

## ABSTARCT

Soil moisture is an important variable in the climate system. Understanding and predicting variations of surface temperature, precipitation, drought, flood, and the impacts of future climate change depend critically on knowledge of soil moisture variations. NASA's Soil Moisture Active Passive (SMAP) satellite measures soil moisture with two instruments-a Radar instrument and a Microwave instrument. The Radar and the Radiometer spectral bands capture soil moisture observations at spatial coverages of  $3\text{km}^2$  and  $36\text{km}^2$  respectively. A couple of months into the satellite's launch in 2015, the Radar with its spectral band stopped functioning while the Radiometer continued to function. As a result, the soil moisture observations at a spatial coverage of  $3\text{km}^2$  at various locations across the planet are no longer captured. Accessing the datasets when both instruments were working, helped us identify images of soil moisture captured by both the Radar instrument with higher resolution compared to the images of the dataset captured by the Radiometer. In this project, we focus on recovering the  $3\text{km}^2$  dataset with the help of the  $36\text{km}^2$  dataset. To perform this experiment, we employed a Convolutional Neural Network approach which trains a 2D- set of  $36\text{km}^2$  images to learn the features of a  $3\text{km}^2$  image. In other words, we implemented the idea of a Super-Resolution-Convolutional Neural-Network approach (SRCNN) which helped us convert a low-resolution image to high-resolution.

## CHAPTER-1

### INTRODUCTION

Soil moisture is a key variable constraining the fluxes between the land surface and atmosphere boundary layer, therefore it plays a key role in regulating the feedbacks between the terrestrial water, carbon and energy cycles. The soil moisture content of soil is the quantity of water it contains. Water content is used in a wide range of scientific and technical areas and is expressed as a ratio, which has a range between 0 (totally dry) and a higher numerical value, usually 1 (totally saturated). Moisture is absorbed at internal surfaces and as capillary condensed water in small pores. In climate science, hydrology, and the agricultural sciences, water content plays an important role for ground recharge, agricultural and soil chemistry. Therefore, there is a need to characterize soil moisture at spatial scales relevant to the representation of land-surface and mesoscale processes in the atmosphere. This can potentially improve representation of evapotranspiration, runoff and precipitation in hydrologic and weather prediction models and result in improved predictive skills. In addition, knowledge of soil moisture at fine spatial scales is necessary to improve farming practices and optimizing irrigation scheduling. Soil moisture spatial variability is regulated by several factors including but not limited to precipitation, soil texture, surface vegetation and topography. A combination of these factors results in high spatial heterogeneity for soil moisture. Current global soil moisture estimates are available at coarse spatial scales (between  $3\text{km}^2$  and  $36\text{km}^2$ ) which limits their suitability for applications such as evapotranspiration modeling, particularly in regions with high atmospheric convection as well as farm Hydrology. Here, unfortunately the Radar instrument stopped functioning after 3 months. However, due to the failure of SMAP Radar instrument in July 2015, only the radiometer was available. Our goal is to see if we can recover the radar data from the radiometer data using Machine Learning. Our approach in this study is to use Neural Networks (NN) to develop a technique to recover highly coarse datasets of the Radar spectral instrument of the SMAP satellite with the help of lesser coarse dataset of the Radiometer instrument. The capability of NNs in learning complex relationships between inputs and target data as well as their quick run-time after training are among the reasons for their popularity in Earth science and remote sensing problems. In recent years, NN has been used to develop soil moisture retrieval algorithms from either passive or active instruments or combination of both.

## CHAPTER-2

### RELATED WORK

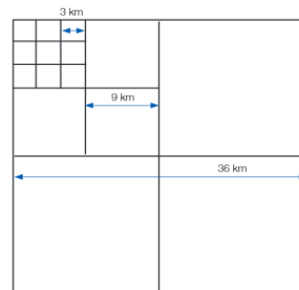
Konstantina et al., 2018[12] formulated the problem of SMAP active measurement reconstruction as an inverse imaging problem, considering as input the passive (radiometer) image and seeking to recover its corresponding active (Radar) observation. The recovery of SMAP's active measurements was implemented by capitalizing on the Sparse Representations framework. To implement this technique, the authors extracted image patches from the active images and represented them as a linear combination of signals encoded in a dictionary matrix that was constructed from active image patches. Instead of observing the active patches directly, the authors observed the corresponding passive image patches that coincided with the active patches. Patches that were extracted were directly mapped onto a dictionary matrix that was constructed by training elements selected from the corresponding passive images. In this paper, the authors have worked on SMAP's Level-2 (L2) dataset to reconstruct the 3Km<sup>2</sup> incomplete dataset with the help of the 36Km<sup>2</sup> dataset using the Coupled Dictionary Learning technique. The results that were achieved in this paper showed that the authors were able to reconstruct the corresponding active measurement from the passive observation with great efficiency. In terms of the PSNR evaluation error metric, the proposed technique assisted the authors to reconstruct and recover data with a similarity index of 0.96, validating the high similarity of the recovered active observation with respect to the ground truth active measurement. They also compared the performance of their technique against the state-of-the-art K-SVD technique. The PSNR achieved by the authors with their technique resulted in an PSNR of 38.66 while the state-of-the-art technique resulted in a PSNR value of 38.66. This helped them show that their coupled dictionary technique was efficient and powerful in helping them reconstruct the 3Km<sup>2</sup> from 36Km<sup>2</sup>.

Seyed et al., 2018 [7] have developed a downscaling algorithm based on a Neural Network approach that uses the two-soil moisture estimates from Soil-Moisture Active-Passive at 36km and 9km for training, and then retrieves soil moisture at 2.25km spatial resolution using the 9km estimates from SMAP. The Neural Network relates the coarse scale soil moisture and Normalized Difference Vegetation Index (NDVI) estimates as well as the fine scale NDVI estimates as input to the fine scale soil moisture estimates as output. To do so, they have assumed that the scaling relationship between 36km and 9km soil moisture estimates is the same as the scaling relationship between 9km and 2.25km estimates. To discuss their results, the authors computed the correlation coefficient (R<sup>2</sup>) and the unbiased Root Mean Square Difference (ubRMSD) between the NN estimates and the target data for the training and validation data. They also compare the metrics aggregated across all the data and spatially at each pixel. They calculated the correlation coefficient (R<sup>2</sup>) which resulted in a value of ~0.98 which helped them show that their scheme was able to learn efficiently and was also able to identify data beyond whatever it was trained on..

## CHAPTER-3

### SOIL MOISTURE ACTIVE PASSIVE DATA SETS

The surface soil moisture layer (0-5 cm) in  $\text{cm}^3/\text{cm}^3$  is derived from brightness temperatures (TBs) and is output on a fixed global 36 km Equal-Area Scalable Earth (EASE)-Grid 2.0. Also included are brightness temperatures in Kelvin representing the weighted average of Level-1B brightness temperatures whose boresights fall within a 36 km EASE-Grid 2.0 cell. The grids are intended to be versatile formats that help in representing global-scale gridded data, including remotely sensed data. Data can be expressed as a digital array with one of many possible grid resolutions, which are defined in relation to one of four possible projections: Northern / Southern Hemispheres (Lambert's equal-area, azimuthal), temperate zones (cylindrical, equal-area), or global (cylindrical, equal-area). Datasets are available in HDF5 format. Data arrays are 1D. SMAP data sets are available on NASA's SMAP Data products website [12]. For our project, we downloaded SMAP's L2 datasets (HDF5) for the Radiometer and Radar Spectral bands from [16] and [17] respectively. To read these datasets, we used the HDFView 3.0.0 package. This package is available on [18]. For the 36K<sup>2</sup> dataset, each half-orbit file is approximately 3 MB. The daily data volume is approximately 42 MB. Coverage spans from 180°W to 180°E, and from approximately 85.044°N and 85.044°S for the global EASE-Grid 2.0 projection. The gap in coverage at both the North and South Pole, called a pole hole, has a radius of approximately 400 km. The swath width is approximately 1000 km, enabling nearly global coverage every two to three days. The native spatial resolution of the radiometer footprint is 36 km. Data are then gridded using the 36 km EASE-Grid 2.0 projection. These Level-2 data are provided on the global cylindrical EASE-Grid 2.0. Each grid cell has a nominal area of approximately 36km<sup>2</sup> regardless of longitude and latitude. EASE-Grid 2.0 has a flexible formulation. By adjusting a single scaling parameter, a family of multi-resolution grids that nest within one another can be generated. The nesting can be adjusted so that smaller grid cells can be tessellated to form larger grid cells. Each Level-2 half-orbit file spans approximately 49 minutes. For the 3K<sup>2</sup> dataset, each half-orbit file is approximately 59 MB. The daily data volume is approximately 1.5 GB. Coverage spans from 180°W to 180°E, and from approximately 85.044°N and 85.044°S. The gap in coverage at both the North and South Pole, called a pole hole, has a radius of approximately 400 km. The swath width is 1000 km, enabling nearly global coverage every three days. The native spatial resolution of the radar footprint is 1 km. Data are then gridded using the 3 km EASE-Grid 2.0 projection. Each Level-2 half-orbit file spans approximately 49 minutes.



**Fig 1. Perfect Nesting in EASE-Grid 2.0:** This feature of perfect nesting provides SMAP data products with a convenient common projection for both high-resolution radar observations and low-resolution radiometer observations, as well as for their derived geophysical products

## CHAPTER-4

### SOIL MOISTURE

Soil Moisture Active Passive (SMAP) data products have been recorded at different levels. The Level-1 (L1) dataset provides all raw data downlinked from the Soil Moisture Active Passive (SMAP) radar during 6:00 a.m. descending and 6:00 p.m. ascending half-orbit passes. The L1 product contains information about the spacecraft position and attitude, as well as the antenna azimuth position over the same time that the radar telemetry covers. In addition to this, it also provides calibrated, geolocated, time-ordered brightness temperatures acquired by the SMAP radiometer during 6:00 a.m. descending and 6:00 p.m. ascending half-orbit passes. In our project we worked on the L2 dataset. Level 2 products are geophysical retrievals of soil moisture on a fixed Earth grid based on Level 1 products and ancillary information; the Level 2 products are output on a half-orbit basis. Level 3 products are daily composites of Level 2 surface soil moisture and freeze/thaw state data. Level 4 products are model-derived value-added data products of surface and root zone soil moisture and carbon net ecosystem exchange that support key SMAP applications and more directly address the driving science questions. In our project, we worked with Level 2 soil moisture product. This product provides estimates of global land surface conditions retrieved by the Soil Moisture Active Passive (SMAP) passive microwave radiometer during 6:00 a.m. descending and 6:00 p.m. ascending half-orbit passes. SMAP L-band brightness temperatures are used to derive soil moisture data, which are then resampled to an Earth-fixed, global cylindrical 36 km Equal-Area Scalable Earth Grid, Version 2.0 (EASE-Grid 2.0). Level 2 products contain output from geophysical retrievals that are based on instrument data and appear in half orbit granules. There are three, L2 soil moisture products resulting from the radar and radiometer data streams. L2\_SM\_A is a high-resolution research-quality soil moisture product that is mostly based on the radar measurements and is posted at 3Km<sup>2</sup>. L2\_SM\_P is soil moisture derived from the radiometer brightness temperature measurements and is posted at 36km<sup>2</sup>.

The radar-only soil moisture (L2\_SM\_A) is a fine-resolution (3 km) soil moisture estimate derived from high-resolution radar backscatter data (L1C\_SO\_HiRes). This data set, along with water body and freeze/thaw flags generated from the radar data, is made available during data processing to the other products as input.

## CHAPTER-5

### PROPOSED APPROACH

- 1) We averaged the soil moisture measurements for different locations for a two-month period ranging from May 1<sup>st</sup>, 2015 to June 30<sup>th</sup>, 2015 for the Radar and Radiometer instruments and plotted the measurements on a geographical map. (Refer Fig.2 and Fig.3)
- 2) After observing the averaged global plots, we selected a region such that the soil moisture data for this location was captured by both instruments of the satellite. In our project, we selected the US-East Coast
- 3) We plotted the soil moisture measurements for individual days on which this region (region selected in Step (2)) was captured by both instruments at the same day and at the same time. (Refer Fig.4 to Fig.7)
- 4) We extracted the latitude, longitude and soil moisture data from the HDF5 files for both instruments for the specific date and time on which both instruments captured soil moisture for the same region. The extracted information was stored separately in a .CSV file
- 5) The soil moisture measurements in the HDF5 files were present as 1D arrays. We reshaped these to 2D to ensure that we had the appropriate format of input for our Neural Network. Moreover, the locations for which soil moisture was absent had fill values of -9999. These values had to be removed before reshaping. Hence, we eliminated all the fill values before resizing the soil moisture from 1D to 2D.
- 6) The resized data arrays were converted to 2D images. This was done for datasets of both instruments that captured soil moisture data on the East Coast on the same day and at the same time. These Images were stored in .PNG format. At the end of this step we had a collection of resized images representing the Radar and Radiometer datasets for the same region on the same date and same time.
- 7) To train our CNN, we divided the 36Km<sup>2</sup> (low resolution images) images into training set and test set with 80% of our collection in the training set and 20% in the test set. We had a total of 23 images with 17 in our training set and 6 in our test set.
- 8) We trained our CNN on the training set during which it learnt to convert low-resolution images to high resolution. (i.e; converting a 36Km<sup>2</sup> image to 3Km<sup>2</sup>).
- 9) After the model was trained, it was tested with a 36Km<sup>2</sup> input image that it did not encounter while training. This step produced an enhanced image as output.
- 10) We compared the output against the corresponding original 3Km<sup>2</sup> image to check if they were the same. We used the "Mean Squared Error" loss function metric to determine the prediction accuracy of our SRCNN model

## CHAPTER-6

### EXPERIMENTAL SETUP

In the initial phase of the project, we created geographical soil moisture plots to represent the average soil moisture measurements captured by the Radar and the Radiometer instruments. This was performed for a two-month period ranging from the May 1<sup>st</sup>, 2015 to June 30<sup>th</sup>, 2015. We created these plots to identify the average density of soil moisture present at different locations over a period of 2 months.

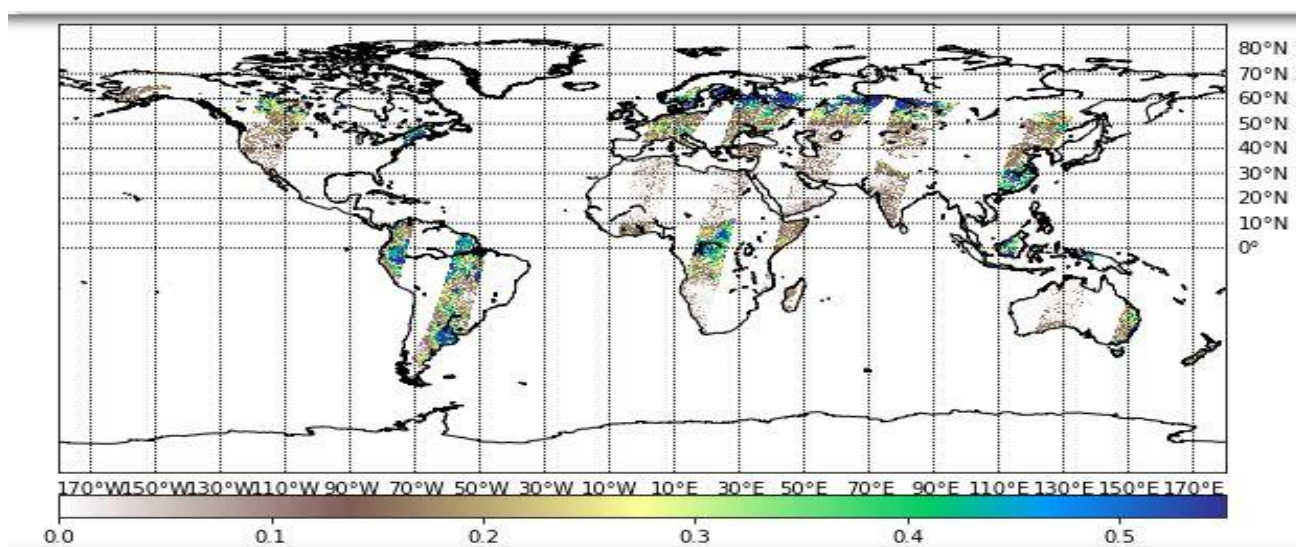


Fig. 2 Average soil moisture density at various locations between May 1<sup>st</sup>, 2015 June 30<sup>th</sup>, 2015 for the Radar spectral band.

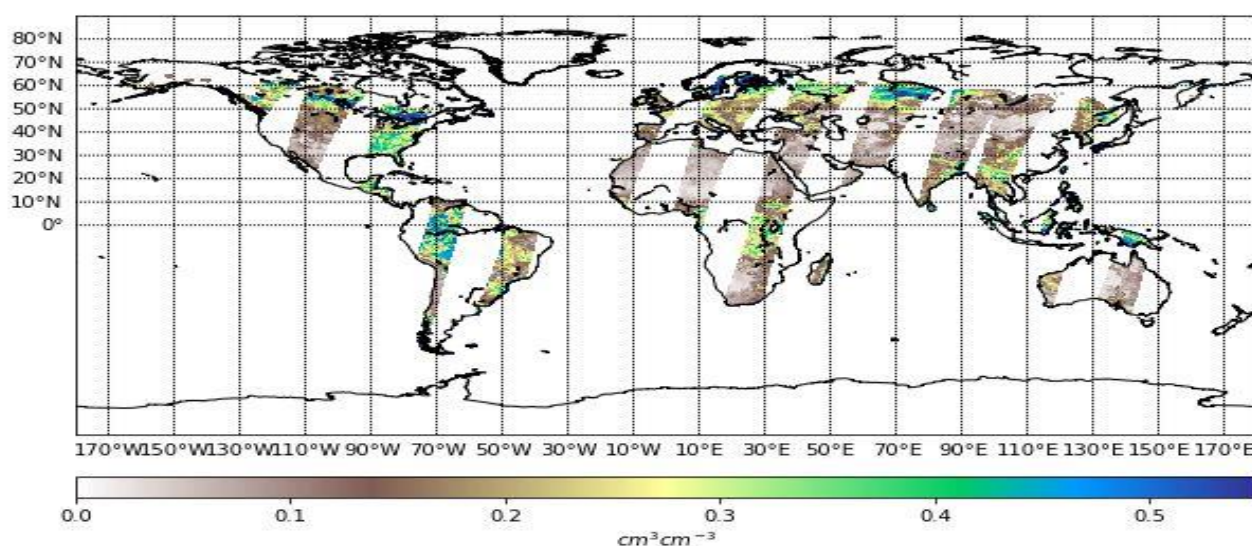
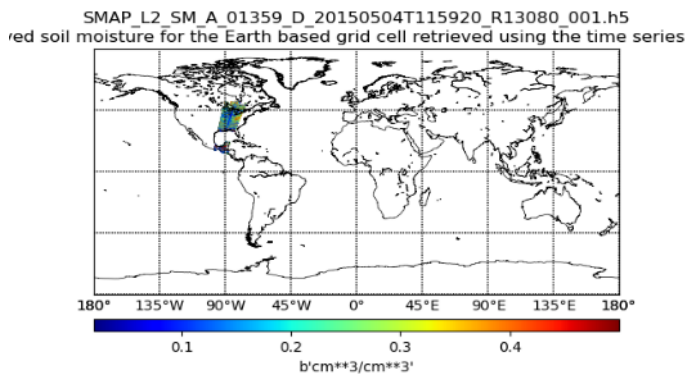
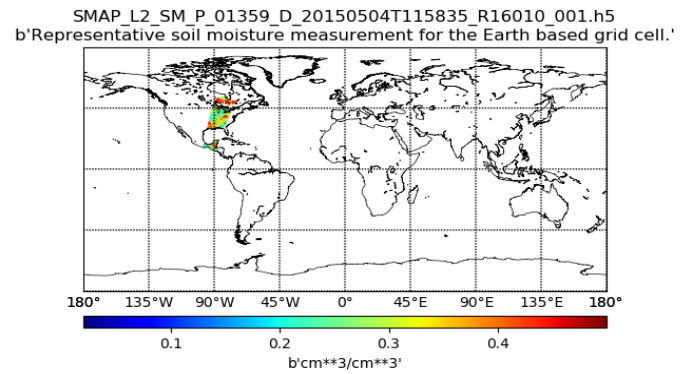


fig.3 Average soil moisture density at various locations between May 1<sup>st</sup>, 2015 June 30<sup>th</sup>, 2015 for the Radiometer spectral band

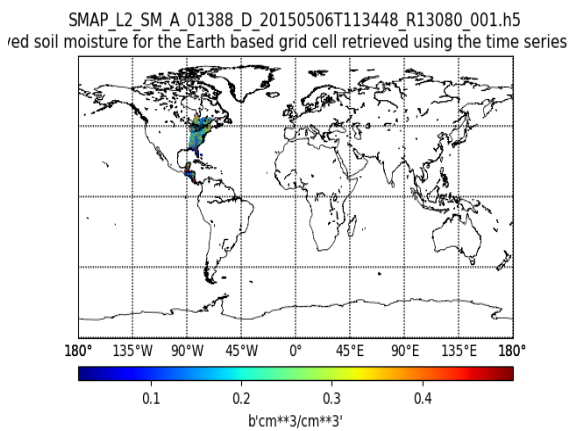
After gathering information on the soil moisture density at different locations from Fig.2 and fig 3, we decided to apply the Super Resolution data reconstruction for a region covering the East Coast of the US. To achieve this task, we plotted the soil moisture density captured by the Radar and Radiometer instruments on the East Coast for each day for the entire duration of May 1st -June 30<sup>th</sup>, 2015. These plots helped us identify those days on which the soil measurements captured by both the instruments coincided with each other in terms of the date, time and location that was captured.



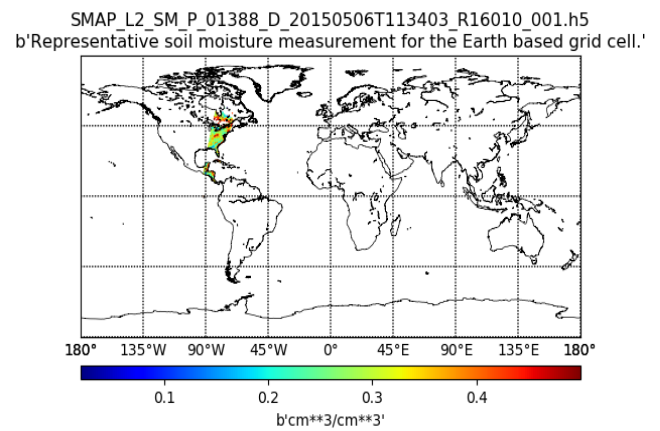
**Fig 4. Soil moisture captured by the Radar band on May 4<sup>th</sup>, 2015 at 11:59:20**



**Fig 5. Soil moisture captured by the Radiometer band on May 4<sup>th</sup>, 2015 at 11:58:35**



**Fig 6. Soil moisture captured by the Radar band on May 6<sup>th</sup>, 2015 at 11:34:48**



**Fig 7. Soil moisture captured by the Radiometer Band on May 6<sup>th</sup>, 2015 at 11:34:03**

Fig.4 and Fig. 5 above, indicate the same date and time for which the data captured by both instruments represented the same region. Fig.6 and Fig. 7 represent the same objective for May 6<sup>th</sup>, 2015. The procedure to create these images have been described in [23] in the “Appendix” section.



## CHAPTER-7

### SUPER RESOLUTION CONVOLUTIONAL NEURAL NETWORK

Consider a single low-resolution image, we first upsampled it to the desired size using bicubic interpolation, which is the only pre-processing we perform. We denote the interpolated image as  $Y$ . Our goal is to recover from  $Y$  an image  $F(Y)$  that is as similar as possible to the ground truth high-resolution image  $X$ . The Super Resolution Convolutional model that we have used for our project is shown Fig. 8. We wish to learn a mapping  $F$ , which conceptually consists of three operations:

- 1) **Patch extraction and representation:** this operation extracts (overlapping) patches from the low-resolution image  $Y$  and represents each patch as a high-dimensional vector. These vectors comprise a set of feature maps, of which the number equals to the dimensionality of the vectors.
- 2) **Non-linear mapping:** this operation nonlinearly maps each high-dimensional vector onto another high-dimensional vector. Each mapped vector is conceptually the representation of a high-resolution patch. These vectors comprise another set of feature map.
- 3) **Reconstruction:** this operation aggregates the above high-resolution patch-wise representations to generate the final high-resolution image. This image is expected to be like the ground truth  $X$ .

#### 7.1 Patch extraction and representation

A popular strategy in image restoration is to densely extract patches and then represent them by a set of pre-trained bases such as PCA, DCT, Haar, etc. This is equivalent to convolving the image by a set of filters, each of which is a basis. In our formulation, we involve the optimization of these bases into the optimization of the network. Formally, our first layer is expressed as an operation:

$$F1: F1(Y) = \max(0, W1 * Y + B1), (1)$$

where  $W1$  and  $B1$  represent the filters and biases respectively, and  $*$  denotes the convolution operation. Here,  $W1$  corresponds to  $n1$  filters of support  $c \times f1 \times f1$ , where  $c$  is the number of channels in the input image,  $f1$  is the spatial size of a filter. Intuitively,  $W1$  applies  $n1$  convolutions on the image, and each convolution has a kernel size  $c \times f1 \times f1$ . The output is composed of  $n1$  feature maps.  $B1$  is an  $n1$ -dimensional vector, whose each element is associated with a filter. We apply the Rectified Linear Unit (ReLU,  $\max(0, x)$ ) on the filter responses.

#### 7.2 Non-linear mapping

The first layer extracts an  $n1$ -dimensional feature for each patch. In the second operation, we map each of these  $n1$ -dimensional vectors into an  $n2$ -dimensional one. This is equivalent to applying  $n2$  filters which have a trivial spatial support  $1 \times 1$ . This interpretation is only valid for  $1 \times 1$  filter. But it is easy to generalize to larger filters like  $3 \times 3$  or  $5 \times 5$ . In that case, the non-linear mapping is not on a patch of the input image; instead, it is on a  $3 \times 3$  or  $5 \times 5$  “patch” of the feature map. The operation of the second layer is:

$$F2(Y) = \max(0, W2 * F1(Y) + B2).$$

Here  $W_2$  contains  $n_2$  filters of size  $n_1 \times f_2 \times f_2$ , and  $B_2$  is  $n_2$ -dimensional. Each of the output  $n_2$ -dimensional vectors is conceptually a representation of a high-resolution patch that will be used for reconstruction. It is possible to add more convolutional layers to increase the non-linearity. But this can increase the complexity of the model ( $n_2 \times f_2 \times f_2 \times n_2$  parameters for one layer), and thus demands more training time.

### 7.3 Reconstruction

The predicted overlapping high-resolution patches are often averaged to produce the final full image. The averaging can be considered as a pre-defined filter on a set of feature maps (where each position is the “flattened” vector form of a high-resolution patch). We define a convolutional layer to produce the final high-resolution image:

$$F(Y) = W_3 * F_2(Y) + B_3. \quad (3) \quad 4$$

Here  $W_3$  corresponds to  $c$  filters of a size  $n_2 \times f_3 \times f_3$ , and  $B_3$  is a  $c$ -dimensional vector. If the representations of the high-resolution patches are in the image domain (i.e., we can simply reshape each representation to form the patch), we expect that the filters act like an averaging filter; if the representations of the high-resolution patches are in some other domains (e.g., coefficients in terms of some bases), we expect that  $W_3$  behaves like first projecting the coefficients onto the image domain and then averaging. In either way,  $W_3$  is a set of linear filters

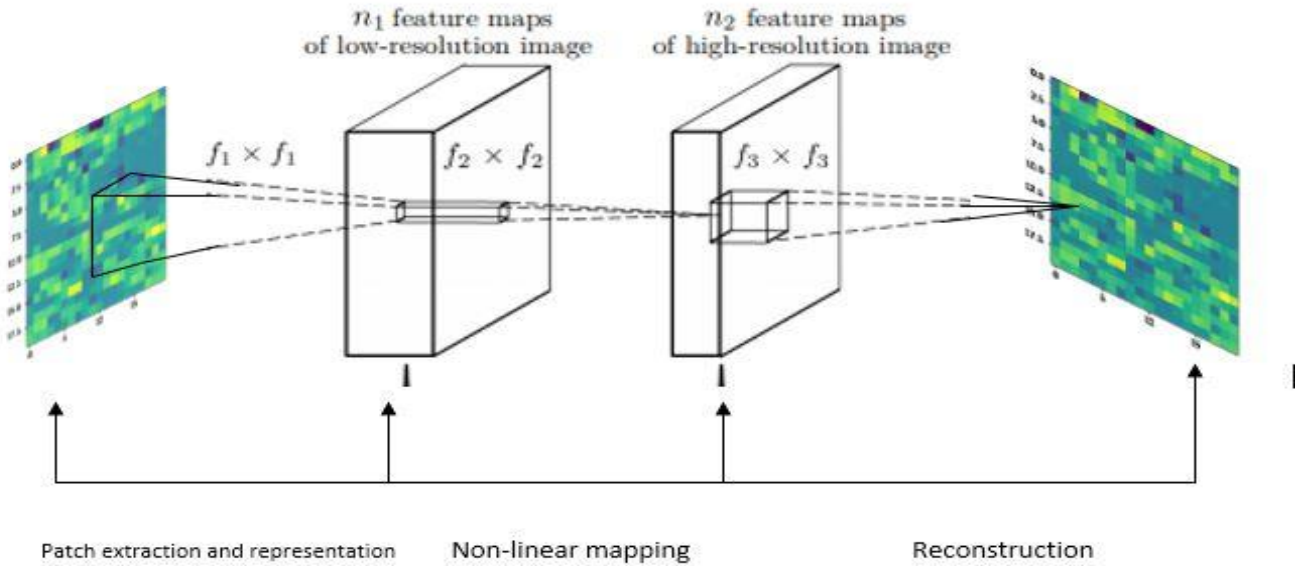


Fig 8. Given a low-resolution image  $Y$ , the first convolutional layer of the SRCNN extracts a set of feature maps. The final layer combines the predictions within a spatial neighborhood to produce the final high-resolution image  $F(Y)$

#### 7.4 Description of our Neural Network

Attribute	Value
Number of convolutional layers	3
Number of filters	n1= 128, n2=64, n3=1
Number of rows	9
Number of columns	9
Number of Pooling layers	3
Initialization	glorot_uniform
Activation technique	ReLu
Optimizer	Adam
Metrics	Mean_Squared_Error

Table 1. Components of our Super Resolution Convolutional Neural Network

**Pooling Layers:** We periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. This function helps to progressively reduce the spatial size of the representation to number of parameters and computation in the network, and hence to also control overfitting. In our model we have a combination of 3 Pooling layers, and 3 filters of different sizes.

**ReLU activation technique:** We used the ReLU activation because it this technique does not involve complex computation and it tends to show better convergence performance compared to other activation techniques that compared to tanh / sigmoid neurons that involve expensive operations (exponentials, etc.), the ReLU can be implemented by simply thresholding a matrix of activations at zero.

**Adam Optimizer:** Adam is an optimization algorithm that can be instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. The advantage of using this optimizer is that it computes training values efficiently, requires little memory and it is invariant to diagonal rescale of the gradients.

#### 7.5 Training

Learning the end-to-end mapping function  $F$  requires the estimation of network parameters  $\Theta = \{W1, W2, W3, B1, B2, B3\}$ . This is achieved through minimizing the loss between the reconstructed images  $F(Y; \Theta)$  and the corresponding ground truth high-resolution images  $X$ . Given a set of high-resolution images  $\{X_i\}$  and their corresponding low-resolution images  $\{Y_i\}$ , we use Mean Squared Error (MSE) as the loss function:

$$L(\Theta) = \frac{1}{n} \sum_{i=1}^n \|F(Y_i; \Theta) - X_i\|^2,$$

where  $n$  is the number of training samples. Using MSE as the loss function favors a high PSNR. The PSNR is a widely-used metric for quantitatively evaluating image restoration quality and is at least partially related

to the perceptual quality. It is worth noticing that the convolutional neural networks do not preclude the usage of other kinds of loss functions, if only the loss functions are derivable. If a better perceptually motivated metric is given during training, it is flexible for the network to adapt to that metric. On the contrary, such a flexibility is in general difficult to achieve for traditional “handcrafted” methods. Despite that the proposed model is trained favoring a high PSNR, we still observe satisfactory performance when the model is evaluated using alternative evaluation metrics, e.g., SSIM, MSSIM. The loss is minimized using stochastic gradient descent with the standard backpropagation. The weight matrices are updated as

$$\Delta_{i+1} = 0.9 \cdot \Delta_i - \eta \cdot \frac{\partial L}{\partial W_i^\ell}, \quad W_{i+1}^\ell = W_i^\ell + \Delta_{i+1},$$

where  $\ell \in \{1, 2, 3\}$  and  $i$  are the indices of layers and iterations,  $\eta$  is the learning rate, and  $\partial L / \partial W_i^\ell$  is the derivative. The filter weights of each layer are initialized by drawing randomly from a Gaussian distribution with zero mean and standard deviation 0.001 (and 0 for biases). The learning rate is  $10^{-4}$  for the first two layers, and  $10^{-5}$  for the last layer. We empirically find that a smaller learning rate in the last layer is important for the network to converge. In the training phase, the ground truth images  $\{X_i\}$  are prepared as  $f_{\text{sub}} \times f_{\text{sub}} \times c$ -pixel sub-images randomly cropped from the training images. By “sub-images” we mean these samples are treated as small “images” rather than “patches”, in the sense that “patches” are overlapping and require some averaging as post-processing but “sub-images” need not. To synthesize the low-resolution samples  $\{Y_i\}$ , we blur a sub-image by a Gaussian kernel, sub-sample it by the upscaling factor, and upscale it by the same factor via bicubic interpolation. To avoid border effects during training, all the convolutional layers have no padding, and the network produces a smaller output  $((f_{\text{sub}} - f_1 - f_2 - f_3 + 3) \times c)$ . The MSE loss function is evaluated only by the difference between the central pixels of  $X_i$  and the network output. Although we use a fixed image size in training, the convolutional neural network can be applied on images of arbitrary sizes during testing.

## CHAPTER-8

### EXPERIMENT

After creating the geographical soil moisture plots represented in figures 4 to 7, we resized the 1D soil moisture data arrays to 2D images for the Radar and Radiometer spectral bands. With this collection of images, we prepared our training set and test set. The procedure for conversion of 1D to 2D is discussed in Chapter-5. Table2 and Table3 contains reshaped images of the Radiometer instrument.

The images in the training set (Table 2) and test set (Table 3) were created in a Keras (version 2) TensorFlow environment with the help of scripts written in Python 3. The experiment that was used to create these images is described in [24] in the Appendix' section. The following packages used to read, analyze and reshape large arrays of numerical data extracted from the .CSV files containing latitude, longitude and soil moisture data have been mentioned below.

**Pandas:** This package available in Python was used in our project to read the .CSV files using the `read_csv()` function provided by this library. We used the latest Pandas version of v 0.23.0. [19]

**Numpy:** This package provided by python was imported in our project to store large 2-dimensional numerical arrays that were reshaped from 1D. The `array()` function was used to store the entire 2D array. The latest version of numpy v 1.17 was used in our project. [21]

**Keras.layers:** This package provided by the Keras was imported in our project to avail the `Input()` function that helped us read the shapes of the Longitude, Latitude and Soil moisture arrays from the .CSV files. The latest version of Keras 2 was used in our project. [22]

#### 8.1 Training Dataset

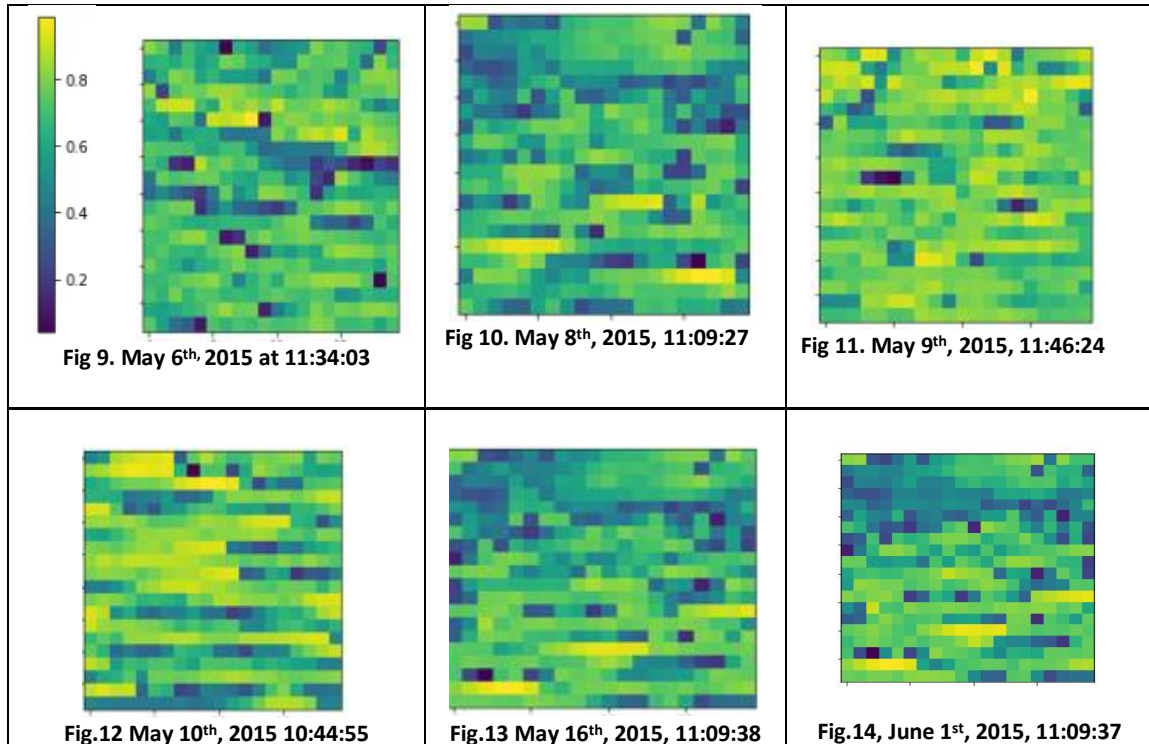


Table 2: Sample training dataset which is a part of our original Training set containing 17 images.

## 8.2 Test Image Set

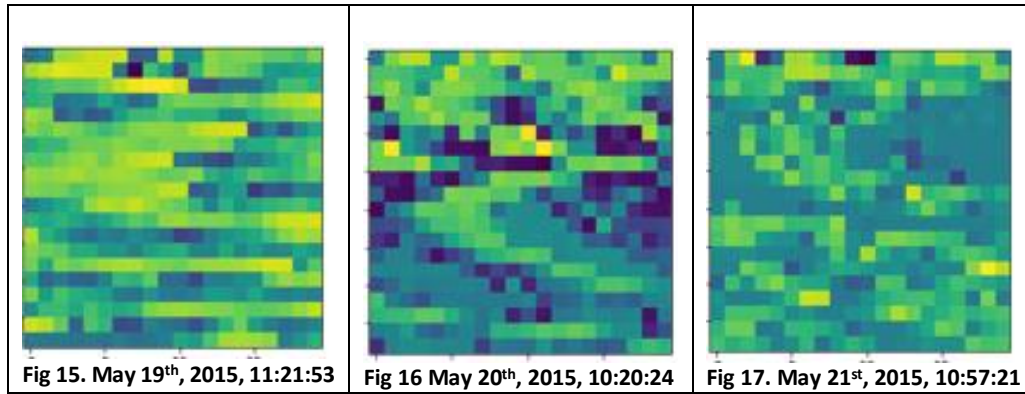


Table 3: Sample images from our test dataset containing 6 images in total

Our Training image set (Table. 2) had 17 images in total while the test set (Table.3) had 6 images in total. The training set consisted of Radiometer spectral images which were of low resolution. During training, our CNN was trained to convert a low-resolution image to high resolution. After training, we analyzed the SRCNN's ability to convert a low-resolution image to high resolution. This was performed on an image from the test dataset that the model had not encounter during training. In our project, we provided Fig. 15 as input to the SRCNN and tested with 200 epochs. The construction of Fig. 19 and Fig. 20 in Table 4 have been described in [25] under "Appendix" section in the Super Resolution Convolutional Neural Network. The results that we obtained has been shown in Section-8.3

## 8.3 Results:

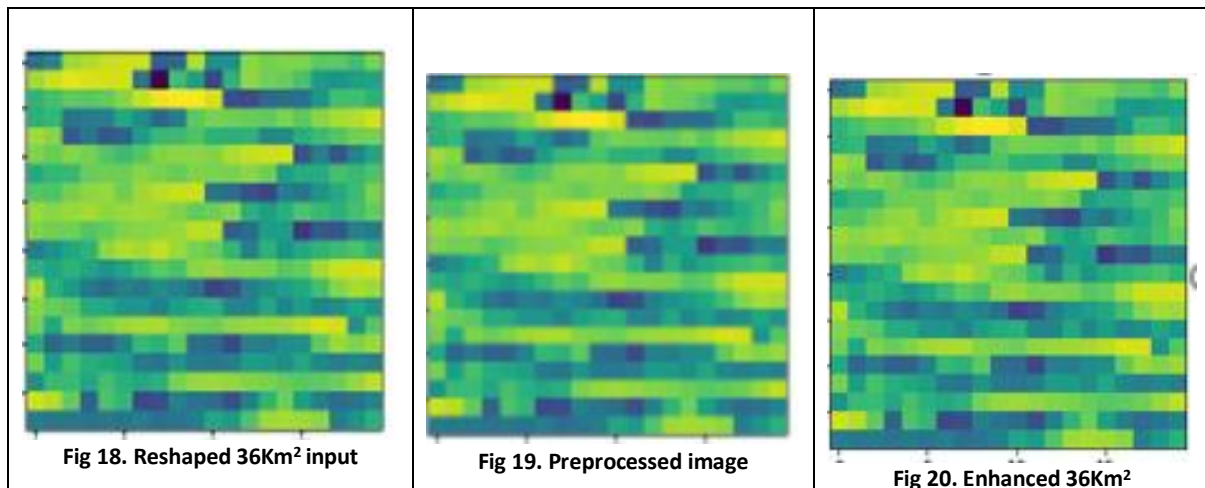


Table 4: Fig.20 represents the output produced by the SRCNN.

This is an enhanced version of Fig. 18 which represents the Radiometer data for May 19<sup>th</sup>, 2015, 11:21:53.

#### 8.4 Comparison with the original Radar image

The output produced in the previous section (Fig. 20) was compared with the original 3Km<sup>2</sup> image (Fig. 21) for the same day and time.

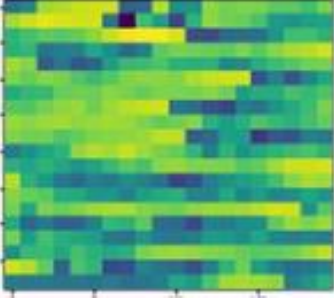
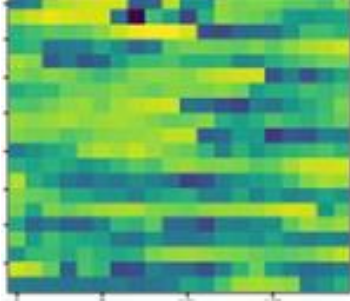
Radar image produced by the SRCNN model	Original Radar image
 Fig 21. Enhanced 36Km <sup>2</sup> output	 Fig 22. Original 3Km <sup>2</sup>

Table 5: Comparing our result with the original image

Table 5 compares the output of our SRCNN with the original Radar image. As we can observe, both these images were almost identical to each other.

#### Improvement in MSE:

Epoch runs	MSE (Mean Square Error)
1-50	0.00297
50-100	0.00206
100-150	0.00176
150-200	<b>0.00162</b>

Table 6: Improvement of MSE in the training phase

Epoch runs	MSE (Mean Square Error)
1-50	0.00313
50-100	0.00231
100-150	0.00210
150-200	<b>0.00197</b>

Table 7: Improvement of MSE in the test phase

## CHAPTER-9

### CONCLUSION

The results we achieved from our experiments (Fig. 23 and section-8.3) show that the Super Resolution Convolutional Neural Network that we developed was able to construct a high-resolution image of the Radar spectral instrument ( $3\text{Km}^2$ ) after being trained to convert a Radiometer image of spatial coverage ( $36\text{Km}^2$ ) (lower resolution). Hence, we can successfully claim that our CNN can be used to recover Radar data with the help of Radiometer data for the same day and time when only the Radiometer data is available. We also calculated the MSE for the training phase and test phase (Table 6 and Table 7). There was constant improvement in the MSE values in both phases.



## CHAPTER-10

### REFERENCES

- [1] Albergel, C., Rüdiger, C., Pellarin, T., Calvet, J.-C., Fritz, N., Froissard, F., Suquia, D., Petitpa, A., Pignatelli, B. and Martin, E.: From near-surface to root-zone soil moisture using an exponential filter: an assessment of the method based on in-situ observations and model simulations, *Hydrol. Earth Syst. Sci.*, 12(6), 1323–1337, doi:10.5194/hess-12-1323-2008, 2008.
- [2] Chan, S. K., Bindlish, R., O'Neill, P., Jackson, T., Njoku, E., Dunbar, S., Chaudhury, J., Piepmeyer, J., Yueh, S., Entekhabi, D., Colliander, A., Chen, F., Cosh, M. H., Caldwell, T., Walker, J., Berg, A., McNairn, H., Thibeault, M., Martínez-Fernández, J., Uldall, F., Seyfried, M., Bosch, D., Starks, P., Holifield Collins, C., Prueger, J., van der Velde, R., Asanuma, J., Palecki, M., Small, E. E., Zreda, M., Calvet, J., Crow, W. T. and Kerr, Y.: Development and assessment of the SMAP enhanced passive soil moisture product, *Remote Sens. Environ.*, (August), doi:10.1016/j.rse.2017.08.025, 2017.
- [3] Ines, A. V. M., Mohanty, B. P. and Shin, Y.: An unmixing algorithm for remotely sensed soil moisture, *Water Resour. Res.*, 49(1), 408–425, doi:10.1029/2012WR012379, 2013.
- [4] Jagdhuber, T., Entekhabi, D., Hajnsek, I., Konings, A. G., McColl, K. A., Alemohammad, S. H., Das, N. N. and Montzka, C.: Physically-based active-passive modelling and retrieval for SMAP soil moisture inversion algorithm, in 2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), pp. 1300–1303, IEEE., 2015.
- [5] a microwave pixel: Application to the Monsoon '90 data, *Remote Sens. Environ.*, 101(3), 379–389, doi: 10.1016/j.rse.2006.01.004, 2006. Merlin, O., Chehbouni, A., Walker, J. P., Panciera, R. and Kerr, Y. H.: A Simple Method to Disaggregate Passive Microwave- Based Soil Moisture, *IEEE Trans. Geosci. Remote Sens.*, 46(3), 786–796, doi:10.1109/TGRS.2007.914807, 2008a.
- [6] Marthews, T. R., Dadson, S. J., Lehner, B., Abele, S. and Gedney, N.: High-resolution global topographic index values for use in large-scale hydrological modelling, *Hydrol. Earth Syst. Sci.*, 19(1), 91–104, doi:10.5194/hess-19-91-2015, 2015. Mascaro, G., Vivoni, E. R. and Deidda, R.: Soil moisture downscaling across climate regions and its emergent properties, *J. Geophys. Res.*, 116(D22), n/a-n/a, doi:10.1029/2011JD016231, 2011.
- [7] Seyed Hamed Alemohammad<sup>1,2</sup>, Jana Kolassa<sup>3,4</sup>, Catherine Prigent<sup>1,2,5</sup>, Filipe Aires<sup>1,2,5</sup>, Pierre Gentile<sup>1,2,6</sup>: **Global Downscaling of Remotely-Sensed Soil Moisture using Neural Networks**, 2018
- [8] Shin, Y. and Mohanty, B. P.: Development of a deterministic downscaling algorithm for remote sensing soil moisture footprint using soil and vegetation classifications, *Water Resour. Res.*, 49(10), 6208–6228, doi:10.1002/wrcr.20495, 2013.
- [9] Srivastava, P. K., Han, D., Ramirez, M. R. and Islam, T.: Machine Learning Techniques for Downscaling SMOS Satellite Soil Moisture Using MODIS Land Surface Temperature for Hydrological Application, *Water Resour. Manag.*, 27(8), 3127–3144, 15 doi:10.1007/s11269-013-0337-9, 2013.

- [10] Wu, X., Walker, J. P., Rudiger, C., Panciera, R. and Gao, Y.: Intercomparison of Alternate Soil Moisture Downscaling Algorithms Using Active–Passive Microwave Observations, *IEEE Geosci. Remote Sens. Lett.*, 14(2), 179–183, doi:10.1109/LGRS.2016.2633521, 2017.
- [11] Coarse-Scale Soil Moisture Observations with Implicit Bias Correction, *IEEE Trans. Geosci. Remote Sens.*, 53(6), 3507–3521, doi:10.1109/TGRS.2014.2378913, 2015.
- [12] **Konstantina Fotiadou<sup>1;2</sup>, Grigorios Tsagkatakis<sup>1</sup>, Mahta Moghaddam<sup>3</sup>, Panagiotis Tsakalides<sup>1;2</sup>**: Recovery of Soil Moisture Active Passive (SMAP) Instrument’s Active Measurements via Coupled Dictionary Learning:2018
- [13] Hongtao Jiang<sup>1</sup> Huanfeng Shen<sup>1</sup>, Senior Member, IEEE Xinghua Li<sup>2</sup>, Member, IEEE Liangpei Zhang<sup>3</sup>, Senior Member, IEEE: **GENERATION OF SMAP 9 KM SOIL MOISTURE USING A SPATIO-TEMPORAL INFORMATION FUSION MODEL**, 2017
- [14] SMAP Data Products: <https://smap.jpl.nasa.gov/data/>
- [15] Western, A. W. and Blöschl, G.: On the spatial scaling of soil moisture, *J. Hydrol.*, 217(3–4), 203–224, doi:10.1016/S0022-694(98)00232-7, 1999.
- [16] <https://nsidc.org/data/SPL2SMP/versions/5>
- [17] <https://nsidc.org/data/SPL2SMA/versions/3>
- [18] <https://support.hdfgroup.org/products/java/release/hdfview3.html>
- [19] <https://pandas.pydata.org/>
- [20] <https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html>
- [21] <https://www.numpy.org/>
- [22] [https://www.tensorflow.org/api\\_docs/python/tf/keras/layers/Input](https://www.tensorflow.org/api_docs/python/tf/keras/layers/Input)

## APPENDIX

**[23] Source code for creating soil moisture plots on a geographical map**

```

import os
import h5py
import matplotlib as mpl
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import numpy as np

def run(FILE_NAME):

    with h5py.File(FILE_NAME, mode='r') as f:

        name = '/Soil_Moisture_Retrieval_Data/soil_moisture'
        data = f[name][:]
        units = f[name].attrs['units']
        longname = f[name].attrs['long_name']
        _FillValue = f[name].attrs['_FillValue']
        valid_max = f[name].attrs['valid_max']
        valid_min = f[name].attrs['valid_min']
        invalid = np.logical_or(data > valid_max,
                                data < valid_min)
        invalid = np.logical_or(invalid, data == _FillValue)
        data[invalid] = np.nan
        data = np.ma.masked_where(np.isnan(data), data)

        # Get the geolocation data
        latitude = f['/Soil_Moisture_Retrieval_Data/latitude'][:]
        longitude = f['/Soil_Moisture_Retrieval_Data/longitude'][:]

        m = Basemap(projection='cyl', resolution='l',
                    llcrnrlat=-90, urcrnrlat=90,
                    llcrnrlon=-180, urcrnrlon=180)
        m.drawcoastlines(linewidth=0.5)
        m.drawparallels(np.arange(-90, 91, 45))
        m.drawmeridians(np.arange(-180, 180, 45),
            labels=[True,False,False,True])
        m.scatter(longitude, latitude, c=data, s=1, cmap=plt.cm.jet,
            edgecolors=None, linewidth=0)
        cb = m.colorbar(location="bottom", pad='10%')
        cb.set_label(units)

        basename = os.path.basename(FILE_NAME)
        plt.title('{0}\n{1}'.format(basename, longname))
        fig = plt.gcf()
        # plt.show()
        pngfile = "{0}.py.png".format(basename)
        fig.savefig(pngfile)

if __name__ == "__main__":

    try:
        hdf_file = os.path.join(os.environ['HDF_ZOO_DIR'], hdf_file)
    except KeyError:
        pass

```

```
run(hdffile)
```

The code above [23] describes the construction of Active and Passive soil moisture plots. The h5Py library was imported to help us read the large HDF5 datasets containing on the soil moisture data, The Matplotlib library provided by Python was used for creating base maps. Base maps represent a geographical map with the coastlines of all the continents. The initial part of the code reads the .H5 files containing latitude, longitude and soil moisture data. The path to each of these attributes in the .H5 files are provided as input in the source code. We create base maps and specify the type of base map that we require along with the latitude and longitude markings. Next, we draw the coastlines on the base maps to clearly indicate the continents. We then collect the latitude, longitude and corresponding soil moisture values present in our datasets and represent soil moisture on the base maps at that particular intersection of latitude and longitude.

#### **[24] Source code to reshape 1D soil moisture arrays to 2D and remove-9999**

```
import pandas as pd

import numpy as np

from keras.layers import Input, conv2D

from keras.layers import model

from PIL import Image

import matplotlib.pyplot as plt

input_file=("/home/flipper/ananthsmmap/<filename.csv>")

lat D=36

lonD=36

varD=1

inputs=Input ((LatD, LonD, VarD))

df=pd.read_csv(input_file)

df= df [[df [SMAP_LABEL]>-2000]

print(df)

orig_x=df[SMAP_LABEL].values

print('1D shape')

print(orig_x.shape)

soilm=orig_x.reshape(-1,36)

print('2D shape')

print(soilm.shape)

print(soilm)

im=Image.fromarray(soilm.astype(np.uint8), mode='L')

plt.imshow(soilm)
```

This code [24] was written in Python 2. It describes the process of converting a 1D soil moisture array to 2D. Here, 2D represents the soil moisture array in an image format. Pandas, Numpy and Keras.layers libraries were imported to help us read, analyze and perform computations on the large arrays of soil moisture, latitude and longitude. The initial section in the code reads the .CSV using the read\_csv() function provided by the Pandas library. Next, we define the size of the 2D array as per the requirements of our experiment. Since our work involves 36Km<sup>2</sup> images, we defined 2 variables with LatD and LonD defined as 36. These values are then read by using the Input function provided by the Keras.layers library. Next, we eliminate the all the fill values of -9999. We store all the soil moisture arrays in a variable named orig\_x and then we display the size of this array on the screen. The, we reshape this size using the reshape function in numpy to reshape it to the dimensions we defined initially (36). We now have an array named “soilm” that contains the reshaped soil moisture array present in 2D. We convert this numerical array to a 2D image using the Image function provided by the PIL library In Python. This program was used to create resized images of 3Km<sup>2</sup> datasets as well. The code had to be modified by defining the desired shape of the soil moisture arrays to 3.

### [25] Super Resolution Convolutional Neural Network

```
from keras.models import Sequential
from keras.layers import Conv2D, Input, BatchNormalization
# from keras.layers.advanced_activations import LeakyReLU
from keras.callbacks import ModelCheckpoint
from keras.optimizers import SGD, Adam
import prepare_data1 as pd
import numpy
import math
import os

def psnr(target, ref):
    # assume RGB image
    target_data = numpy.array(target, dtype=float)
    ref_data = numpy.array(ref, dtype=float)

    diff = ref_data - target_data
    diff = diff.flatten('C')

    rmse = math.sqrt(numpy.mean(diff ** 2.))

    return rmse

def model():
    # lrelu = LeakyReLU(alpha=0.1)
    SRCNN = Sequential()
    SRCNN.add(Conv2D(nb_filter=128, nb_row=9, nb_col=9,
init='glorot_uniform',
activation='relu', border_mode='valid', bias=True,
input_shape=(32, 32, 1)))
    SRCNN.add(Conv2D(nb_filter=64, nb_row=3, nb_col=3,
init='glorot_uniform',
activation='relu', border_mode='same', bias=True))
    # SRCNN.add(BatchNormalization())
    SRCNN.add(Conv2D(nb_filter=1, nb_row=5, nb_col=5,
init='glorot_uniform',
activation='linear', border_mode='valid',
bias=True))
```

```

adam = Adam(lr=0.0003)
SRCNN.compile(optimizer=adam, loss='mean_squared_error',
metrics=['mean_squared_error'])
return SRCNN

def predict_model():
    # lrelu = LeakyReLU(alpha=0.1)
    SRCNN = Sequential()
    SRCNN.add(Conv2D(nb_filter=128, nb_row=9, nb_col=9,
init='glorot_uniform',
activation='relu', border_mode='valid', bias=True,
input_shape=(None, None, 1)))
    SRCNN.add(Conv2D(nb_filter=64, nb_row=3, nb_col=3,
init='glorot_uniform',
activation='relu', border_mode='same', bias=True))
    # SRCNN.add(BatchNormalization())
    SRCNN.add(Conv2D(nb_filter=1, nb_row=5, nb_col=5,
init='glorot_uniform',
activation='linear', border_mode='valid',
bias=True))
    adam = Adam(lr=0.0003)
    SRCNN.compile(optimizer=adam, loss='mean_squared_error',
metrics=['mean_squared_error'])
    return SRCNN

def train():
    srcnn_model = model()
    print(srcnn_model.summary())
    data, label = pd.read_training_data("./crop_train.h5")
    val_data, val_label = pd.read_training_data("./test.h5")

    checkpoint = ModelCheckpoint("SRCNN_check.h5", monitor='val_loss',
verbose=1, save_best_only=True,
save_weights_only=False, mode='min')
    callbacks_list = [checkpoint]

    srcnn_model.fit(data, label, batch_size=128,
validation_data=(val_data, val_label),
callbacks=callbacks_list, shuffle=True,
nb_epoch=200, verbose=0)
    # srcnn_model.load_weights("m_model_adam.h5")

def predict():
    srcnn_model = predict_model()
    srcnn_model.load_weights("3051crop_weight_200.h5")
    IMG_NAME = "/home/flipper/ananthsmmap/SRCNN-
keras/Test/P18212458.png"
    INPUT_NAME = "input2.jpg"
    OUTPUT_NAME = "pre2.jpg"

    import cv2
    img = cv2.imread(IMG_NAME, cv2.IMREAD_COLOR)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
    shape = img.shape

```

```

Y_img = cv2.resize(img[:, :, 0], (shape[1] // 2, shape[0] // 2),
cv2.INTER_CUBIC)
Y_img = cv2.resize(Y_img, (shape[1], shape[0]), cv2.INTER_CUBIC)
img[:, :, 0] = Y_img
img = cv2.cvtColor(img, cv2.COLOR_YCrCb2BGR)
cv2.imwrite(INPUT_NAME, img)

Y = numpy.zeros((1, img.shape[0], img.shape[1], 1), dtype=float)
Y[0, :, :, 0] = Y_img.astype(float) / 255.
pre = srcnn_model.predict(Y, batch_size=1) * 255.
pre[pre[:] > 255] = 255
pre[pre[:] < 0] = 0
pre = pre.astype(numpy.uint8)
img = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
img[6: -6, 6: -6, 0] = pre[0, :, :, 0]
img = cv2.cvtColor(img, cv2.COLOR_YCrCb2BGR)
cv2.imwrite(OUTPUT_NAME, img)

# psnr calculation:
im1 = cv2.imread(IMG_NAME, cv2.IMREAD_COLOR)
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2YCrCb)[6: -6, 6: -6, 0]
im2 = cv2.imread(INPUT_NAME, cv2.IMREAD_COLOR)
im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2YCrCb)[6: -6, 6: -6, 0]
im3 = cv2.imread(OUTPUT_NAME, cv2.IMREAD_COLOR)
im3 = cv2.cvtColor(im3, cv2.COLOR_BGR2YCrCb)[6: -6, 6: -6, 0]

print("bicubic:")
print(cv2.PSNR(im1, im2))
print("SRCNN:")
print(cv2.PSNR(im1, im3))

# psnr calculation:
im1 = cv2.imread(IMG_NAME, cv2.IMREAD_COLOR)
im1 = cv2.cvtColor(im1, cv2.COLOR_BGR2YCrCb)[6: -6, 6: -6, 0]
# im2 = cv2.imread(INPUT_NAME, cv2.IMREAD_COLOR)
# im2 = cv2.cvtColor(im2, cv2.COLOR_BGR2YCrCb)[6: -6, 6: -6, 0]
im3 = cv2.imread(OUTPUT_NAME, cv2.IMREAD_COLOR)
im3 = cv2.cvtColor(im3, cv2.COLOR_BGR2YCrCb)[6: -6, 6: -6, 0]

# print("bicubic:")
# print(cv2.PSNR(im1, im2))
print("SRCNN:")
print(cv2.PSNR(im1, im3))

if __name__ == "__main__":
    train()
    predict()

```

The code above represents the functioning of our Super Resolution Convolutional Neural Network. It has four different functions-model (), predict\_model(), train() and test(). Model () and predict\_model() define the structure of the Neural Network in the training and test phase. We define the activation technique, Optimizer and the number of layers in our Network in the training and test phases.

In the training phase the Neural Network is trained to convert a 36Km<sup>2</sup> low resolution image to 3Km<sup>2</sup> high resolution image. The information that is collected by the Neural Network during training is stored in the SRCCNN\_check.h5 HDF5 file. The contents in this file are updated every time the network

is trained. The data stored in this .H5 file is utilized by the model in the prediction or test phase when we provide a test image of low resolution that is to be enhanced. During the test phase, the test image is initially preprocessed to a certain level to ensure that the model extracts as much information as it can to construct an enhanced version of the input. After preprocessing, the color spaces that were used by the mode while converting the test input to the preprocessed form in again converted back to the same color space of the input. This ensures that the output image which is enhanced lies in the same in color space as the input image.