**Landslide Detection using various Deep Learning Architectures**

horizontal line

# 

# Introduction

In general, landslides are one of the natural earth surface processes and an example of land degradation; it changes features of landscape, reduces physical extent of the soil of ecosystems, causes to erosion and sediment yield and loss of soil resources, and subsequently damage to houses and basic infrastructures, agricultural lands, and economics and human welfare. Through destructive impacts of landslides and their consequences, research institutions and governments have long attempted to delineate unstable or landslide susceptible areas for improving disaster preparedness and damage prevention. Hence, the detection of slope stability not only provides an insight into control of land degradation; but, can also form a basis for safer strategic planning of future developmental activities in the region. Machine learning approaches are considered more efficient than other approaches such as expert's opinion based methods and analytic methods for spatial prediction of landslides. Main principle of these approaches is that slope stability is assessed using machine learning algorithms to analyze the spatial relationship between past landslide events and a set of conditioning factors from which the potential probability of landslide occurrence is determined. Many types of machine learning algorithms have been developed and applied for producing landslide susceptibility maps in many regions of the world. Literature review shows that the Random Forest and Boosted Regression Trees have the best performances in comparison to other MLTs (Pourghasemi and Rahmati, 2018). Similarly, a set of conditioning factors were taken, models such as Decision Tree Classifier, Random Forest Classifier, Naive Bayes Classifier, SVM and XGBoost Classifier were applied and it was observed thatXGBoost outperformed most of the models. Prior knowledge plays an important role in determining the accuracy in these methods. They are also time-consuming and labour-intensive, especially in large areas. Scientists, therefore, developed automated and semi-automated methods for remote sensing data to detect landslides. Researchers are constantly trying to untangle the combination of remote sensing images and topographical factors (e.g., altitude, slope, curvature) that best discriminate landslides from other objects. To the extent that landslides do not have unique spectral signatures and unique shapes, it may be useful to combine features from different modalities to detect landslides in complex areas. Both pixel and object-based methods have limitations in addressing this problem. Several methods based on deep learning have recently been proposed for the detection of landslides. (Chen *et al.*, 2018) used a deep CNN in combination with change detection methods for landslide mapping. Studies using deep learning to detect landslides are limited, and it is hard to make any conclusive statements about their performance compared to other pixel and object-based models. However, given their powerful ability to extract features from images, they may be a good choice for detecting landslides in data from different modes.

This research proposes a state-of-the-art model U-Net (U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015) to effectively detect landslides from high resolution satellite images. The main contribution of this study is designing a U-net architecture with few convolutional layers that can extract complementary features from satellite images and topographic factors.

1. **Literature Review**

(Pourghasemi and Rahmati, 2018) This research intend to present the first comprehensive comparison among the performances of ten advanced machine learning techniques (MLTs) including artificial neural networks (ANNs), boosted regression tree (BRT), classification and regression trees (CART), generalized linear model (GLM), generalized additive model (GAM), multivariate adaptive regression splines (MARS), naïve Bayes (NB), quadratic discriminant analysis (QDA), random forest (RF), and support vector machines (SVM) for modeling landslide susceptibility and evaluating the importance of variables in GIS and R open source software. This study was carried out in the Ghaemshahr Region, Iran

(Wang *et al.*, 2021) This paper proposes a novel machine learning and deep-learning method to identify natural-terrain landslides using integrated geodatabases.First, landslide-related data are compiled, including topographic data, geological data and rainfall-related data. Then, three integrated geodatabases are established; namely, Recent Landslide Database (RecLD), Relict Landslide Database (RelLD) and Joint Landslide Database (JLD). After that, five machine learning and deep learning algorithms, including logistic regression (LR), support vector machine (SVM), random forest (RF), boosting methods and convolutional neural network (CNN), are utilized and evaluated on each database.

(Pham *et al.*, 2016) This research is an assessment of the Uttarakhand area of India using five machine learning models namely SVM, Logistic Regression, Fisher’s Linear Discriminant Analysis, Bayesian Network and Naive Bayes and the performance of these methods has been evaluated using ROC curve and statistical index based methods .

(Wang, Fang and Hong, 2019) This research is to investigate a convolutional neural network (CNN) framework for landslide susceptibility mapping (LSM) in Yanshan County, China.

The two primary contributions of this study are summarized as follows: Firstly, this research describes the first time that the CNN framework is used for LSM. Secondly, different data representation algorithms are developed to construct three novel CNN architectures.

(Bhatt *et al.*, 2019) In this study, a deep learning approach to detect landslides is discussed. Convolutional Neural Networks are used for feature extraction for our proposed model. As there was no source of an exact and precise data set for feature extraction, therefore, a new data set was built for testing the model. They have tested and compared this work with their proposed model and with other machine-learning algorithms such as Logistic Regression, Random Forest, AdaBoost, K-Nearest Neighbors and Support Vector Machine .

(Sameen and Pradhan, 2019) This research designed residual networks for landslide detection using spectral bands and topographic information. Sameen and Pradhan compared a one-layer CNN with two of it’s deeper counterparts and residual networks with two fusion strategies to detect landslides in Cameron Highlands, Malaysia.

1. **About the Dataset**

This dataset is collected directly through the satellites. It contains **8 high resolution RGB(Red-Green-Black) images** which combine to form the whole region of the area of interest of working and are represented by **GEO1 to GEO8** in the dataset. The dataset also contains an image collected through a **DEM (Digital elevation model)** for the altitude information in the area of interest and it also contains a **Landslide image** which shows the landslide information for the region represented through pixel colors varying in the range of **0 to 255**. Although the DEM image and the landslide image covers a greater region than the combined GEO1 to GEO8 images, the region of interest was extracted from it later in the preprocessing step. Each image of GEO covers 15 minutes x 15 minutes geographical area and the original resolution of these images in pixels are as follows:

**GEO 1 - 27230 x 27392**

**GEO 2 - 27229 x 27136**

**GEO 3 - 27369 x 27392**

**GEO 4 - 27441 x 27392**

**GEO 5 - 27739 x 27392**

**GEO 6 - 27441 x 27392**

**GEO 7 - 27513 x 27392**

**GEO 8 - 27812 x 27392**

**DEM - 3201 x 4396**

**Landslide - 2369 x 3720**

1. **Data preprocessing :**

Image preprocessing is the term for operations on images at the lowest level of abstraction. These operations don't increase image information content but they decrease it if entropy is a system of measurement . The main aim of pre-processing is to improve the image data that suppresses undesired distortions or enhances the image features relevant for further processing while doing analysis tasks.

The preprocessing includes:

1. Changing the CRS(Coordinate reference system) of the images.
2. Making the resolution of all the images the same so as to accurately find the results.
3. Saving the images in GEOTIFF format.

The problem with using the raw data from satellites is that it has different resolutions for each of the bands. There is a need to convert the data in the same resolution so that all the data files are aligned correctly corresponding to every part of the area of interest so as to get accurate results.

The images are converted from **EPS:32644 WGS 84/ UTM zone 44N to EPS:4326 WGS 84 CRS format** using QGIS software and saved to images changing the CRS using the export function. GEO1 was taken as the standard resolution and rest of the images were scaled to the same. This step was also carried out using the tools provided in the QGIS software.

After converting all the images to the same resolution, the resulting images had the following resolution

**GEO1 - 27230 x 27392**

**GEO2 - 27230 x 27067**

**GEO3 - 27229 x 27252**

**GEO4 - 27230 x 27181**

**GEO5 - 27526 x 27182**

**GEO6 - 27230 x 27181**

**GEO7 - 27230 x 27110**

**GEO8 - 27526 x 27110**

**DEM - 111122 x 130183**

**Landslide - 82482 x 109904**

These images were saved in **GEOTiff format** and these files were used directly for the collection of the actual data and modelling so as to obtain the results.

1. **Dividing images into landslide and DEM as 8 parts in area :**

In this first step, the DEM as well as Landslide image were divided into its 8 parts with respect to its area. This step was done manually with the help of QGIS Software.

QGIS functions allowed us to analyze and edit spatial information, in addition to composing and exporting graphical maps. QGIS supports both raster and vector layers; vector data is stored as either point, line, or polygon features.

To do such division of image, the raster clipping tool was used and the resultant image was fed with the dimensions of the image to be clipped from a larger image or the area coordinates of the region of interest, which was the area covered in the GEO1 to GEO8 images. The large DEM and Landslide image was given as the input and provided the respective GEO images one by one to generate landslide1 to landslide8 images and DEM1 to DEM8 images. Hence, at the end of this process a total number of 8 images were obtained from the DEM images which contained the altitude information of the respective areas covered in the corresponding GEO images and 8 images from the original landslide image which consist the landslide information or the class label information corresponding to each of the GEO images.

1. **Checking if all the new GEO images overlap with DEM and landslide images**

In this step, There was a need to check if all the areas clipped in the new images were equal corresponding to the same dem and landslide image. If that's not the case the data would have generated inaccurate predictions. This was ensured manually by comparing the coordinates the height and the width of the images in the QGIS software as well as using python code comparing the height and width of the images to carry out the next step. If the code ensures that the data follows our condition we then proceed to the next step which is to clip the images using the sliding window technique.

1. **Clipping 16x16x4 images having 4th band as DEM and label as landslide**

At the starting of this phase, a total 24 images (8 images for each area, Landslide and DEM) were there. These were broken down as explained in the following steps

1. **Clipping Images:**

For clipping images into the size of 16x16 with available bands, sliding window technique was used. Sliding windows is one naive but effective approach to detect the objects within the surroundings. The image was scanned and some parts of it were taken as input to a pre-trained classifier to see if it observes in it, what it's trained to observe.

Here, the crop\_to\_bounding\_box() method provided by tensorflow was used to crop an image to a specified bounding box . This function by tensorflow uses the distance of the pixel values from the left and the upper boundary and the target area of the desired image to crop the desired image from a larger input.The top-left corner of the bounding box is at offset\_height, offset\_width in image, and therefore the lower-right corner is at **offset\_height + target\_height, offset\_width + target\_width**.

All the code for this was written in python using libraries namely tensorflow and keras which is later used for modelling. The function was fed with all the images one by one in a batch of 3 (For example : [GEO1,DEM1,lLandslide1]) to obtain the corresponding cropped images. The target area for each of the cropped images is **16 x 16 pixels** which are stacked with the DEM cropped images in the next step.

1. **Stacking Dem:**

Stacking of images here means to add the pixel value at axis level with another image. There were dem images with respect to the area image, dem was stacked as 4Th band using the channel wise concatenation of the imagesthe python function **concatenate() in NumPy Module** was used for this purpose and the cropped DEM 16 x 16 image was stacked with the GEO 16 x 16 image corresponding to the same area. To ensure the same area coordinates for both of the aspects,the images were enumerated with labels and the same number was later taken to recombine.

Thus, 16 x 16 x 4 channel images were extracted for every part of the smaller region in the data. After this, the images were assigned labels.

1. **Assigning Labels:**

Now that **16 x 16 x 4 channel images** were present corresponding to every part of the area of interest. With respect to the area of interest there were also landslide cropped images with the same height and width but with one channel as it was a grayscale image and according to the pixel values assignment of the label was done. The value ranges from 0-255 where,

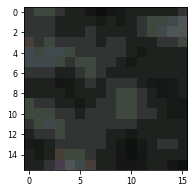
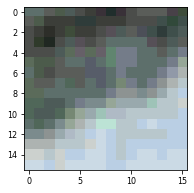
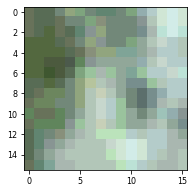
**255 – white : the areas where the landslide does not occur and assigned the label 0.**

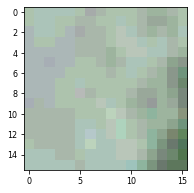
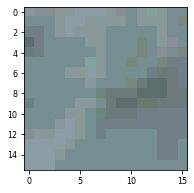
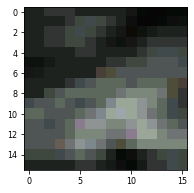
**0 - 254 : the gray and the black areas used as the areas where the landslide occurs and assigned the label 1.**

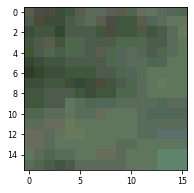
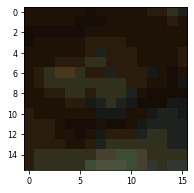
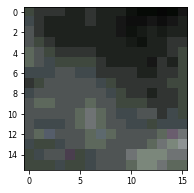
Since this is a binary classification there were two different labels which were used to classify the data (later described in the modelling section).

These images were stored with label 0 as non-landslide and 1 as landslide.The images are stored with a number appended to the label which it is assigned to then directly the label was used during the modelling from the image name itself. This eliminated the need to create an additional file to map each image to its classification label.

The following is a sample from the image.







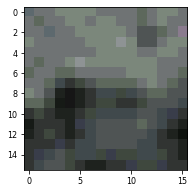
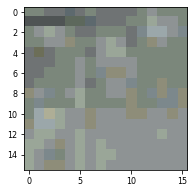
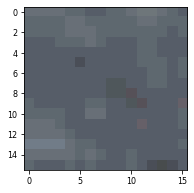


Fig. 1. Landslide view with 4 (RGB+DEM) Bands.

Fig. 2. Non-Landslide view with 4 (RGB+DEM) Bands.

1. **Sampling**

In this phase, there were a total of **4599913** images in which **4565528** images were of **non-landslide** and **34385** were of **landslides**. Here, the data related to landslides was very less as compared to the non-landslide data i.e. the problem of Imbalanced class distribution.

Imbalanced classification involves a dataset where the category distribution isn't equal. The amount of examples that belong to each of the classes within the training dataset varied widely. it's not uncommon to possess a severe skew within the class distribution, like 1:10, 1:1000 or maybe 1:1000 ratio of examples within the minority class to those within the majority class.

Although often described in terms of two-class classification problems, class imbalance also affects those datasets with quite two classes which will have multiple minority classes or multiple majority classes.

This is a drag because the minority class is strictly the category that we care most about in imbalanced classification problems.

The reason for this is often because the bulk class often reflects a traditional case, whereas the minority class represents a positive case for a diagnostic, fault, fraud, or other sorts of exceptional circumstance.

The most popular solution to an imbalanced classification problem is to vary the composition of the training dataset.

Techniques designed to vary the category distribution within the training dataset are generally mentioned as sampling methods or resampling methods as we are sampling an existing data sample.

The reason that sampling methods are so common is because they're simple to know and implement, and since once applied to rework the training dataset, a set of ordinary machine learning algorithms can then be used directly.

This means that any of tensor many machine learning algorithms developed for balanced (or mostly balanced) classification can then be fitted on the training dataset with none modification adapting them for the imbalance in observations.

This can be done in two ways:

**· Oversampling** methods duplicate examples within the minority class or synthesize new examples from the examples within the minority class.

**· Undersampling** methods delete or select a subset of examples from the bulk class.

Here, Random Undersampling was used to degrade the size of the larger (non-landslide) class. Random undersampling involves randomly selecting examples from the bulk class to delete from the training dataset.

This also had some effects of removing the important instances of majority class within the transformed version of the training dataset. This process is often repeated until the required class distribution is achieved, like an equal number of examples for each class.

For example, a dataset with 1,000 examples within the bulk class and 100 examples within the minority class are getting to be undersampled such that both classes would have 100 examples in the transformed training dataset.

For this, all the images were loaded with the label of the major class, shuffled and finally the number of images as the minor class possessed were obtained i.e.; 34377 images randomly selected from 4599913.

1. **Train-Test split of images:**

The train-test split procedure was employed to estimate the performance of machine learning algorithms to make predictions on data which was not used to train the model.

It was a quick and straightforward procedure, the results of which permitted it to match the performance of machine learning algorithms for predictive modelling of the problem. Although simple to implement and interpret, there are times when the procedure shouldn't be used, like with a little dataset and situations where additional configuration is required, like when it's used for classification and therefore the dataset isn't balanced.

In this phase, there were a total 68762 images in which 34377 images were of non-landslide and 34385 were of landslides i.e.; a 1:1 ratio for both categories. Here, the data set was split into 70:30 for the train set and the test set respectively.

Steps followed for splitting:

1. Loading of all cropped Images
2. Collecting all images at one
3. Shuffling the images
4. Split with **70:30**

In this way, there were 48132 images in the train and 20630 images in the test.

1. **Performance Evaluation Metrics**

After doing the usual Data collection, Preprocessing, implementing a model and getting some output in forms of a probability or a class, the next step was to find out how effective the model was based on suitable metrics using test datasets. Different performance metrics were used to evaluate different Machine Learning Algorithms. Classification performance metrics namely Accuracy, F1 Score, MCC and AUC ROC were used.

**Accuracy**: It is the ratio of the number of correct predictions to the total number of input samples.

It works well only if there are an equal number of samples belonging to each class.

**F1 Score**: F1 Score is the harmonic mean between precision and recall. It tells you how precise your classifier is (how many samples it classifies correctly), as well as how robust it is (it does not miss a significant number of samples).

The F1 score represents a more balanced view but could give a biased result in some scenarios since it doesn’t include TN.

**MCC**: Unlike the other metrics discussed above, MCC takes all the cells of the confusion matrix into consideration in its formula.

If you are looking at a metric to measure and maximize the overall accuracy of the classification model, MCC seems to be the best bet since it is not only easily interpretable but also robust to changes in the prediction goal.

**AUC ROC**: AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive sample higher than a randomly chosen negative sample. AUC is the area under the curve of plot False Positive Rate vs True Positive Rate at different points in [0, 1].

1. **Various Models**

After preparation of data, modelling of the data was done using various different CNN(Convolutional Neural Network) architectures. A convolutional neural network (CNN) is a type of [artificial neural network](https://searchenterpriseai.techtarget.com/definition/neural-network) used in [image recognition](https://searchenterpriseai.techtarget.com/definition/image-recognition) and processing that is specifically designed to process pixel data.

CNNs are powerful image processing, artificial intelligence ([AI](https://searchenterpriseai.techtarget.com/definition/image-recognition)) that use deep learning to perform both generative and descriptive tasks, often using machine vision that includes image and video recognition, along with recommender systems and natural language processing (NLP).

Here in the modelling, the data was fed to the various models that were chosen from various different machine learning and Deep Learning landslide detection research papers and about the procedure applied to solve the specific problem of Landslide detection using Machine learning and Deep Learning concepts. Since the data set is of images, Neural networks were preferred more precisely Convolutional NN architectures since they perform well with images as compared to the classical machine learning models.

Some of the research papers also focussed on the classical machine learning model like SVMs (Support vector models) and tree based models but most of them used CNNs to solve the problem.

Moving further as the data preparation steps are completed in the steps above, there was a need to decide the model architectures that were to be used.There were 2 models selected from a research paper (Wang *et al.*, 2021) which used a similar stacking procedure as the one we used. One of the CNN architectures was a small network with less number of layers and the other was a deep convolutional neural network which used a greater number of layers. We used both of the models with respect to the data we prepared.

**Resnet with Fusion**

Sameen et al. in (Sameen and Pradhan, 2019) gave ResNet based model with a small number of parameters which gave better results than many pre-existing models.

The model used a total of 7 layers.The first being a convolution layer with 8 filters, of size 3X3 and ReLU activation function with same padding. It was followed by a Max Pooling Layer with 8 filters with a size of 2X2 and same padding. Then a batch normalization was done and ReLU activation was done to it. It is then followed by a Convolution Layer similar to the previous one followed by Batch Normalization Layer and ReLU. Then, another similar Convolution was done. The output from this layer was added to the output from the Max Pooling Layer. This helped in giving more weight to lower level features discovered in the previous layer. Then batch normalization and relu activation of the output was. Now, Global Average Pooling of the output was obtained which extracted the average of each layer giving us just 1 value from each layer, hence just 8 variables. A dropout at this layer was added with a probability of 0.3 to avoid overfitting. This was then fed to the final layer i.e. Dense layer with sigmoid activation function.

The model is really light weight in terms of computational complexity. It has a total of 1578 parameters. It was trained for 100 epochs ,with a batch size of 128 and decreasing the Learning rate using ReduceLROnPlateau callback from keras Each epoch took about 5 seconds.

RESULTS (ON TRAIN DATA):

**Accuracy : 68.8%**

**F1 Score : 0.687**

**Matthews correlation coefficient : 0.376**

**ROC AUC : 0.688**

RESULTS (ON TEST DATA):

**Accuracy : 68.6%**

**F1 Score : 0.687**

**Matthews correlation coefficient : 0.372**

**ROC AUC : 0.686**

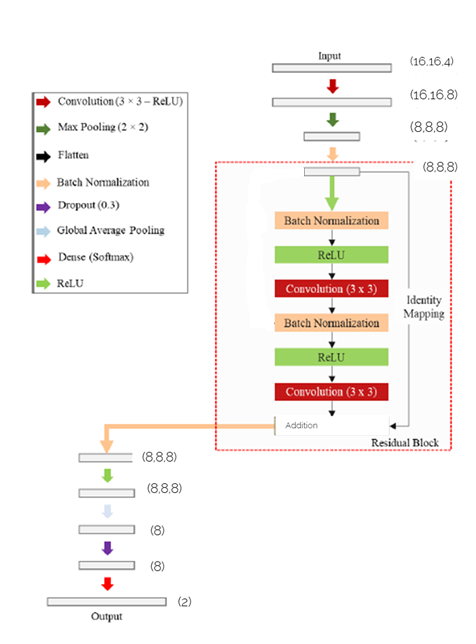


Fig. 3. Architecture of Resnet Model with 7 Layers.

**CNN-6**

This model is from the paper (Wang *et al.*, 2021) by Wang et al. which used a total of 6 layers with (3x3) kernel size and same padding with every convolution layer as well as stride = 1 in every convolution layer. It also uses 2 max Pooling layers one with stride 1 and the second one with stride 2.The activation function used is ReLu(Rectified linear activation function). The filter size for layer one is 64 and the filter size for the second convolution layer is 128. A dropout layer was added in between the Densely connected layers with a dropout rate of 0.5. The filter size for densely connected layers are 256 and 2 respectively. The activation function for the last dense layer is sigmoid. The loss used is binary cross entropy or log loss for binary classification and the metrics defined above in the metrics section were used.

The total parameters with the model is 1,682,626 .It was trained for 100 epochs ,with a batch size of 128 and decreasing the Learning rate using ReduceLROnPlateau callback from keras Each epoch took about 5 seconds.

RESULTS (ON TRAIN DATA):

**Accuracy : 66.5%**

**F1 Score : 0.652**

**Matthews correlation coefficient : 0.331**

**ROC AUC : 0.665**

RESULTS (ON TEST DATA):

**Accuracy : 66.1%**

**F1 Score : 0.648**

**Matthews correlation coefficient : 0.325**

**ROC AUC : 0.662**

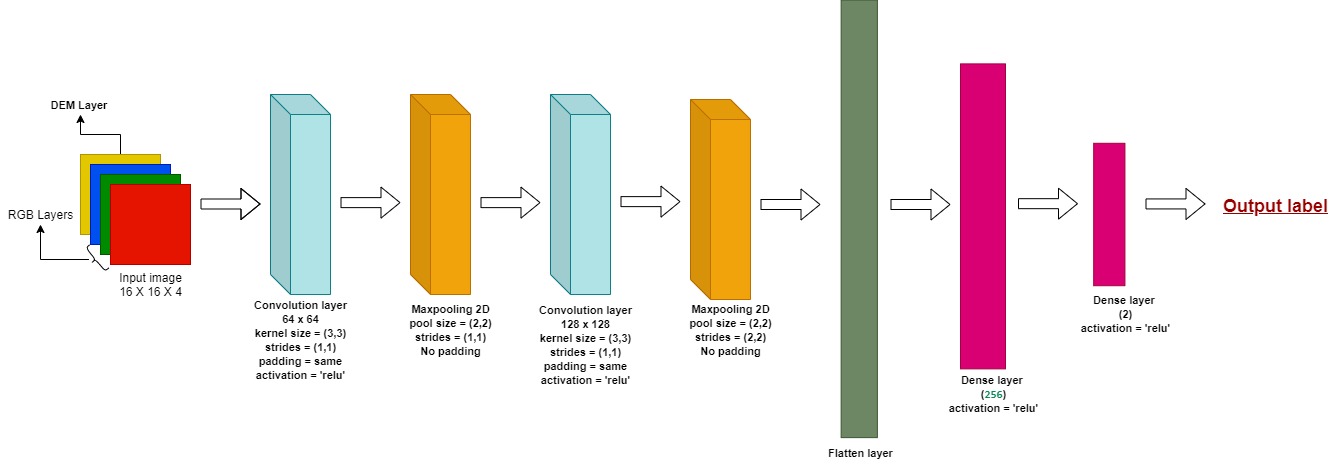
****

Fig. 4. Architecture of Deep CNN Model with 6 Layers

**DCNN-11**

This model is from the paper (Wang *et al.*, 2021) by Wang et al. which used a total of 11 layers with (3x3) kernel size and same padding with every convolution layer as well as stride = 1 in every convolution layer. It uses 4 max

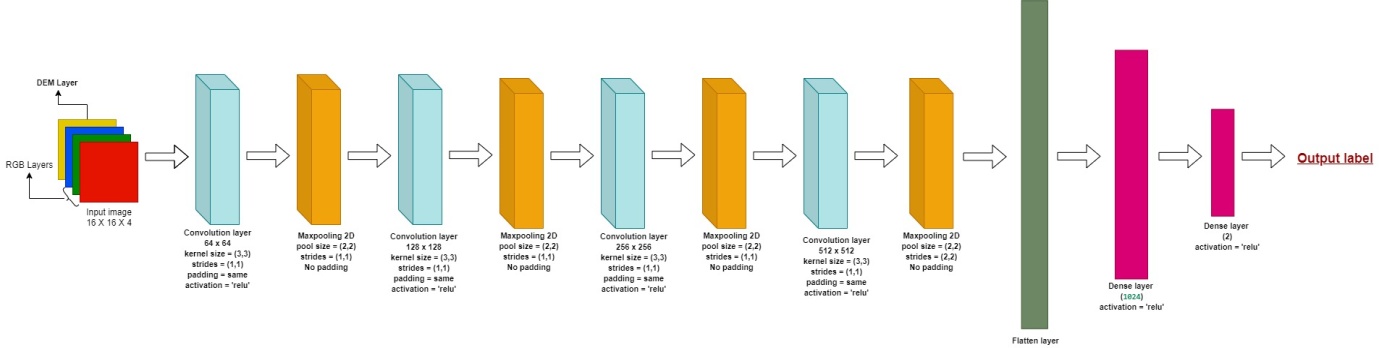


Fig. 5. Architecture of Deep CNN Model with 11 Layers

Pooling layers each with stride 1 and the last one with stride 2.The activation function used here is also ReLu. The filter size for layers are 64 for first,128 for second,256 for third and 512 for fourth .A dropout layer in between the Densely connected layers with a dropout rate of 0.5 was inserted. The filter size for densely connected layers are 1024 and 2 respectively.The activation function for the last dense layer is sigmoid.

The loss used is binary cross entropy and the metrics used are as previously mentioned in the section 5. The total parameters with the model is 15,711,426 which is larger than the previous smaller model .It was trained 100 epochs and each epoch took about 18 seconds which is three times the time for each epoch compared to each epoch in the CNN-6. The batch size used is 128 and decreasing the learning rate using ReduceLROnPlateau callback from keras.

RESULTS (ON TRAIN DATA):

**Accuracy : 85.7%**

**F1 Score : 0.848**

**Matthews correlation coefficient : 0.721**

**ROC AUC : 0.857**

RESULTS (ON TEST DATA):

**Accuracy : 69.4%**

**F1 Score : 0.673**

**Matthews correlation coefficient : 0.391**

**ROC AUC : 0.693**

Although the results on this model are better than the previous one, looking at the stats from test and train data it can be concluded that the model might be overfitting to achieve the results we see. So this model is not appropriate.

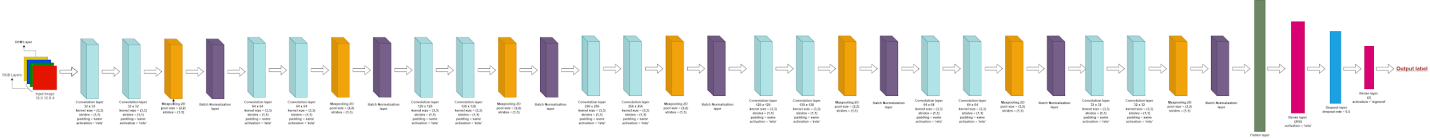
**U-Net Architecture**

After exploring much U-Net based model architecture was found. It is a [convolutional neural network](https://en.wikipedia.org/wiki/Convolutional_neural_network) that was developed for biomedical [image segmentation](https://en.wikipedia.org/wiki/Image_segmentation) at the Computer Science Department of the [University of Freibur](https://en.wikipedia.org/wiki/University_of_Freiburg)g. The network is based on the fully convolutional networkand its architecture was modified and extended to work with fewer training images and to yield more precise segmentations. The main idea was to supplement a usual contracting network by successive layers, where pooling operations are replaced by [upsampling](https://en.wikipedia.org/wiki/Upsampling) operators. Various U-net model architectures were tried on the data and the most appropriate model was chosen which gave the best results as compared to the other models. It also has more layers as compared to the other architectures used for modelling previously in this paper.

The U-Net architecture uses a total of 30 layers with (3x3) kernel size and same padding with every convolution layer as well as stride = 1 in every convolution layer. It uses 7 max Pooling layers each with stride 1. The activation function used here is also ReLU.

The filter size in a U-Net type architecture has two phases,the expanding phase which has an increasing number of filters with every layer and the contracting phase which has a decreasing number of filters with every layer. The filter size of layers in the expanding phase are 32 for first two,64 for next two ,128 for next two and 256 for the next layer.Now in the contracting phase the number of filters start to decrease from 256.The next layer has 256 filters ,128 for the next two layers,64 for the next two and 32 for the last two convolution layers . A dropout layer was used in between the Densely connected layers with a dropout rate of 0.5. The filter size for densely connected layers are 256 and 2 respectively .The activation function for the last dense layer is sigmoid. The model also uses 7 Batch Normalization layers each after the max pooling layers.

The loss used is binary cross entropy and the previously mentioned metrics to judge the performance of the model. The total parameters with the model is 2,420,738 which is much higher compared to the other models mentioned in this paper .This model was trained for 100 epochs and each epoch took about 21 seconds which is greater than the time for each epoch compared to each epoch in the previous models. The batch size used is 128 and decreases the Learning rate using ReduceLROnPlateau callback from keras.



RESULTS (ON TRAIN DATA):

Fig. 6. Architecture U-Net used

**Accuracy : 77.5%**

**F1 Score : 0.773**

**Matthews correlation coefficient : 0.55**

**ROC AUC : 0.774**

RESULTS (ON TEST DATA):

**Accuracy : 74.5%**

**F1 Score : 0.745**

**Matthews correlation coefficient : 0.491**

**ROC AUC : 0.745**

1. **Results**

Table 1. Result Analysis of various Models

| **Model** | **Performance on Train Data** | | | | **Performance on Test Data** | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Accuracy | F1 Score | MCC | ROC AUC | Accuracy | F1 Score | MCC | ROC AUC |
| **RP** | 68.8% | 0.687 | 0.376 | 0.688 | 68.6% | 0.687 | 0.372 | 0.686 |
| **Smaller** | 66.5% | 0.652 | 0.331 | 0.665 | 66.1% | 0.648 | 0.325 | 0.662 |
| **Larger** | 85.7% | 0.848 | 0.721 | 0.857 | 69.4% | 0.673 | 0.391 | 0.693 |
| **Unet** | 77.5% | 0.773 | 0.55 | 0.774 | **74.5%** | **0.745** | **0.491** | **0.745** |

**References:**

Bhatt, J. *et al.* (2019) ‘A research on deep learning advance for landslide classification using convolutional neural networks’, *International Journal of Innovative Technology and Exploring Engineering*, 8(6 Special Issue 4), pp. 903–906. doi:

10.35940/ijitee.F1184.0486S419.

Chen, Z. *et al.* (2018) ‘Automated landslides detection for mountain cities using multi-temporal remote sensing imagery’, *Sensors (Switzerland)*, 18(3). doi: 10.3390/s18030821.

Pham, B. T. *et al.* (2016) ‘A comparative study of different machine learning methods for landslide susceptibility assessment: A case study of Uttarakhand area (India)’,

*Environmental Modelling and Software*, 84, pp. 240–250. doi: 10.1016/j.envsoft.2016.07.005.

Pourghasemi, H. R. and Rahmati, O. (2018) ‘Prediction of the landslide susceptibility: Which algorithm, which precision?’, *Catena*, 162(May), pp. 177–192. doi: 10.1016/j.catena.2017.11.022.

Sameen, M. I. and Pradhan, B. (2019) ‘Landslide Detection Using Residual Networks and the Fusion of Spectral and Topographic Information’, *IEEE Access*, 7, pp. 114363–114373. doi: 10.1109/ACCESS.2019.2935761.

Wang, H. *et al.* (2021) ‘Landslide identification using machine learning’, *Geoscience Frontiers*, 12(1), pp. 351–364. doi: 10.1016/j.gsf.2020.02.012.

Wang, Y., Fang, Z. and Hong, H. (2019) ‘Comparison of convolutional neural networks for landslide susceptibility mapping in Yanshan County, China’, *Science of the Total Environment*, 666, pp. 975–993. doi: 10.1016/j.scitotenv.2019.02.263.