S-PLNT-001

Reconstructing Satellite Remote Sensing Data with Deep Learning Autoencoders for Fine-grained Ecoregional Soil Moisture Predictions

Ram Bhagat
Guilderland High School, New York

Abstract

Soil moisture is a critical variable that regulates land-atmosphere interactions (e.g., via evapotranspiration) and is directly linked with plant productivity and plant survival. Information on soil moisture is crucial for designing appropriate irrigation strategies and increasing crop yield. Additionally, long-term soil moisture coupled with climate information provides insight into potential agricultural thresholds and risks.

Soil moisture data currently comes from satellite remote sensing. However, this data has two issues. (1) Although satellites can provide global information, they are limited to coarse spatial resolution (at the multi-kilometer scale), a scale too large to provide meaningful, representative data. (2) Satellites are also unable to measure soil moisture in areas of dense vegetation, snow cover, or extremely dry surfaces, resulting in gaps in the data.

Advances in Artificial Intelligence, called deep learning autoencoders have shown to learn efficient compressions of data by minimizing the error between the compressed and subsequently reconstructed output versus the original input. Since the compression forces the autoencoder to learn only the essential latent features, the reconstruction ignores nonessential variation such as random noise and missing gaps of data. By combining publicly available datasets of satellite-derived soil moisture measurements from the European Space Agency (ESA), fine-grained, gap-free soil moisture predictions are generated using the denoising capabilities of a deep learning autoencoder.

This project reveals that the denoising deep autoencoder significantly improves the spatial resolution of soil moisture in both simulated and real ecoregional data by an average of 39%, demonstrating a novel application for precision agriculture and crop yield optimization.

1. **Introduction**

Agriculture has existed for thousands of years and yet humans still obtain unpredictable and mixed crop results. One reason for this is the lack to adapt agricultural practices that fit the environmental conditions caused by soil moisture. Soil moisture is the water stored in the soil, which is influenced by factors such as precipitation, temperature, and soil characteristics; soil moisture, therefore regulates the Earth's plant land-atmosphere interactions [1]. Moreover, soil moisture helps define how suitable a crop is to a biome. By measuring the moisture values in the soil, we can predict droughts, design more precise and efficient irrigation systems, monitor floods, and increase crop yields for a given year [2,3].

Soil moisture has been routinely measured since the 1980s. However, attempts at capturing global soil moisture data have been a struggle because remote satellite sensing technologies poorly measure soil moisture in areas of dense vegetation, snow cover, or extremely dry surfaces. Additionally, due to the limitations of remote sensing satellites, this data is measured at a coarse spatial resolution (at the multi-kilometer scale), a scale too large to provide meaningful, representative data for agricultural interests. Several techniques have been proposed for reconstructing/refining the data but often only result in specific regions having fine-grained data [3].

Neural networks are mathematical models that can extract nonlinear relationships by analyzing training examples. Initially, neural networks start out with a randomized set of weights and biases, but as more and more training examples are provided, these weights and biases become fine-tuned [4]. Neural networks can manifest in many neural architectures and models. However, in recent literature, deep convolutional neural networks have demonstrated exceptional ability in understanding high dimensional patterns in 2D matrix-based datasets [5]. Autoencoders are special types of neural networks which rely on unsupervised learning to learn an approximation to the identity function. Briefly, within an autoencoder neural network, an input is passed through layers of compression known as the "encoding" phase to the "latent space representation", where a separate decoder model attempts to reconstruct the data through layers of expansion. As the input goes through the compression of the encoding phase, the size gets smaller, while trying to stay as similar to the original as possible. The method for trying to stay as similar to the original as possible involves using activation functions to indicate what is and what isn't the important information. Eventually, it reaches a size where it can no longer be compressed known as the latent space representation. Here, the encoder analyzes the given image and learns to obtain non-linear relationships. The decoder phase contrasts the encoder phase. Here, the size increases from the latent space representation through multiple upsampling convolutional layers, eventually reconstructing the original input. Deep learning autoencoders have shown to improve upon the standard Autoencoder and learn to compress data by minimizing the error between the compressed and subsequently reconstructed output versus the original input [6, 7]. Since the compression forces the autoencoder to learn only the essential

latent features, the reconstruction ignores nonessential variation such as random noise and missing gaps of data, serving a significant role in many denoising applications [6, 7, 8, 9].

2. **Methodology and Materials**

Data Extraction

The European Space Agency agency provided publicly available soil moisture datafiles with a spatial resolution of 0.25 degrees and a temporal resolution of one day starting from the 1990s [10]. The autoencoder was solely trained on the data collected by the European Agency for this study. This study focuses primarily on soil moisture from every continent except Antarctica, due to there being no findings of soil moisture by the satellite. This data was then divided into smaller image samples using latitude and longitude because. For this project, a 120 x 120 grid size, 240 x 240 grid, and 360 x 360 size were used for experimentation. Different grid sizes provide the opportunity to analyze the autoencoder's ability to understand and rescale complex contextual soil moisture patterns from a local to a global scale.

Dataset Distribution

A training dataset is vital for the creation of any deep learning model. An autoencoder will struggle to interpret reconstruct input soil moisture input arrays if it is trained on flawed datasets. Flawed datasets can heavily influence the effectiveness of an autoencoder and lead to model errors such as underfitting(when a machine learning model is not complex enough to accurately capture relationships between a dataset's features and a target variable). A testing dataset is equally as important as a training dataset. Briefly, a testing dataset includes soil moisture data that the model will not originally learn from during the training phase. The training dataset, thus, allows for the evaluation of the autoencoder on unseen data and provides insight into the generalization capabilities of the autoencoder.

In order for the autoencoder to develop the non-linear relationships, it must be trained with soil moisture matrices with sufficient data. In order to filter out the soil moisture data with and without data, a threshold was placed for each image to meet. If a given image had soil moisture measurements of less than 8 %, then it is not added to the training dataset because it is considered a sea point or coast point. This threshold value was calculated using supervised principal median component analysis.

This training data (which consists of 80% of the complete dataset) is split into several batches for the autoencoder to experiment from. The reason the training dataset is split into several batches is that batches allow the autoencoder to handle large amounts of samples once at a time.

A subset of the data (testing dataset) is withheld from the analysis for the autoencoder to test the non-linear relationship it is developing. At the end of each epoch (the time it takes for an

autoencoder to go through one batch of training data), the final reconstruction is compared to the testing dataset using mean squared error, which is calculated as:

$$(1)\ MSE\ =\ \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2$$

Data Generation

When using satellite data, the number of matrices without any gaps is very small, and is difficult to provide enough training data when only data with no gaps are used. Therefore, the aim is to derive a reconstruction strategy that can cope with the large amounts of missing data typically found in remote-sensing data.

To do this, large amounts of gaps were created on the training data. This was done by locating a value within a given image and then taking all values within a distance of 15 degrees latitude and 30 degrees longitude for 120x120 grids, 30 degrees latitude, and 60 degrees longitude for 240x240 grids, and 45 degrees latitude and 90 degrees longitude for 360x360 grids. Each element of the grid was set to a value of -9999, which was the NaN(Not a Number) value formatted in the ESA soil moisture data.

Before going through the autoencoder, however, the datasets are transformed through several operations to allow activation functions such as “ReLU”(rectified linear unit) and “sigmoid” to work properly:

- Add 9999 to all values in every dataset (ReLU best handles data that is non-negative).
- Rescale the array by dividing the dataset by the maximum value in the dataset. This transforms the dataset into values between 0 and 1, which is calculated as follows:

$$(2)\ x\ =\ x/max(x)$$

- Reshape the array to include a new dimension to represent the number of channels within the image (to allow the data to process through typical convolutional encoder layers).

The complete dataset is thus represented by an array of the size $(samples,\ grid\ size,\ grid\ size,\ 1)$.

Model

The overall structure of the neural network is a convolutional auto-encoder. Its main building blocks are convolutional layers which downsample the input during the encoding phase and upsample the latent representation during the decoding phase. The number of encoding and decoding layers in an autoencoder were symmetric to each other. The number of layers would vary depending on the latitude and longitude dimension of each image. This is because the encoding process would always divide the presented size by two, and so eventually the encoder would reach an odd number where it is no longer compressible.

Encoder

The encoder neural architecture starts with an input channel which receives a matrix of size $(samples,\ grid\ size,\ grid\ size,\ 1)$. Briefly, the convolutional operation is executed by summing the dot product of the input region by another matrix called the kernel. The kernel is a matrix that consists of weights that can be used to discover features and document pattern occurrences in a hierarchical-based procedure. The kernel in this case has a window size of 3x3 that convolves around the input. The stride of the kernel is 1 and it represents the distance the kernel convolves between regions. Activation is used to normalize values, whereas pooling summarizes feature maps for dimensionality reduction and documenting highly associated values. In essence, the repeating structure of 16 total convolution and max-pooling layers passes a compression pipeline that yields a latent representation of the input. During the training phase, that is, backpropagation, the weights for the encoder kernels are optimized to reduce the reconstruction loss.

Decoder

The decoder neural architecture starts with the latent representation, a matrix with a size of (15, 15, 16) for 240x240 and 120x120 training samples of a matrix with a size of (45, 45, 16) for 360x360 samples. The upsampling operation doubles the dimensions of the input and performs an inverse convolution operation (defined previously in the encoder section). The reconstruction layer is the final input of matrix size $(samples,\ grid\ size,\ grid\ size,\ 1)$.

The 120x120, 240x240, and 360x360 model summaries are presented in Tables I, II, and III, respectfully. A graphical visualization of the model is underneath the Tables.

Table I: *120x120 Model Summary*

Layer (type)	Matrix Shape	Parameters
Input (InputLayer)	[(None, 120, 120, 1)]	0
Conv2d (Conv2D)	(None, 120, 120, 16)	160
max_pooling2d (MaxPooling2D)	(None, 60, 60, 16)	0
dropout (Dropout)	(None, 60, 60, 16)	0
Conv2d (Conv2D)	(None, 60, 60, 16)	2320
Max_pooling2d (MaxPooling2	(None, 30, 30, 16)	0
Dropout (Dropout)	(None, 30, 30, 16)	0
Conv2d (Conv2D)	(None, 30, 30, 16)	2320
Max_pooling2d (MaxPooling2	(None, 15, 15, 16)	0
Conv2d (Conv2D)	(None, 15, 15, 16)	2320
Up_sampling2d (UpSampling2D)	(None, 30, 30, 16)	0

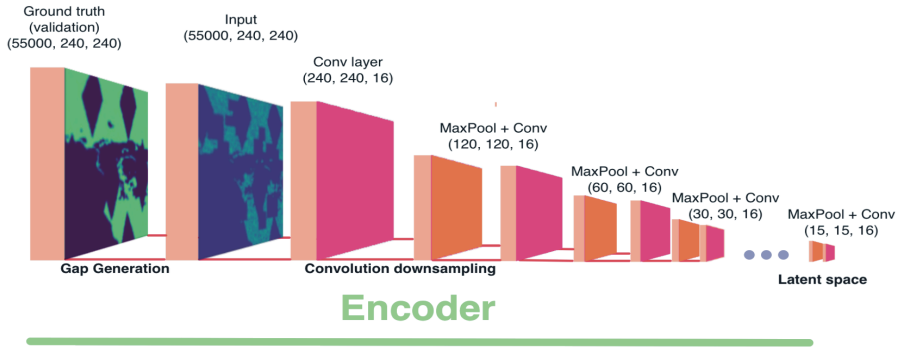
Dropout (Dropout)	(None, 30, 30, 16)	0
Conv2d (Conv2D)	(None, 30, 30, 16)	2320
Up_sampling2d (UpSampling2	(None, 60, 60, 16)	0
Dropout (Dropout)	(None, 60, 60, 16)	0
Conv2d (Conv2D)	(None, 60, 60, 16)	2320
Up_sampling2d (UpSampling2	(None, 120, 120, 16)	0
Conv2d (Conv2D)	(None, 120, 120, 1)	145

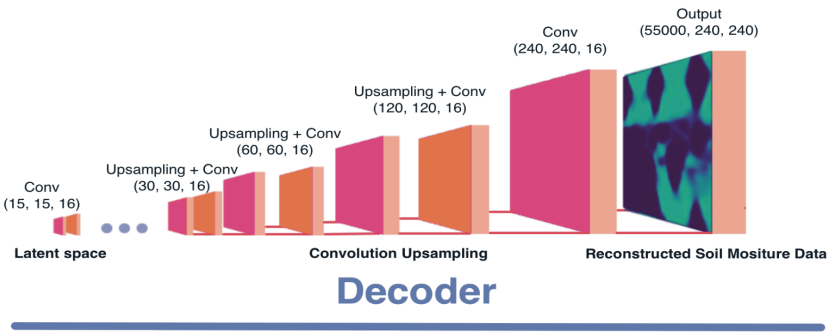
Table II: *240x240 Model Summary*

Layer (type)	Output Shape	Param #
Input (InputLayer)	[(None, 240, 240, 1)]	0
Conv2d (Conv2D)	(None, 240, 240, 16)	160
max_pooling2d (MaxPooling2	(None, 120, 120, 16)	0
dropout (Dropout)	(None, 120, 120, 16)	0
conv2d (Conv2D)	(None, 120, 120, 16)	2320
max_pooling2d (MaxPooling2	(None, 60, 60, 16)	0
conv2d (Conv2D)	(None, 60, 60, 16)	2320
max_pooling2d (MaxPooling2	(None, 30, 30, 16)	0
conv2d (Conv2D)	(None, 30, 30, 16)	2320
max_pooling2d (MaxPooling2	(None, 15, 15, 16)	0
conv2d (Conv2D)	(None, 15, 15, 16)	2320
Up_sampling2d (UpSampling2	(None, 30, 30, 16)	0
Dropout (Dropout)	(None, 30, 30, 16)	0
Conv2d (Conv2D)	(None, 30, 30, 16)	2320
Up_sampling2d (UpSampling2	(None, 60, 60, 16)	0
Conv2d (Conv2D)	(None, 60, 60, 16)	2320
Up_sampling2d (UpSampling2	(None, 120, 120, 16)	0
Conv2d (Conv2D)	(None, 120, 120, 16)	2320
Up_sampling2d (UpSampling2	(None, 240, 240, 16)	0
Conv2d (Conv2D)	(None, 240, 240, 1)	145

Table III: *360x360 Model Summary*

Layer (type)	Output Shape	Param #
Input (InputLayer)	[(None, 360, 360, 1)]	0
Conv2d (Conv2D)	(None, 360, 360, 16)	160
Max_pooling2d (MaxPooling	(None, 180, 180, 16)	0
Dropout (Dropout)	(None, 180, 180, 16)	0
Conv2d (Conv2D)	(None, 180, 180, 16)	2320
Max_pooling2d (MaxPooling	(None, 90, 90, 16)	0
Conv2d (Conv2D)	(None, 90, 90, 16)	2320
Max_pooling2d (MaxPooling	(None, 45, 45, 16)	0
Conv2d (Conv2D)	(None, 45, 45, 16)	2320
Up_sampling2d (UpSampling	(None, 90, 90, 16)	0
Dropout (Dropout)	(None, 90, 90, 16)	0
Conv2d (Conv2D)	(None, 90, 90, 16)	2320
Up_sampling2d (UpSampling	(None, 180, 180, 16)	0
Conv2d (Conv2D)	(None, 180, 180, 16)	2320
Up_sampling2d (UpSampling	(None, 360, 360, 16)	0
Conv2d (Conv2D)	(None, 360, 360, 1)	145





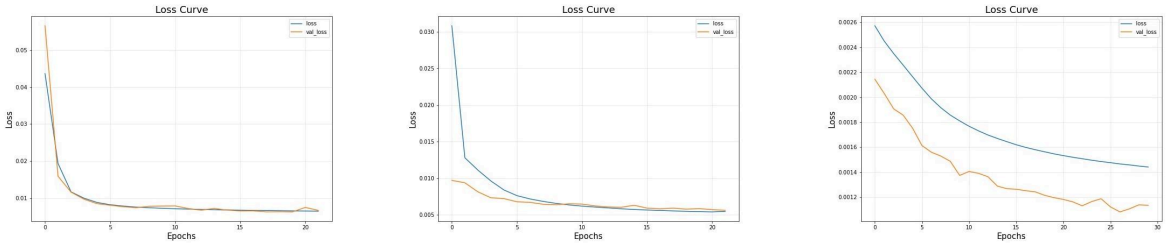
Training the Model

The input dataset is randomly shuffled and partitioned into mini-batches of 256 matrices. The loss function “mean square error” was used to train the autoencoder while being optimized with “rmsprop”. Mean square error was used as the loss function because it provides a better interpretation of the similarity between two-dimensional matrices than other loss functions that might focus on correlations and variances. RMS prop is an optimized gradient function. Furthermore, the model used Adam as its stochastic optimizer, and overfitting was addressed with the use of dropout (with a probability of 0.3). Each model was trained for 25 epochs. The results of the model were evaluated using k-fold cross-validation (k=5) and, for all datasets, were evaluated on an 80:20 train-test split. The full model was implemented in Python with Tensorflow 2.4.0 as the deep learning backend.

3. Results

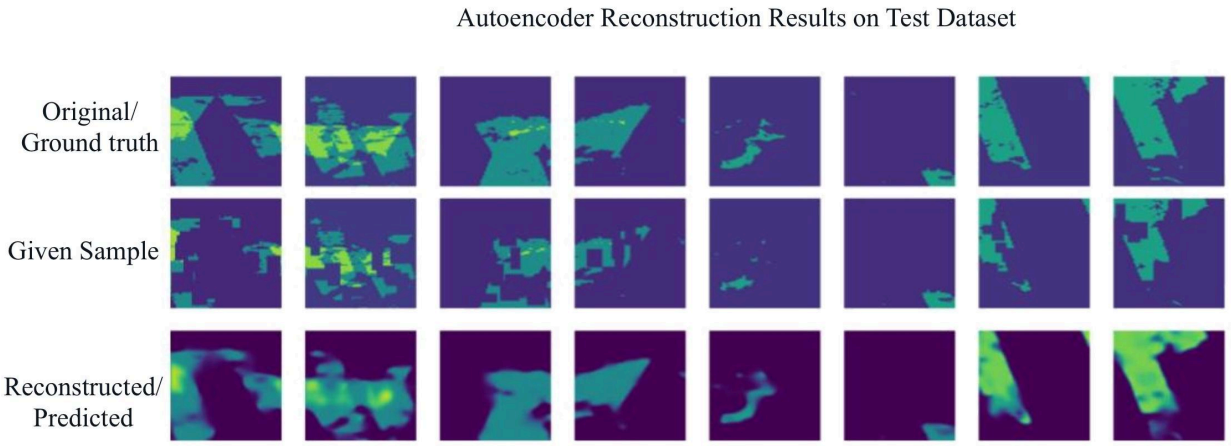
As mentioned previously, the validation reconstruction loss of each model determines how accurately the autoencoders were able to reconstruct the original dataset given only gapped and noisy soil moisture data. The reconstruction loss was measured over each epoch during the training session of each model. The reconstruction loss graphs are represented in the below figure.

Reconstruction Loss Curves for Various Grid Sized Soil Mositure Autoencoders



(a) 120x120 model (b) 240x240 model (c) 360x360 model

In addition to the reconstruction loss comparisons, the average percent change of reconstruction output soil moisture data compared to the original test dataset soil moisture datasets over a span of 1990-2020 was calculated using matrix element-wise analysis. The calculated percent change was measured at +39%. The below figures represent samples that the autoencoder predicted (*the autoencoder never saw the original/ground truth data, it only uses the given sample to predict the original data*).



6. Conclusions

This project presents a consistent way to handle missing data in satellite data using neural networks. Specifically, a deep learning autoencoder neural architecture was exploited to generate fine-grain predictions on noisy inputs. Three models were generated using different training size inputs, 120x120, 240x240, and 260x360. The calculated percent change for the autoencoder's improvement in spatial resolution was measured at +39%. Additionally, the expected error predicted by the neural network provides a good indication of the accuracy of the reconstruction. Provided the success of this model, it is expected to be a critical tool in the advancement of precision agriculture applications.

7. References

[1] Shafi, Uferah, et al. "Precision Agriculture Techniques and Practices: From Considerations to Applications." *Sensors*, vol. 19, no. 17, 2019, p. 3796., doi:10.3390/s19173796.
[2] Sishodia, R.P.; Ray, R.L.; Singh, S.K. Applications of Remote Sensing in Precision Agriculture: A Review. Remote Sens. 2020, 12, 3136. https://doi.org/10.3390/rs12193136
[3] Kolassa, J.; Reichle, R.H.; Liu, Q.; Cosh, M.; Bosch, D.D.; Caldwell, T.G.; Colliander, A.; Holifield Collins, C.; Jackson, T.J.; Livingston, S.J.; Moghaddam, M.; Starks, P.J. Data Assimilation to Extract Soil Moisture Information from SMAP Observations. Remote Sens. 2017, 9, 1179. https://doi.org/10.3390/rs9111179
[4] Larry Hardesty | MIT News Office. "Explained: Neural Networks." *MIT News | Massachusetts Institute of Technology*, news.mit.edu/2017/explained-neural-networks-deep-learning-0414**.**
[5] A. Ajit, K. Acharya and A. Samanta, "A Review of Convolutional Neural Networks," 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), Vellore, India, 2020, pp. 1-5, doi: 10.1109/ic-ETITE47903.2020.049.
[6] Ashfahani, Andri, et al. "DEVDAN: Deep Evolving Denoising Autoencoder." Neurocomputing, vol. 390, 2020, pp. 297–314., doi:10.1016/j.neucom.2019.07.106.

[7] Perera, Lokukaluge P., and B. Mo. “Ship Performance and Navigation Data Compression and Communication under Autoencoder System Architecture.” *Journal of Ocean Engineering and Science*, vol. 3, no. 2, 2018, pp. 133–143., doi:10.1016/j.joes.2018.04.002.
[8] He, Chengxun, et al. “TSLRLN: Tensor Subspace Low-Rank Learning with Non-Local Prior for Hyperspectral Image Mixed Denoising.” Signal Processing, 2021, p. 108060., doi:10.1016/j.sigpro.2021.108060.
[9] Moradzadeh, Alireza, and N. R. Aluru. “Molecular Dynamics Properties without the Full Trajectory: A Denoising Autoencoder Network for Properties of Simple Liquids.” *The Journal of Physical Chemistry Letters*, vol. 10, no. 24, 2019, pp. 7568–7576., doi:10.1021/acs.jpclett.9b02820.
[10]“Soil Moisture ECV Product User Guide (PUG) Supporting Product Version v05.2 Deliverable ID: D4.2 Version 1.” Earth Observation Data Centre for Water Resources Monitoring (EODC) GmbH, 29 May 2020.