INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

DEPARTMENT OF MATHEMATICS AND STATISTICS

MTH393A

Statistical Downscaling using Deep Learning of Rainfall Data Over Indian Region

---

Author:

P S Priti Sudha

190578

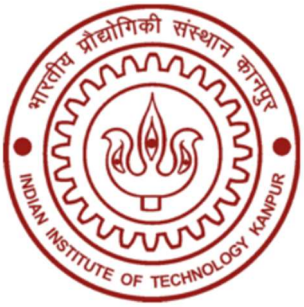
Supervisor:

Dr. B V Rathish Kumar

forwarded

Rathish

Nov 1, 2022



Annexure-II

DECLARATION

I/We hereby declare that the work presented in the project report entitled Statistical downscaling using deep learning of rainfall data over Indian region contains my own ideas in my own words. At places, where ideas and words are borrowed from other sources, proper references, as applicable, have been cited. To the best of our knowledge this work does not emanate from or resemble other work created by person(s) other than mentioned herein.

Name and Signature: P S Priti Sudha P. S. Pritisudha

Date: 1st November 2022

Abstract

Downscaling means that given a poor image, we come up with a detailed or improved image. It means that when given a low-resolution image, we may not get all the features. Therefore, we want a high-resolution image to get all features. We can convert low-resolution image to high-resolution image using the downscaling algorithms based on AI/ML. So, the problem statement here is, given low-resolution information on rainfall, we would have to come up with potential high-resolution information resolution via CNN architecture, namely Stacked Super Resolution CNN, that we would have got otherwise by using a high-resolution camera. It is hence shown that predictions from this deep learning architecture is comparable to the expected images of high spatial resolution.

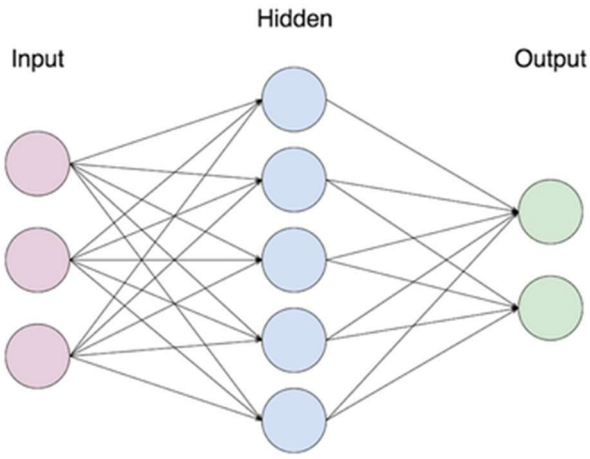
Contents

Introduction

Neural Networks:

A Neural Network (NN) is a series of algorithms that attempts to identify underlying relationships and patterns in a large dataset. It replicates the working of a human brain. Neural networks typically consist of three layers –

1. Input layer
2. A hidden layer (where feature extraction takes place, tweaks are made to train faster and function better)
3. Output layer



Neuron is a basic building block of a NN. It takes weighted values, performs mathematical calculation, and produce output. It is also called a unit, node, or perceptron.

Let n_l denote the number of layers in our network; thus $n = 3$ in our example. We label layer as l, so layer *1* is the input layer, and layer *3* is the output layer. Our neural network has parameters (W, b), where we write W_{ij} to denote the parameter (or weight) associated with the connection between unit j in layer l, and unit i in layer $l+1$. Also, b_i is the bias associated with unit i in layer $l+1$. Bias units don't have inputs or connections going into them, since they always output the value +1. We also let s_l denote the number of nodes in layer l. a_i denotes the activation (meaning output value) of unit i in layer l. For $l=1$, we also use $a_i = x_i$ to denote the i-th input. Given a fixed setting of the parameters W, b, our neural network defines a hypothesis $h_{W,\,b}(x)$ that outputs a real number. Specifically, the computation that this neural network represents is given by [8]:

$$a_1 = f(W_{11}x_1+W_{12}x_2+W_{13}x_3+b_1)$$

$$a_2 = f(W_{21}x_1+W_{22}x_2+W_{23}x_3+b_2)$$

$$a_3 = f(W_{31}x_1+W_{32}x_2+W_{33}x_3+b_3)$$

$$h_{W,\,b}(x) = f(W_{11}a_1+W_{12}a_2+W_{13}a_3+b_1)$$

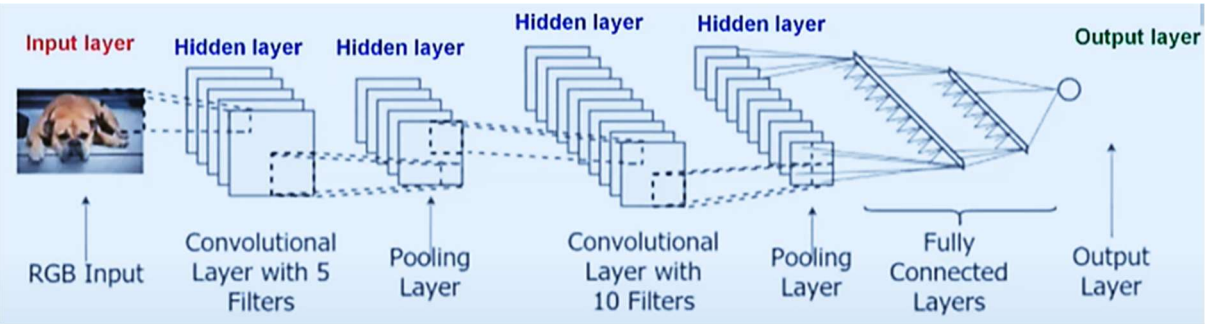
Let z_i denote the total weighted sum of inputs to unit I in layer l, including the bias term. Specifically, if we extend the activation function $f(\cdot)$ to apply to vectors in an element-wise fashion (i.e., $f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$), then we can write the equations above more compactly as [8]:

$$z = W x + b$$

$$a = f(z)$$

CNN (Convolution Neural Networks):

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.[11]



The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlaps to cover the entire visual area.[11]

- CNN has more than one hidden layer.
- The inputs are images which are convolved by a kernel of preferably smaller size in CNN.
- It sequentially covers the full image to generate different number of layers.

CNNs **have an input layer, some hidden layers, and an output layer** that outputs the transformed input. The hidden layers inside a CNN use **filters**, detectors for patterns.

There are 3 main types of layers to build CNN architectures:

1. **Convolutional Layer**:
    - It performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field.
    - The image is composed of three (RGB) channels means the kernel height and width will be spatially small, but the depth extends up to all three channels.[11]
    - During the forward pass, the kernel slides across the height and width of the image producing a two-dimensional representation of the image known as an activation map that gives the response of the kernel at each spatial position of the image.[11]

2. **Pooling Layer**:
    - It reduces the spatial size of the representation, which decreases the required amount of computation and weights in the network hence reduce overfitting.
    - Like kernels, pooling moves a n x n filter (i.e., 3x3 filter) over the features and takes one value determined over the input pixels.

- The pooling operation is processed on every slice of the representation individually.[11]

3. **Fully Connected Layer:**
   - Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer that maps the representation between the input and the output.[11]

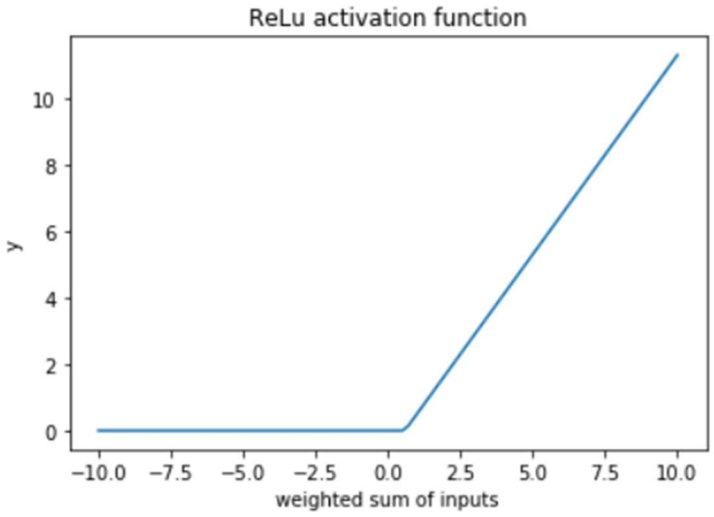
Activation function –

- It is used to increase non-linearity of the network without affecting receptive fields of convolutional layers.
- It applies a non-linear transformation to input layer and acts as a mathematical gate between two layers.
- It incorporates non-linear properties to a network, allowing it to discover more complex functions.
- It converts an input signal of a node to a correctly measured output signal. This output signal is then used as an input in the next layer in the network.

Some commonly used activation functions are: -

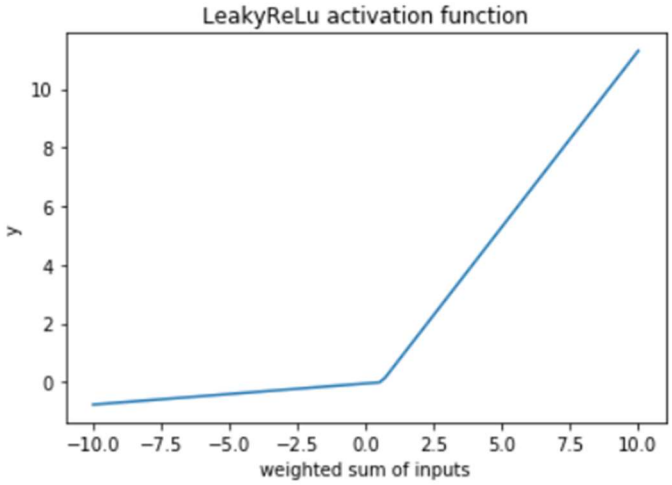
1. **ReLU -** The rectified linear activation function is a piecewise linear function that will return the number itself if the input is a positive number, otherwise, it will return 0, if the input is a negative number. It is defined as [10]

$$f(x) = \max(0, a) = \max(0, \sum_{i=1}^{i=n} w_i x_i + b)$$



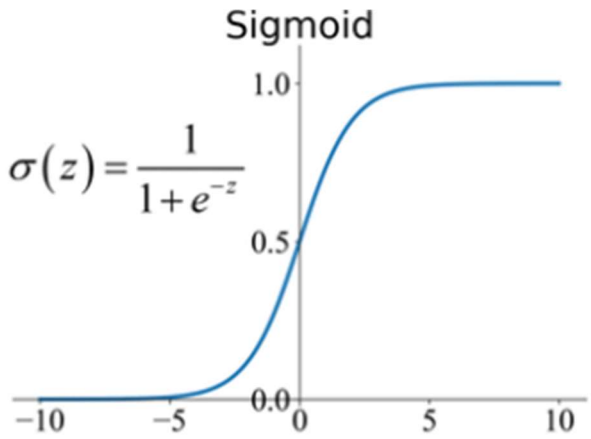
2. **Leaky ReLU** - Leaky ReLU function is an improved version of the ReLU activation function. If the input is a positive number, the function returns the number itself and if the input is a negative number, then it returns a negative value scaled by 0.01(or any other small value). It is defined as [10]

$$f(x) = \max(0.01a, a) = \max(0.01a, \sum_{i=1}^{i=n} w_i x_i + b)$$

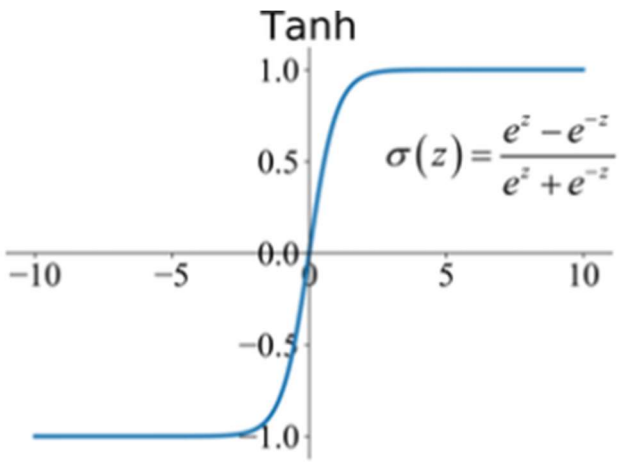


3. **Sigmoid** - The sigmoid activation function is also called the logistic function. It is traditionally a trendy activation function for neural networks. The input

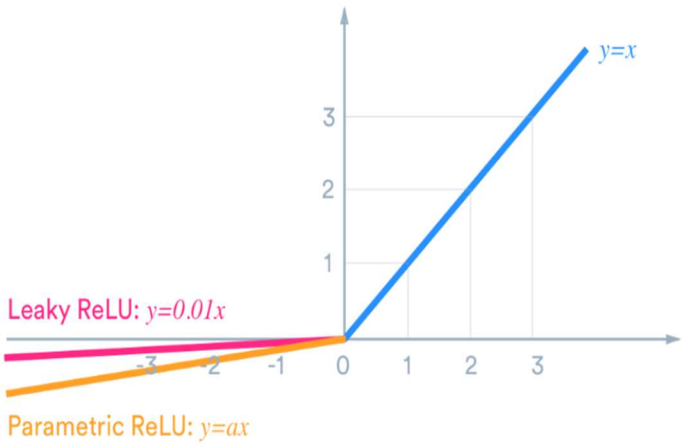
data to the function is transformed into a value between 0.0 and 1.0. It is defined as [10]



4. **Tanh** - The hyperbolic tangent function, also known as tanh for short, is a similar shaped nonlinear activation function. It provides output values between -1.0 and 1.0. It allows for better training and performance. It is defined as [10]



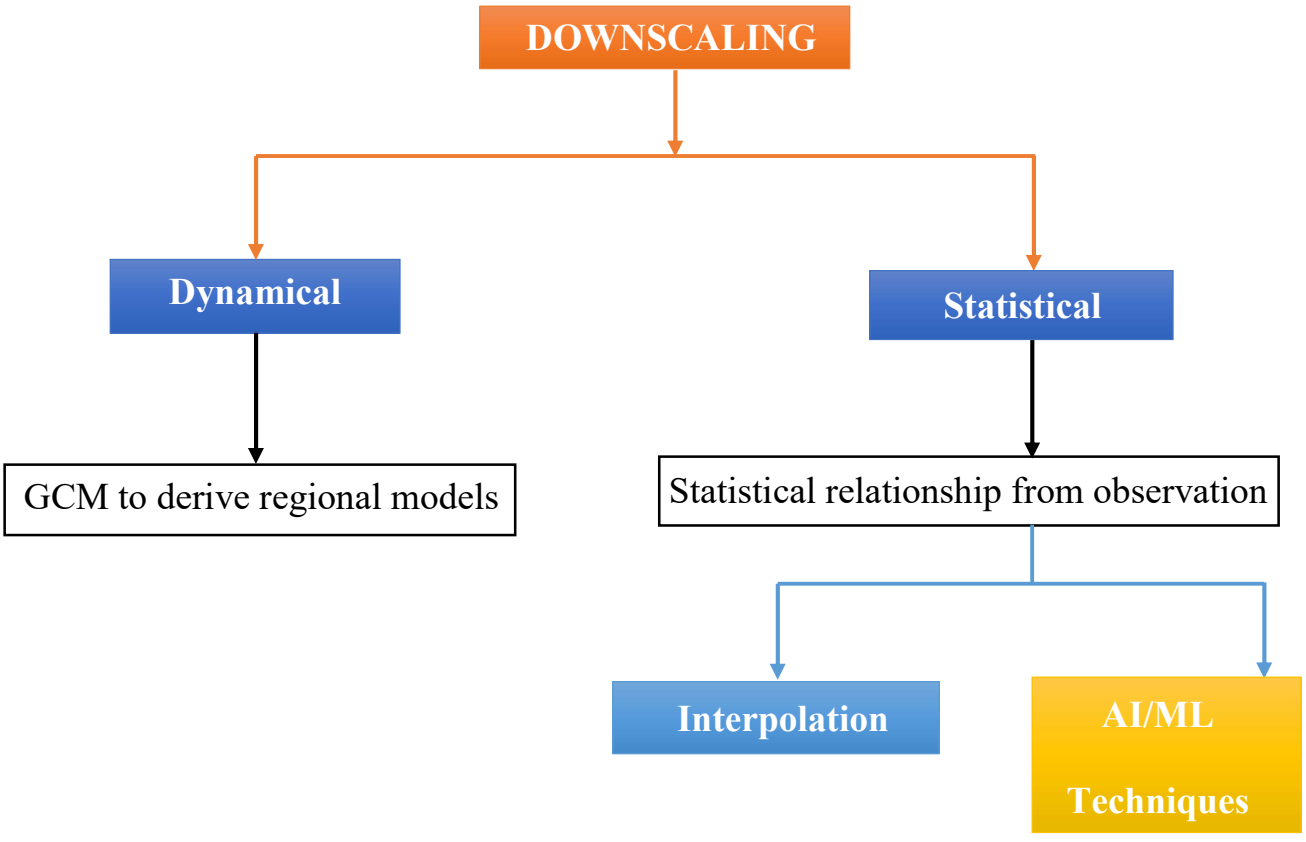
5. **Parametric ReLU (PReLU) –** The coefficient a is not lock at 0.01 (Leaky ReLU), but it is free to estimate. It's a generalisation of the ReLU, the algorithm learns the rectifier parameter. If a = 0, the parametric ReLU is equivalent to the ReLU activation function. If a = 0.01, the parametric ReLU correspond to the Leaky ReLU. [4]



DOWNSCALING:

Downscaling is the process of relocating coarse resolution Global Climate Models (GCM) to fine spatial scale. Its purpose was to bring the GCM model data in closer agreement with the station level data.[12] Downscaling is needed for –

- To generate high resolution data.
- To get local scale projections.



Downscaling techniques can be divided into two broad categories:

1. **Dynamical: -**
    - It refers to the use of high-resolution regional simulations to dynamically extrapolate the effects of large-scale climate processes to regional or local scales of interest.[12]
    - It uses the initial and boundary conditions from a global model and runs a high-resolution regional model to generate a local forecast.
2. **Statistical:** -
    - It encompasses the use of various statistics-based techniques to determine relationships between large-scale climate patterns resolved by global climate models and observed local climate responses.[12]
    - It uses the outputs from the global dynamical models or observations, as inputs to statistical models to generate high resolution information.
    - It is a popular low-cost method (or poor man's approach) to improve the rainfall information on regional scales. It increases the resolution to 8-12 times.

Background

SRCNN (Super resolution convolutional neural network): -

Dong et al. proposed SRCNN, which is the first deep learning model to solve super-resolution task.

In SRCNN, CNN for image classification is used for single image super resolution. Super-resolution refers to the process of upscaling or improving the details of the image. When increasing the dimensions of an image, the extra pixels need to be interpolated somehow.[3]

- SR methods aim to accurately estimate a high-resolution image (HR) from a given low resolution (LR) image. (Dong et al. 2014)
- A single low-resolution image is up scaled to the desired size using bicubic interpolation in SRCNN.
- Let interpolated image be Y and high-resolution image be X.

  Then, we need to learn a mapping F and F(Y) being the output, we have $F_1(Y) = \max(0, W_1 * Y + B_1)$, where W_1 and B_1 represent the filters and biases respectively, and '$*$' denotes the convolution operation.[3]

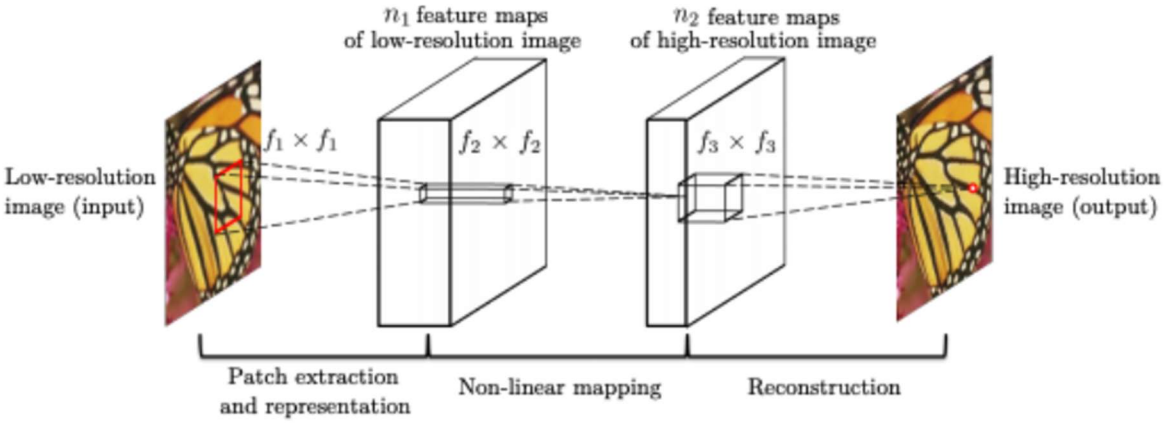
SRCNN surpasses the classical non-learning based bicubic baseline with just a few training iterations and outperforms the sparse-coding-based method (SC) with moderate training. The performance is further improved with more training iterations. [3]

- In 2014, Dong et al. (2014) first used convolutional neural network to propose SRCNN algorithm on single image super resolution.
- The algorithm reconstructs the image through **three-layer convolution layer network**:
    1. Image block extraction
    2. Feature representation
    3. Feature nonlinear mapping.

    The algorithm lays the foundation of convolution neural network applied to image super-resolution task.
- SRCNN performs better than conventional models on meteorological data (Vandal et al. IJCAI 18).

SR using SRCNN:

- Given a low-resolution image Y, first convolutional layer of SRCNN extracts a set of feature maps.
- Second layer maps these features nonlinearly to high resolution patch representations.
- The last layer combines predictions within a spatial neighbourhood to produce the final high resolution image F(Y).



- SRCNN comprises three convolutional blocks corresponding to patch extraction, non-linear mapping, and reconstruction.[7]
- The first layer convolves the input image with a 9x9 kernel with padding and expands the three-channel image into 64 feature maps.
- The second layer applies a 1x1 kernel to condense to 32 feature maps.
- The third layer applies a 5x5 kernel to generate the output image.
- Both the first and second convolutional layers are succeeded by a ReLU activation function.[7]

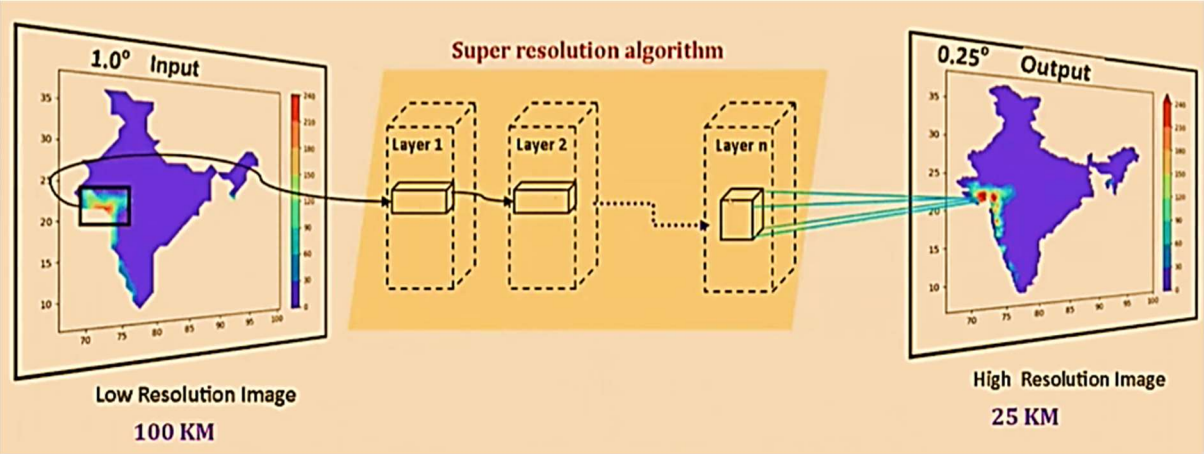
- Convolutional blocks can be treated as a subset of linear models with a small number of parameters.
    - The 9x9 kernels of the first SRCNN layer apply the same 81 parameters (values for each pixel of the kernel) to a full image, such that each pixel of the resulting image corresponds to the level of similarity between the corresponding 9x9 pixel patch of the input image and that kernel.[7]
    - The nonlinear mapping step connects these input feature maps to a reduced set of output feature maps.
    - The 5x5 kernels can be viewed as higher resolution/granularity image features represented by the pixels of the output feature maps.[7]
- For training and validation, mean squared error (MSE) is used for the loss function.[7]

$$L(\Theta) = \frac{1}{n}\sum_{i=1}^{n} ||F(\mathbf{Y}_i;\Theta) - \mathbf{X}_i||^2$$

For super resolution, the loss function L is the average of mean square error (MSE) for the training samples (n), which is a kind of standard loss function.

SRCNN on meteorological data:

- Single image super – resolution is the combination of classical and example-based SR techniques.
- Deep learning-based SR methods have achieved significant improvements, both quantitatively and qualitatively in meteorological data (Vandal et al. 2018).



Here, the low-resolution 1˚ x 1˚ spatial resolution IMD 2D data is used as input to the AI/ML network. Taken some patches and passed these patches to different hidden layers and found a patch for high resolution 0.25˚ x 0.25˚ image.[1]

Methods

SRCNN Algorithm:

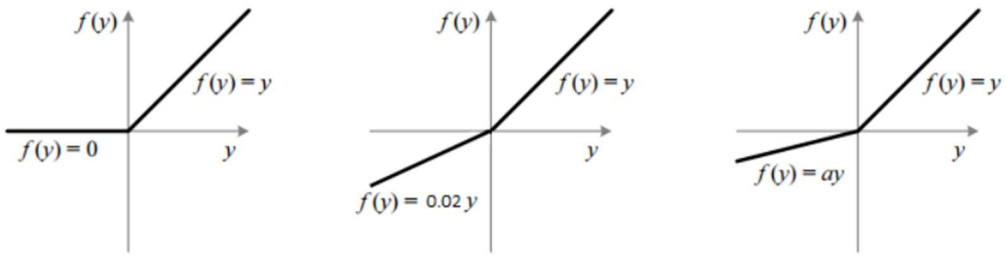
- Input is 1.0° data and label are 0.25° data. Label data means how the data is being trained.
- Sliced into small images or patches (Patch extraction)
    - 64 pieces (8 vertical and 8 horizontal) of size 35 x 35.
- This small image then enters the convolution layers
    - (3 layers) output image to a size of 23 x 23.
- CNN architecture can be designed to learn a functional mapping LR and HR using 3 operations-
    - Patch extraction
    - Non-linear mappings
    - Reconstruction
- First layer has 64 filters of kernel size of (9,9)
- Second layer has 32 filters → kernel size of (1,1)
- Third layer has 1 filter → kernel size of (5,5)
- Activation function PReLU is used in all the three layers.
- Parameters used in SRCNN:[4]
    - Activation function = **PReLU**

Activation function will take a set of inputs, perform some operations on them and return a set of corresponding outputs. This set of outputs lies in a range (generally 0 to 1 or -1 to 1) depending upon the function.

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}$$

Above, y_i is any input on the i^{th} channel and ai is the negative slope which is a learnable parameter.

- If $a_i = 0$, f becomes ReLU
- If $a_i > 0$, f becomes leaky ReLU
- If a_i is a learnable parameter, f becomes PReLU



(Left) ReLU, (Middle) LeakyReLU and (Last) PReLU

- a (slope parameter) can be learnt using backpropagation, as a result PReLU be able to adapt well to sudden changes.

- Kernel Initializers = GLOROT Normal, Random Normal, Truncated normal

- These are **used to statistically initialise the weights in the model**. This will generate the weights and distribute them; it can be used as the starting weights. [10]
- GLOROT Normal: Here, the weights are selected from a normally distributed range of values with mean (**μ)=**0 and standard deviation (**σ**)= √2/√Fan-in + Fan-out.[10]
- Random Normal: It's the initializer that generates tensors with a normal distribution.[5]
- Truncated Normal: It's the initializer that generates a truncated normal distribution.[5]

- Optimizer = ADAM
  - It is a combination of the 'gradient descent with momentum' algorithm and the 'RMSP' algorithm.
  - Root mean square propagation (RMSP) is an adaptive learning algorithm that tries to improve AdaGrad. Instead of taking the cumulative sum of squared gradients like in AdaGrad, it takes the '**exponential moving average**'. [13]

$$w_{t+1} = w_t - \frac{\alpha_t}{(v_t+\varepsilon)^{1/2}} * \left[\frac{\delta L}{\delta w_t}\right]$$

where,

$$v_t = \beta v_{t-1} + (1-\beta) * \left[\frac{\delta L}{\delta w_t}\right]^2$$

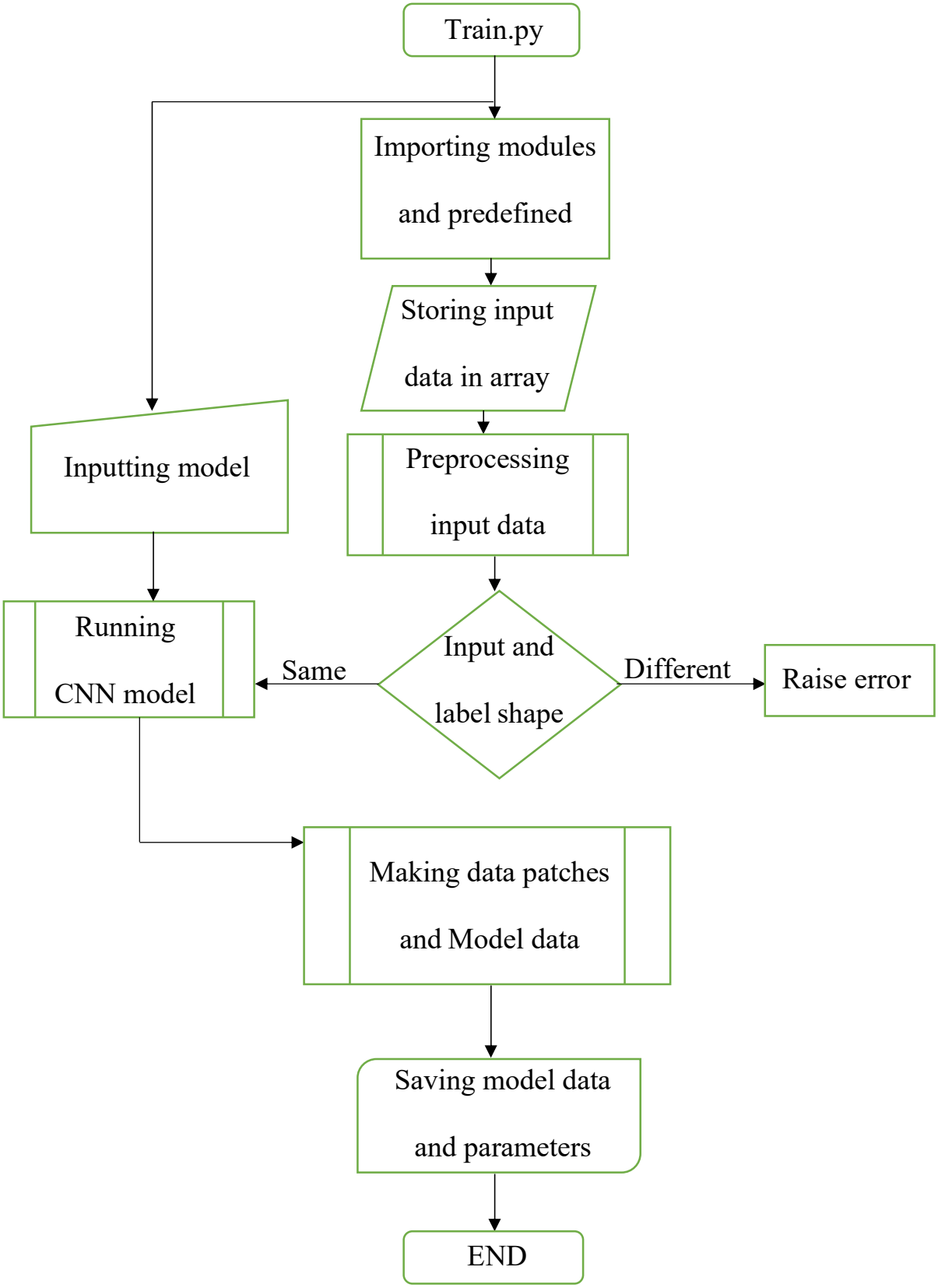
W_t = weights at time t
W_{t+1} = weights at time t+1
α_t = learning rate at time t
∂L = derivative of Loss Function
∂W_t = derivative of weights at time t
V_t = sum of square of past gradients. [i.e sum($\partial L/\partial Wt-1$)] (initially, $V_t = 0$)
β = Moving average parameter (const, 0.9)
ϵ = A small positive constant (10^{-8})

- Loss function = Mean squared error (MSE)
  - The loss function L is the average of MSE for the training samples (n) between a set of HR images and LR images.

$$L = \frac{1}{n}\sum_{i=1}^{n} ||HR_i - LR_i||^2$$

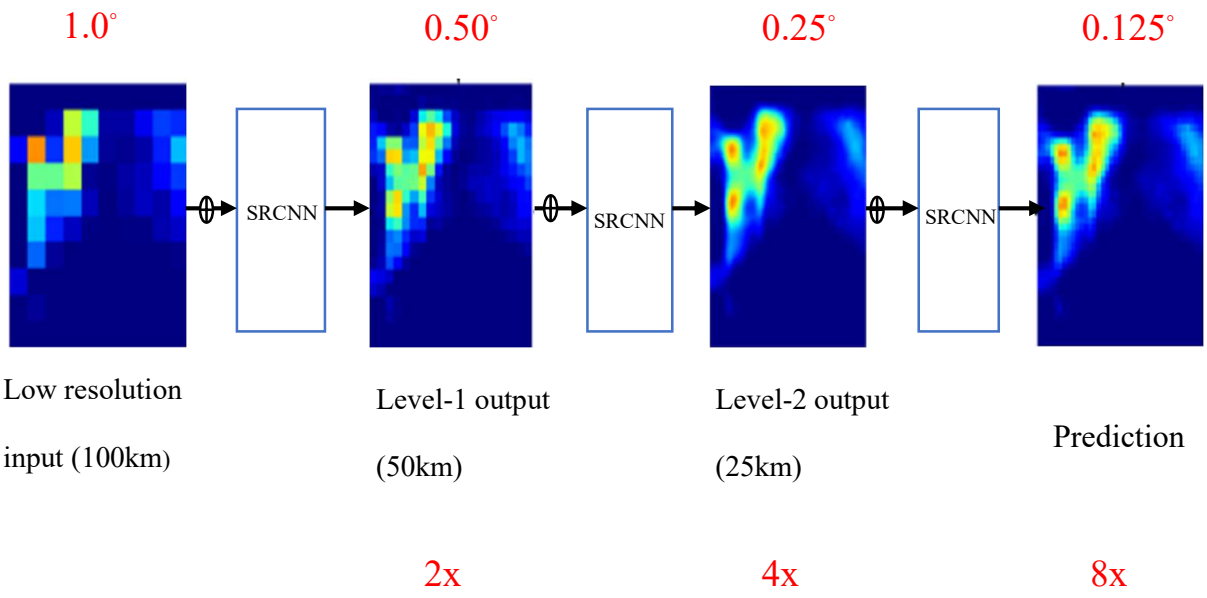
- Epoch = 500 (due to limitation of available resources) [5]
  - It signifies the number of times the training data is visible for a neural network to train. The training process is iterative to achieve efficiency.
- Batch size = 20
  - It denotes the size of the input data.[5]
- Learning rate = 10^{-4}
  - It denotes the time required for a network to update its parameters and learn.[5]
- No. of years of data = 10 Years (1995 – 2004)

Flowchart: Model Training



Stacked SRCNN:

- Traditional SR methods are built for resolution enhancements of factors from 2 to 4.
- Statistical downscaling conservatively requires resolution increases of factors from 8 to 12.
- Rather than enhancing resolution directly to 8 – 12x, "Stacked SRCNN" can be used to increase the resolution by a factor of 'x'.
- This allows the model to learn spatial patterns at multiple scales, requiring less complexity in the spatial representations.
- In this, each SRCNN is trained independently using their respective input/output resolutions and stacked at test time.[2]
- The ability of arbitrarily upscaling ground truth images to lower resolution allows for input/output pairs to be produced at multiple scales to train stacked SRCNNs.[2]
- However, while training a cascading model, the output of each SRCNN is the input to the following SRCNN, which may be leading to unnecessary error propagation through the network. Therefore, through experimentation it was found that cascading SRCNNs performed worse than stacked SRCNNs. [2]

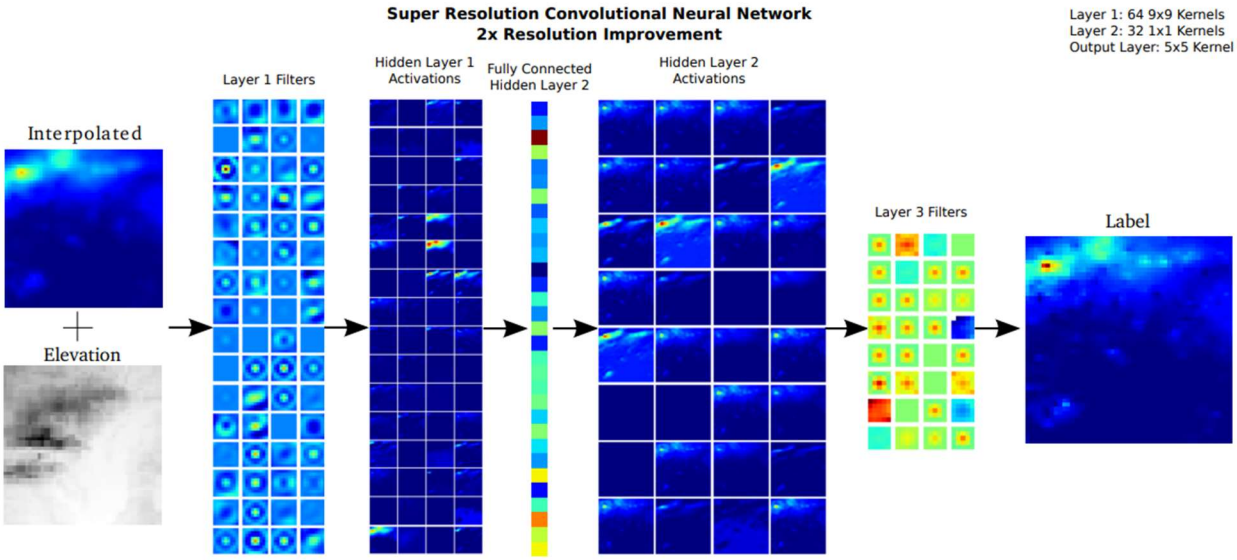


- The above figure shows the process of stacked SRCNN with a precipitation event and its various resolution improvements.
- In the above figure, inference is executed by starting with the lowest resolution image with its associated HR elevation to predict the first resolution enhancement. The next resolution enhancement is estimated from the previous layer's estimate and its associated HR elevation. This process is repeated for each trained SRCNN.[2]
- The above stacked process allows the model to capture both regional and local patterns.

DeepSD:

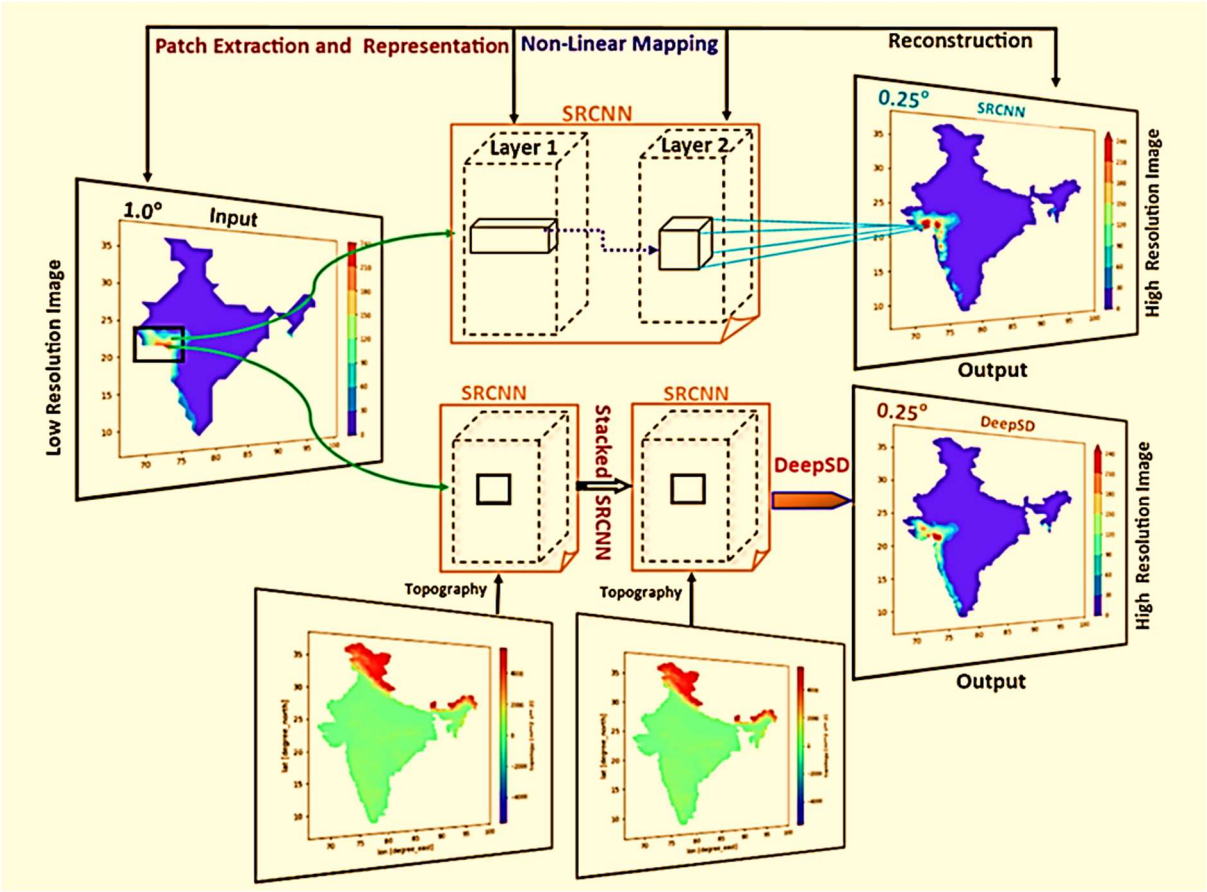
It is used for getting benefit of more variables. It is an augmented and specific architecture of stacked SRCNNs. So, it augments SRCNN with multiscale channels to maximise predictability in statistical downscaling.

- During statistical downscaling (SD), high-resolution data may be coinciding with the low-resolution image to estimate the high-resolution (HR) images. When downscaling precipitation, there are two types of inputs including low resolution (LR) precipitation and static topographical features such as HR elevation and land/water masks to estimate HR precipitation. [2]
- As topographical features are known beforehand at very high resolutions and generally do not change over the period of interest, they can be leveraged at each scaling factor.[2]
- As done when training stacked SRCNNs, each SRCNN is trained independently with its associated input/output pairs.



- In the above figure, a 3-layer CNN is constructed to produce a high-resolution estimate. Here, Layer 1 consists of 64 filters of 9x9 kernels, layer 2 consists of 32 fully connected neurons (1x1 filters), and the output layer uses a 5x5 kernel. Higher resolution models which have a greater number of sub-images, may gain from larger kernel sizes and an increased number of filters.

Overview: SRCNN – DeepSD



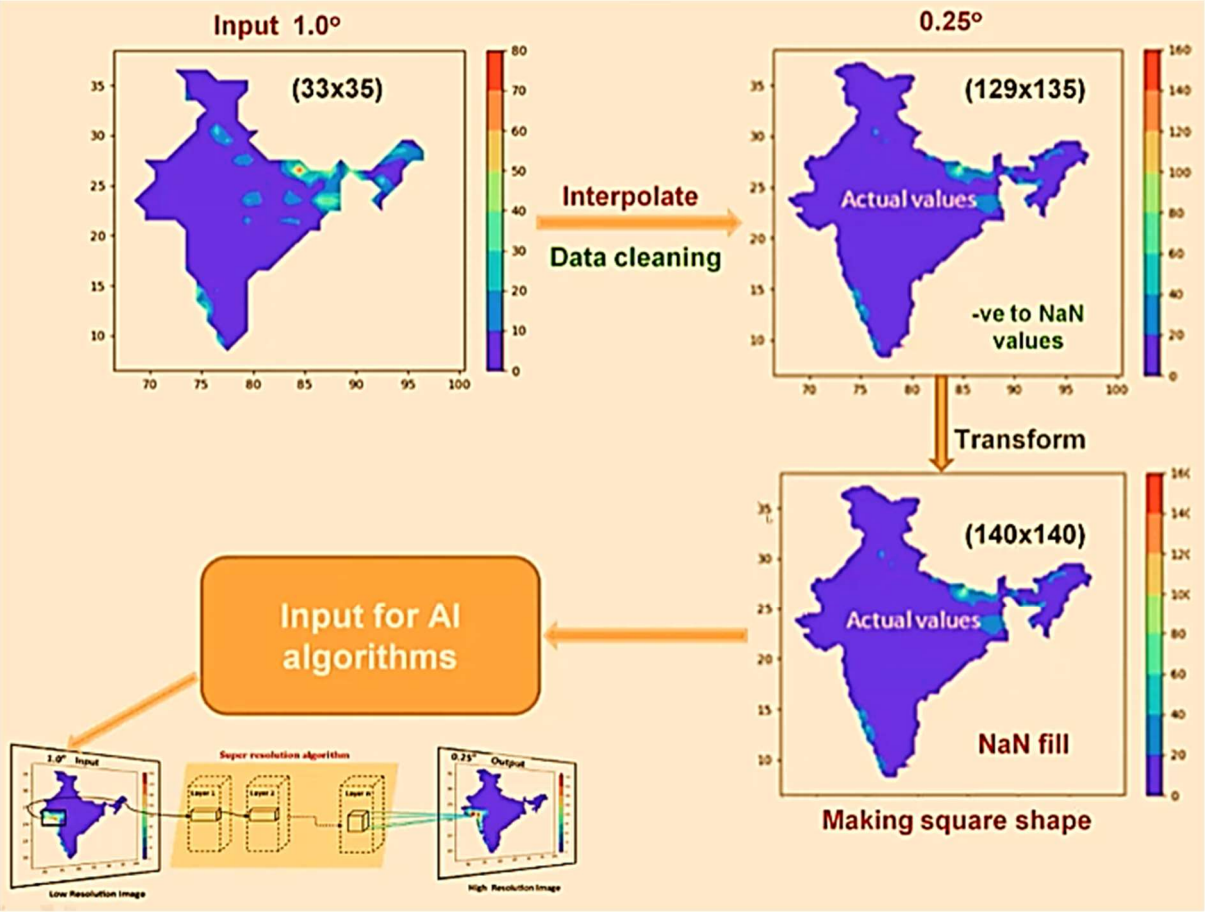
Have a SRCNN algorithm. Taken some of the rainfall data. Considered these rainfall data as image. Given this as an input to SRCNN, got a high-resolution image. Also, given this input image to stacked SRCNN to get the high-resolution image. But if given multivariable into the model training, will get the better results. [1]

Data Preprocessing

Target: Creating 0.25-degree data from 1.0-degree IMD gridded data

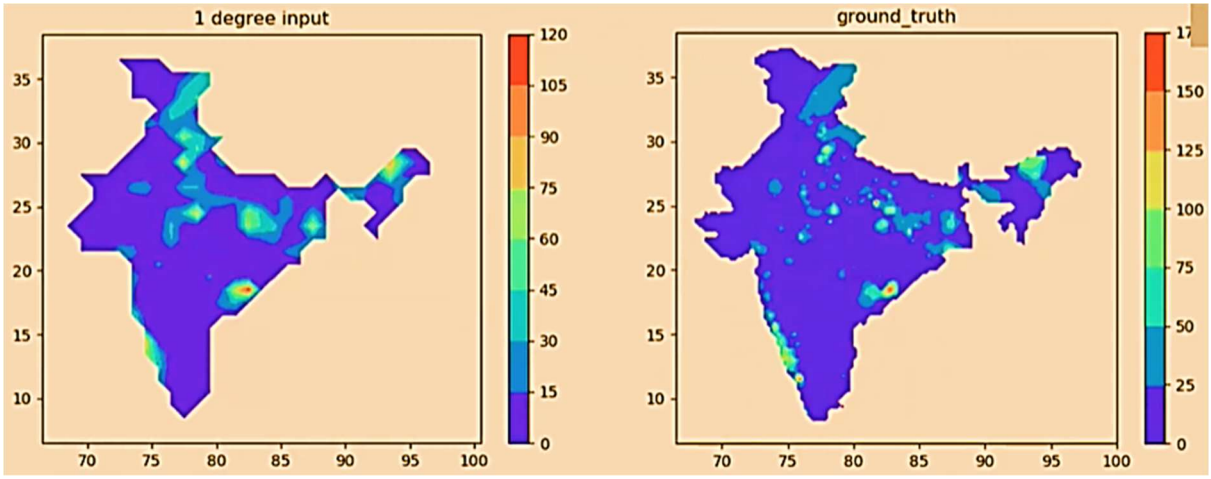
- Data used (IMD gridded data)
    - 1.0° (100km) – size is 33 x 35 (Input)
    - 0.25° (25km) – size is 129 x 135 (label or the ground truth)
- 1 degree data of matrix size 33 x 35 converted to the matrix size of 129 x 135 by bi-linear interpolation. Bilinear interpolation means applying a linear interpolation in two directions.[1]
- Converted all the values outside the Indian boundary to "nan" because we don't have those data accurate.
- We are making the whole input matrix as a square matrix. Because we are applying a convolution kernel and when we do convolution, we need the full matrix as a square matrix to cover all the patches nicely. So, converted both 1-0.25 degree interpolated and 0.25-degree ground truth to 140 x 140 filling "nan".[1]
    - The patches are overlapping to each other. In convolution we are sliding 1 step taking the kernel, as a result converting everything into a square matrix. It creates a lot of heterogeneity by taking an image.
    - 33 x 35 sub-images are extracted at a **stride of 23** to provide heterogeneity in the training set.[1]

- There are **64** sub images for a day.[1]
- Number of sub-images per year are **23360**.[1]



Training inputs:

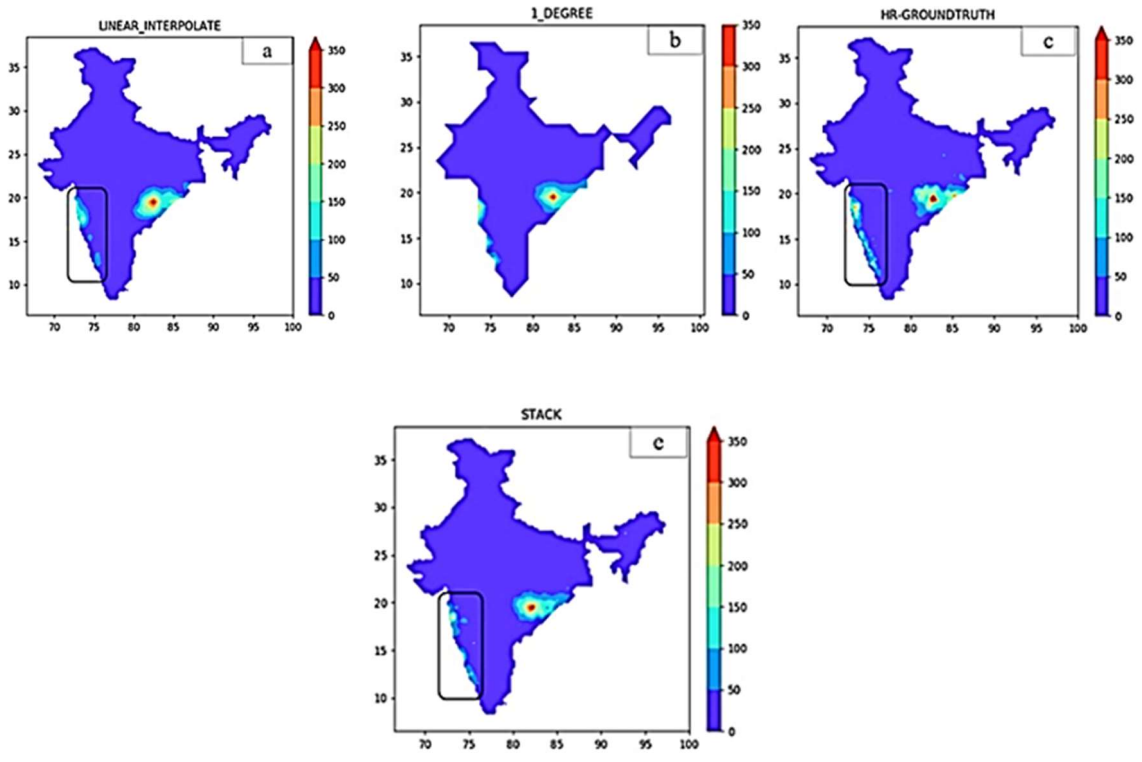
- Taken some 1-degree input and its ground truth is also used for comparing the output.
- Same input and label data are used for training Stacked SRCNN model.[1]



Results

- Stacked SRCNN has been applied.
- Data used
    - IMD Gridded Data
- The target was to generate HR image
    - $1.0^{\circ} \rightarrow 0.25^{\circ}$
- Resulted HR image compared with linear interpolation and ground truth.

<u>Comparison of results for a particular day:</u>

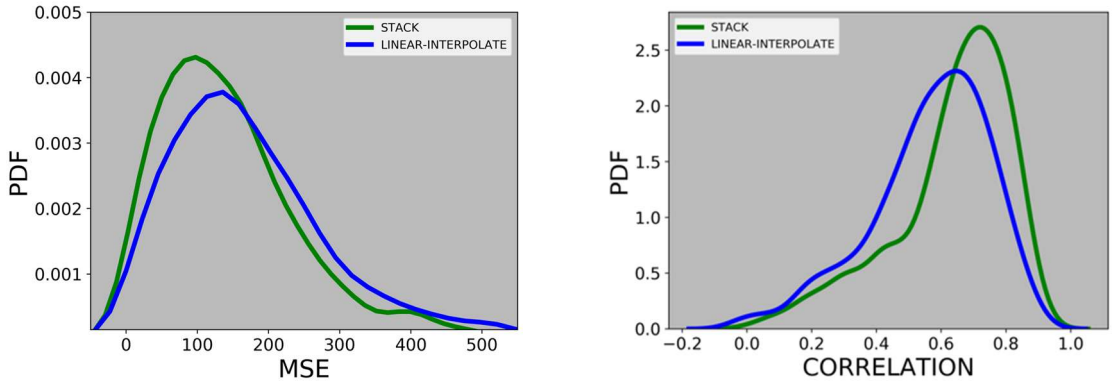


1_DEGREE is the input, and HR_GROUNDTRUTH is the actual image. On doing the linear interpolation, 1_degree image is obtained from HR_GROUNDTRUTH. In the groundtruth image, there are so many things in the

western ghat region that is not fully captured via linear interpolation. But some of the things are captured by the Stacked SRCNN even better.

Statistical analysis:

Seeing a particular day is not same as calculating the overall statistics.



- Did pattern correlation and generated the PDF and found that **correlation is better in Stacked SRCNN.**
- **MSE is less in Stacked SRCNN** compared to linear interpolation.

Conclusion

- Meteorological data is attempted to downscale using deep learning techniques.
- Stacked SRCNN is employed. To show better results, the technique must make use of multivariable, hence the method is called DeepSD.
- IMD 1 degree data is not from same stations as 0.25 degree.
- This can be used to produce higher resolution maps/data obtained from model/observation.
- Multivariable input is required during model training to improve the resolution by Stacked SRCNN.

References

[1] Deep learning–based downscaling of summer monsoon rainfall data over Indian region(https://doi.org/10.1007/s00704-020-03489-6)

[2] DeepSD: Generating High Resolution Climate Change Projections through Single Image Super-Resolution

[3] Image Super-Resolution Using Deep Convolutional Networks

[4] https://medium.com/@shauryagoel/prelu-activation-e294bb21fefa

[5] https://www.fireblazeaischool.in/blogs/deep-learning-interview-guide-for-2021/

[6] Image super resolution reconstruction based on improved invertible rescaling net

[7] Image Super-Resolution Via a Convolutional Neural Network

[8] http://deeplearning.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

[10] https://becominghuman.ai/basics-of-neural-network-bef2ba97d2cf

[11] Analyzing different types of activation functions in neural networks — which one to prefer? | by vikashraj luhaniwal | Towards Data Science

[12] https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[13] https://www.gfdl.noaa.gov/climate-model-downscaling/

[14] https://www.geeksforgeeks.org/intuition-of-adam-optimizer/