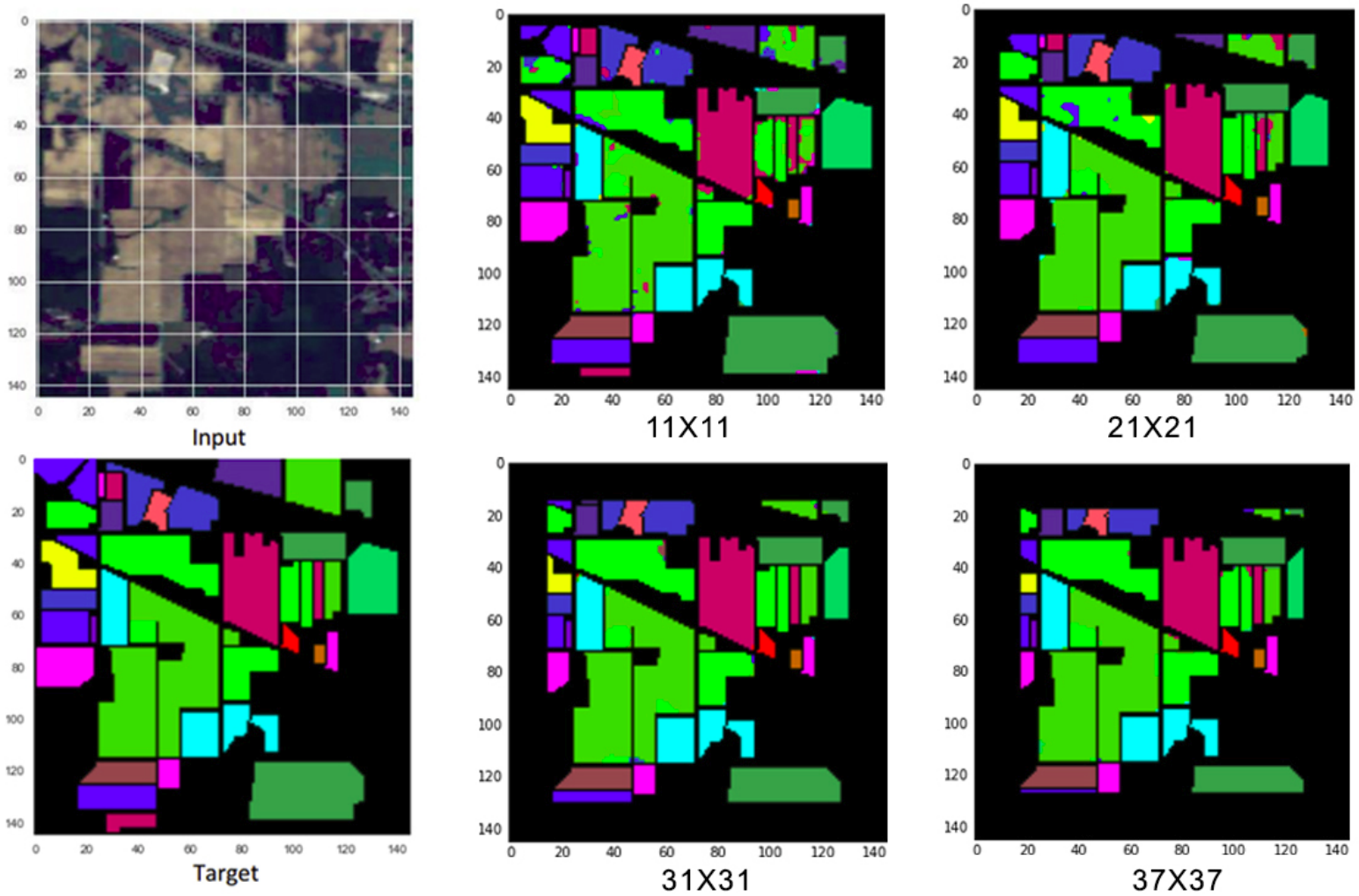
The following are the details of the learning algorithm used:

- Parameter update algorithm used: Adagrad
  - Batch size: 100
  - Learning rate: 0.01
- Number of steps: until best validation performance

Performance



Decoding generated for different input patch sizes:



Description of the repository

- `IndianPines_DataSet_Preparation_Without_Augmentation.ipynb` - does the following operations:

    - Loads the Indian Pines dataset
    - Scales the input between [0,1]
    - Mean normalizes the channels
    - Makes training and test splits
    - Extracts patches of given size
    - Oversamples the training set for balancing the classes
- `Spatial_dataset.py` - provides a highly flexible Dataset class for handling the Indian Pines data.
- `patch_size.py` - specify the required patch-size here.
- `IndianPinesCNN.ipynb` - builds the TensorFlow Convolutional Neural Network and defines the training and evaluation ops:
    - `inference()` - builds the model as far as is required for running the network forward to make predictions.
    - `loss()` - adds to the inference model the layers required to generate loss.
    - `training()` - adds to the loss model the Ops required to generate and apply gradients.
    - `evaluation()` - calcuates the classification accuracy
- `CNN_feed.ipynb` - trains and evaluates the Neural Network using a feed dictionary
- `Decoder_Spatial_CNN.ipynb` - generates the landcover classification of an input hyperspectral image for a given trained network
- `IndianPinesMLP.py` - builds the TensorFlow Multi-layer Perceptron and defines the training and evaluation ops:
    - `inference()` - builds the model as far as is required for running the network forward to make predictions.
    - `loss()` - adds to the inference model the layers required to generate loss.
    - `training()` - adds to the loss model the Ops required to generate and apply gradients.
    - `evaluation()` - calcuates the classification accuracy
- `MLP_feed.ipynb` - trains and evaluates the MLP using a feed dictionary
- `Decoder_Spatial_MLP.ipynb` - generates the landcover classification of an input hyperspectral image for a given trained network

- `credibility.ipynb` - summarizes the predictions of an ensemble and produces the land-cover classification and class-wise confusion matrix.

Setting up the experiment

- Download the Indian Pines data-set from here.
- Make a directory named `Data` within the current working directory and copy the downloaded .mat files `Indian_pines.mat` and `Indian_pines_gt.mat` in this directory.

In order to make sure all codes run smoothly, you should have the following directory subtree structure under your current working directory:

```
|-- IndianPines_DataSet_Preparation_Without_Augmentation.ipynb
|-- Decoder_Spatial_CNN.ipynb
|-- Decoder_Spatial_MLP.ipynb
|-- IndianPinesCNN.ipynb
|-- CNN_feed.ipynb
|-- MLP_feed.ipynb
|-- credibility.ipynb
|-- IndianPinesCNN.py
|-- IndianPinesMLP.py
|-- Spatial_dataset.py
|-- patch_size.py
|-- Data
|   |-- Indian_pines_gt.mat
|   |-- Indian_pines.mat
```

- Set the required `patch-size` value (eg. 11, 21, etc) in `patch_size.py` and run the following notebooks in order:

1. `IndianPines_DataSet_Preparation_Without_Augmentation.ipynb`
2. `CNN_feed.ipynb` OR `MLP_feed.ipynb` (specify the number of fragments in the training and test data in the variables `TRAIN_FILES` and `TEST_FILES`)
3. `Decoder_Spatial_CNN.ipynb` OR `Decoder_Spatial_MLP.ipynb` (set the required checkpoint to be used for decoding in the `model_name` variable)

Outputs will be displayed in the notebooks.

Acknowledgement

This repository was developed by Anirban Santara, Ankit Singh, Pranoot Hatwar and Kaustubh Mani under the supervision of Prof. Pabitra Mitra during June-July, 2016 at the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. The project is funded by Satellite Applications Centre, Indian Space Research Organization (SAC-ISRO).

Hyperspectral is maintained by KGPML.
This page was generated by GitHub Pages.