Technische Universität Berlin
Faculty:
Geodesy and Geoinformation Science

Professor:
Prof. Dr. Martin Kada

Master Thesis

Proposal:

Lunar Crater Detection via Deep Learning and Validation at Apollo 17 Landing Site

Author:
Tahir, Waqas

Spervisors:
Amgad Agoub
Philipp Gläser

November 17, 2019

Contents

1 Introduction

Studying impact craters present a valuable information about the geology of planets and characteristics of the surface. Computation of crater density has provided a gateway for the establishment of evolution of planet terrains chronologically [Martins et al., 2009]. This study provide not only the information about geological processes of the planet but also the history of our solar system. According to geologists impact craters are the only modifying and surface forming process for other planets and moons of all planets [Koeberl, 1994]. Impact craters have also been a useful source of information for planetologist in providing the relative age of surfaces. If the relationship between crater size and impact energy is known and flux caused by the impact is understood then craters with larger densities indicate older surfaces [Ivanov, 2002].

Various processes alter crater populations specially of those with smaller diameters [Opik, 1965] more often in planets with an atmosphere. This results deviations in crater size-frequency distributions (CSFD). Factors like deceleration, ablation, fragmentation of meteors while passing through the atmosphere before striking the surface, the effect of target properties on scale of crater formation and postformation changes of craters by erosion and deposition play an important role in altering crater populations. Therefore, crater counts with relatively smaller diameters (i.e. $d < 1$ km) are at a larger risk of representing an age which could be misinterpreted if the modeled production function does not take into account the factors responsible for altering CSFD in an observed range of diameter [Hartmann, 1981]. Moreover, several factors may lead to surface age as well as statistical uncertainties because smaller crater's identification is prone to certain biases such as resolution limits, illumination effects, compact crater count areas or limited number of craters [Soderblom, 1970].

In case of earth's Moon, modeling of impact craters depends on the knowledge of age-dated lunar samples with correlation to observed CSFDs [Williams et al., 2018]. Chronology is a system composed of two elements such as a production function describing the CSFD shape and a chronology function which is related to the accumulation of craters density to absolute time. Both of these functions collectively provide a predicted CSFD or isochron for a given time length a surface has been under crater strikes. This is valid for lunar surface as Moon has no atmosphere. This chronology can also be applied to other solar system objects but with an addition of factors such as impactor flux and surface gravity [Ivanov, 2002].

If CSFD and the diameter of crater is known then age of the lunar surface can be determined using both of these parameters. It is only possible to determine the age of the same area where crater count has been performed. There are production functions proposed by planetary scientists which provide an approximation to surface age if CSFD and crater diameters are known. The examples of such functions are Hartmann Production Function (HPF) and Neukum Production Function (NPF).

Crater-counts gives the frequency of crater distribution of a certain area. This can be performed by counting craters manually. This task could not only be time consuming but also expensive and labor intensive depending upon the number of images and amount of craters present in images. Satellites are the source of images which provide opportunity to count craters depending on the type of cameras on board. Also number of craters in a certain area may change with a passage of time which would mean that possibly same task for the same surface has to be performed again.

Since early 2000 many missions and satellites have been deployed to Moon for research purposes including crater studies. LRO (Lunar Reconnaissance Orbiter) is one of the satellites on lunar mission operated by USA and is collecting lunar images since 2009. LRO has gathered millions of images since its launch. Lunar Reconnaissance Orbiter Camera (LROC) was specifically designed for the assessment of meter and smaller scale features to help carry out safety analysis for future lunar landing sites on moon including polar region. The Narrow Angle Cameras (NAC) mounted on LRO are able to detect craters with diameter of 2.5 m or greater [Robinson et al., 2010]. It is common to find craters less than 100 m in diameter on lunar surface [Robinson et al., 2010].

Images taken from LRO are large in size (resolution of 1.009). The data is available publicly and can be seen or downloaded through this link *(http://wms.lroc.asu.edu/lroc/)*. Such images cannot be processed without a competitive machine. There are methods developed to perform image processing tasks which are highly dependent on a machine's memory and performance. Therefore, developing techniques without advancement in computer technology was simply not enough. This problem was already known during 1960s and therefore it lead to the development of Graphical processing unit (GPU). A GPU performs quick math calculations and frees the space for CPU to do other tasks. Unlike CPU it has thousands of cores designed for multi-tasking. A much needed hardware to perform computationally expensive calculations without exhaustion.

In 2000s many companies like Intel, Nvidia and AMD/ATI stepped into the race of manufacturing faster GPUs and dominated market. This competition continues till today and GPUs are becoming more and more powerful. Meanwhile, the need for GPUs is also increasing because of big data processing needs. Nvidia introduced a chip capable of programmable pixel shaders i.e. compute color, position, depth or stencil of a pixel, the Ge-Force 3. A short program could now process each pixel before projecting it to the screen, this processing included but not limited to addition of image textures. By 2002 ATI Radeon 9700 the world's first Direct3D 9.0 accelerator was introduced my Microsoft containing support for multiple render targets, floating point texture formats, multiple-element textures and stencil buffer methods. In 2010 Nvidia began a partnership with Audi to power car dashboards. This mainly increased the functionality of navigation and entertainment systems. In 2014 the PS4 and Xbox One were released powered by GPUs based on AMD's Radeon HD 7850 and 7790. Lately RTX was released by Nvidia with the aim of enabling real time ray tracing. This was a new development in computer graphics for generation of interactive images reacting to lighting, reflections and shadows. RTX also includes artificial intelligence (AI) integration like enhancing video processing, graphics and images directly into applications. Today, parallel GPUs are making complex computations in the fields of oil exploration, machine learning, image processing, 3D reconstruction, statistics and even in stock market for stock rates determination.

In the past automatic detection of craters turned out to be a difficult task in cases when rims were not clear or overlapped or if image is noisy [Sawabe et al., 2006]. Multiple automated methods were presented by [Sawabe et al., 2006] using the data acquired by Clementine and Apollo. One of the methods was to thin down a set of edge pixels to one pixel using Hilditch's thinning algorithm. The lines were then connected depending upon the direction and length. If the resultant lines were closed with roundness more than 0.8 then lines were regarded as crater. During last decade new techniques have been developed in object detection, localization and semantic segmentation these techniques are referred to as Deep Learning. Like

mentioned earlier, these methods can now put into application because of GPUs of today, which is not only capable of performing computationally intensive tasks but also shortens the amount of processing time.

1.1 Problem Statement

Traditional methods of finding craters are manual by visual inspection of images. This approach is not practical when dealing large amount of images with craters of various sizes on either Moon or any other planet. This has resulted in a database generated by humans that are either spatially comprehensive and restricted to largest craters or size comprehensive and limited to a specific geographic region [Stepinski et al., 2012]. Manual crater counting by experts can result into disagreements as high as 40% [Greeley and Gault, 1970]. Additionally, this requires skilled man power and time to find craters of various sizes across different geographical regions.

The goal of this research is to build a framework to perform a pixelwise classification of lunar craters on optical images taken by LRO and utilizing the results of deep learning model to estimate the age of geographic location.

This approach would automate crater detection process which would be faster as compared to manual counts. The efficiency of crater detection would be tested in two ways. First, the automatic prediction will be compared against manually detected craters and secondly the age approximation would be compared to the already known Apollo 17 landing site estimation. Analysis of craters will provide diameter and CSFD required to estimate geographic age.

1.2 Research Approach

The model to perform pixelwise segmentation is based on Convolutional Neural Network. In this model images will be subjected to convolutional operation (mathematical element wise-multiplication) and such type of neural networks are known as ConvNets or simply CNNs. These networks belong to category of supervised machine learning that have proved their success by winning Large Scale Visual Recognition Challenge (ILSVRC-2010) competition which was composed of 1.2 million high-resolution images. [Krizhevsky et al., 2012a] achieved top-1 and top-5 error rates of 37.5% and 17.0% respectively.

An algorithm based on CNN will be trained and tested for lunar crater detection. The inputs are images taken by Lunar Reconnaissance Orbiter irrespective of specific camera limitations. The outcome of this model will be a trained version of this model also called as a checkpoint with minimum loss function. This trained model will be tested on lunar images and it will give predictions of craters in those images (pixel wise detection of crater in an image). This will provide the frequency of craters. As mentioned before, determination of diameters of craters would require post-processing, afterwards both of these values (crater frequency and diameter) will serve as an input to a lunar production function. This production function will be plotted to find out the age of target surface.

2 Background

But why is it important to study craters? The fact that studying the surface of other planets is a source of learning about our own planet. How crater impacts can effect the life on Earth, climatic change globally and possible consequences of what can happen to our planet in an extreme case. Finding age from crater-counts is a commonly accepted method. The results of crater-counts achieved by a deep learning model will be utilized to find out the lunar surface age. As mentioned before, crater-counts provide a less expensive way to find out the age of surface as compared to Radioactive Age-Dating. As for the second, a rock sample is needed and age can only be determined for a specific area whereas by crater-counts it is possible to find out the age of a larger area.

The problem of detecting objects could be complicated specially when object background is complex. With advancements in computer technology it was no longer optimum to rely on manual detection of objects. With big data and challenge of solving tasks in limited time, it was required to develop a mechanism that could solve such tasks in faster time compared to expensive manual operations and meanwhile limit man power. This gave rise to research in deep learning approaches to solve this problem.

CNN models composed of multiple processing layers learn the representation of data with numerous levels of abstraction by deep learning. These methods have made dramatic improvement in state of the art visual object recognition, speech recognition, object detection and many other areas such as genomics and drug discovery. Deep learning uses backpropagation algorithm to indicate how a machine should change its internal parameters of each layer from representation in the previous layer. This allows learning of machine to correctly represent i.e. patterns in an input.

Deep learning methods are the representations of multiple levels obtained by composing non linear modules such that each module transform the representation from one level to a higher level making it more abstract. Composition of such transformations makes it possible to learn complex functions. For classification, it is performed in different layers. The image is in the form of an array of pixel values, typically the first layer represents presence or absence of edges on specific locations in image. The second layer detects a shape by visualizing edges. The third layer may densify the shape into more prominent features and later layers make use of these computations to detect objects as a resultant of combination of shapes.

Object detection and classification can be performed in supervised machine learning. In a typical supervised machine learning method a system is built that can classify different objects (belonging to several classes such as human, animals, trees, cars and bikes etc.) then first it is required to have a labeled data set of such objects. For example to categorize horses and people, it is needed to have a data set which will consist of labeled images containing horses and people. This dataset would be required to train the machine about learning that how horses and people looks like so that it can classify the categories. The algorithm then outputs a vector of scores for each category. To get the category rightly identified, it is required to have the highest score for desired category out of all categories. This is not likely to happen without training of the algorithm. It is needed to compute a loss function that calculates the distance or error between the output scores and desired pattern of scores. This function helps algorithm to adjust its internal parameters that could be changed to reduce this error. These parameters are commonly referred to as weights and are real numbers.

2.1 Bounding Box Proposal

In object detection, a bounding box (also referred to as region of interest or box proposal) shows the existence of object. It is a rectangular region of the input image containing the object. Bounding boxes can be generated by some heuristic search methods such as finding region proposal by objectness, region proposal network (RPN) or by selective search method. A bounding box can be either represented by storing its two corner coordinates (x_0, y_0, x_1, y_1) or most commonly by storing its center location along with width and height such (x, y, w, h). A bounding box is generated on the basis of confidence score that how likely an object exists inside the box. The difference between two bounding boxes is usually measured as the L2 distance of their vector representations. Another simplest way is to compare the images by taking pixelwise difference and summing up all the differences. To perform such procedure, given two images can be represented as vectors I_1, I_2 then L1 can be computed as:

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

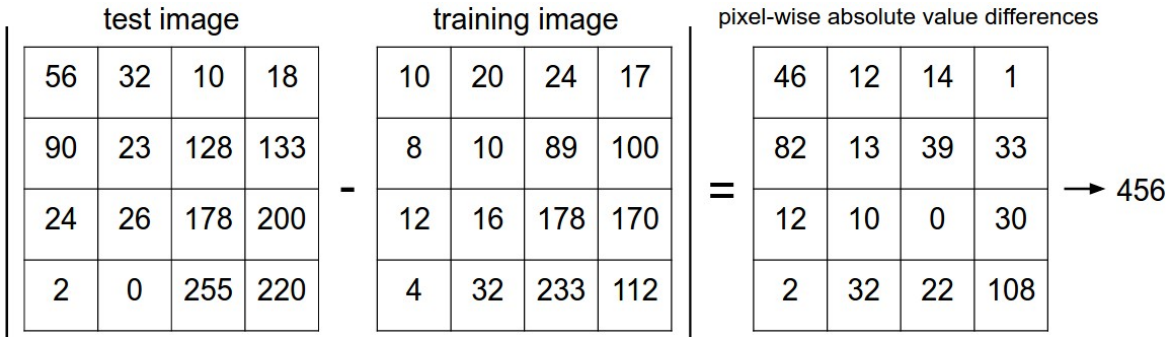


Figure 1: Example of two images which are represented as vectors and pixelwise difference is calculated to compare both images with L1 distance. This example shows one color channel. All the pixel-wise differences are added to denote a single digit value. If this value is close to zero then it indicates that images are identical. A large value shows that both images are very different.

The difference between bounding boxes can also be measured by the L2 distance. Similar like L1, images are represented in a vector form. w and h can be log-transformed before calculating distance. L2 has the geometric representation of computing euclidean distance between two vectors as:

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

In simple words this operation also computes the pixelwise difference like L1 but here all the differences are squared then added and finally equation is subjected to square root.

Difference in measurement between both metrics differs in a way that L2 prefers many medium disagreements to one big one when it comes to difference between vectors.

In case of multiple bounding boxes a common algorithm to merge them is non maximum suppression (NMS). A bounding box that overlaps another one having higher confidence score with a factor such as its intersection over union (IoU) is greater than IoU threshold, is removed. Intersection over union provides similarity between two bounding boxes. *Area of Overlap/Area of Union* = IoU. The bounding boxes play an important role in setting up dataset for the algorithm. Learning of the algorithm largely depends on correctly placed bounding box on the object.

With sliding windows, a bounding box can be attained but it may not properly fit the object. With bounding box and stride size, it might be only able to cover a part of the object and not the complete object. This problem can be solved using an approach developed by [Redmon et al.,] where a picture is divided into multiple grids and an image classification and localization algorithm is applied to each grid. Every grid has a label y which represents some parameters as shown below:

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \end{bmatrix}$$

where p_c is whether or not there is an image, b_x, b_y, b_h and b_w is to specify bounding box, c_1 is object class (i.e, craters). If there is no crater then p_c will be zero and hence the entire vector is to be dropped. When $p_c = 1$, it represents there is a crater and all the bounding box parameters would specify position of this box and hence the value of c_1 would be 1. Each grid cell will have this output vector y. The idea is to feed an input image and run forward pass (convolution, max pooling and relu) to get this output vector y.

The evaluation of object detection algorithm can be performed using IoU. As mentioned before, it gives the ratio between area of overlap and area of union. So if the algorithm outputs a bounding box which does not properly fit the ground truth bounding box representing the object then by convention the answer is taken correct if $IoU \geq 0.5$. If predicted and ground truth bounding boxes overlaps perfectly then IoU would be 1. The higher the IoU, the more accurate the bounding box is.

2.2 Linear Classification

It is one of the simplest technique to classify an object. Eventually it is extended to the entire Convolutional Neural Network. It has two major components. One is score function which does the mapping of raw data to the class scores and other one is a loss function that defines how different predicted score is from the ground truth labels. Typically loss function is higher initially and thus required to be minimized with respect to the score function parameters. This minimization is also referred to as optimization.

To define a score function which is the first component of linear classification, it is required to have information about number of images $\boldsymbol{N}$ with dimension $\boldsymbol{D}$ i.e. the size of image (pixels and channels) and distinct categories $\boldsymbol{K}$ which are the number of classes. A simplest possible function for linear mapping:

$$f(x_i, W, b) = Wx_i + b$$

Here x_i is the image which is flattened out to a single column vector of shape $[D \times 1]$. $\boldsymbol{W}$ is of size $[K \times D]$ and is often referred to as weights, whereas b is called bias vector with size $[K \times 1]$. Multiplication of matrices Wx_i gives separate classifiers in parallel where each classifier represents a row of weight matrix. The goal is to set the weights and bias parameters in a way that the output score match the ground truth labels of training data set as such that the correct classes has higher score than the incorrect classes.

A linear classifier computes the score of a class by taking weighted sum of all pixel values in all the channels of an image. If there is an image of ship then most likely surrounding pixels of ship would be blue (because of water presence). That means across blue channel weights will be largely positive whereas mostly negative in other channels i.e. red and green. Positive weights in blue channel increases score of ship class.

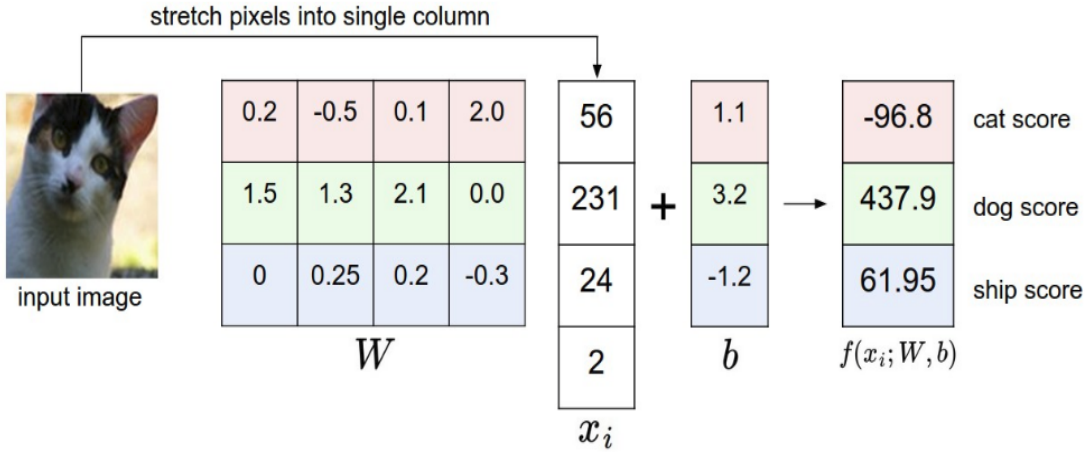


Figure 2: The graphical representation of eq ($f(x_i, W, b) = Wx_i + b$). It shows that an image x_i is represented as a one dimensional vector which is multiplied by set weights and resultant is added with a bias leading to the highest score for detection of cat image. The weights are set badly as classifier thinks it is a dog instead of a cat.

2.3 Normalization

In machine learning normalization is a typical practice in preprocessing of data before its final preparation to run it through a neural network. Data can be standardized meaning subtracting a mean of dataset from each data point and dividing the dataset by standard deviation, mathematically standardization z written as:

$$z = \frac{x - m}{s}$$

Another way of normalization is to normalize a dataset by bringing it in a range of 0 to 1. This is typically true for image processing when pixels ranging from 0 to 255 which are normalized from 0 to 1. It is necessary to normalize data because without it some numerical data points can be very high and other might be very low. The larger data points in non-normalized datasets can cause instability in neural networks because the relatively large inputs can cascade down through the layers in the network which may cause imbalance gradients which may therefore cause the exploding gradient problem. This makes a network drastically harder to train and additionally significantly decrease training speed.

In neural network exploding gradient refers to a problem when weight is larger than identity and it is multiplied by pixels of image (ignoring bias in this case) then resultant is very high, this will further pass on to next neuron and it will be even higher, thus in a network specially in a deep network it will result into a very large value. The opposite problem is a vanishing gradient when a weight is less than identity then multiplication of weight results into even a smaller number therefore output is a very small value which means that network has barely learned in all of the layers.

2.4 Backpropagation

The CNN requires to update its weight for a given training data in order to reduce loss. Backpropagation is an efficient method for computation of gradients which are required for gradient based optimization of weights or kernel parameters in neural networks [Rumelhart et al., 1988]. This optimization problem refers to minimize the loss function which is performed by the specific combination of weights. Backpropagation requires the computation of loss function at each iteration therefore loss function should be differentiable and continuous.

For a multiplication function of two numbers i.e. $f(x, y) = xy$, it is possible to derive the partial derivative for either of input:

$$f(x, y) = xy \quad \rightarrow \quad \frac{\partial f}{\partial x} = y \quad \frac{\partial f}{\partial y} = x \tag{1}$$

The derivation function of a variable indicate the rate of change of a function with respect to that variable surrounding an infinitely small region near a specific point.

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{2}$$

In above equation the operator $\frac{d}{dx}$ (a derivative operator) is applied to a function f. The resultant is a derivative. It can be interpreted as; if h is very small then s straight line determines approximation of function and slope of the line is derivative. The sensitivity of an expression on a value is determined by the derivative. If $x = 4, y = -3$ then $f(x, y) = -12$ and this would return a derivative on $x\frac{\partial f}{\partial x} = -3$. It is clear that if value of this variable is increased by a tiny amount, it would create an impact of three times decrement because of its negative sign. It also be seen by rearranging the above equation such that; $f(x + h) = f(x) + h\frac{df(x)}{dx}\Big)$. In other way as $\frac{\partial f}{\partial y} = 4$, it is also expected that by increasing the value of y by a tiny amount which is h would also increase the function output by $4h$ because of positive sign.

The derivative of function $f(x, y)$ is a vector of partial derivatives which is $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] = [y, x]$. Gradients can also be calculated for addition operation such as:

$$f(x, y) = x + y \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1 \quad \frac{\partial f}{\partial y} = 1 \tag{3}$$

This indicates that derivative of both x, y is one regardless of values of x, y. Increasing either x or y will increase the value of output to function f independent of actual values of x, y unlike the above case of multiplication. Another operation is a max operation:

$$f(x, y) = \max(x, y) \quad \rightarrow \quad \frac{\partial f}{\partial x} = 1(x >= y) \quad \frac{\partial f}{\partial y} = 1(y >= x) \tag{4}$$

That is the gradient is 1 on the input which is larger and 0 on the other input. If $x = 4, y = 2$, then max is x and the function is not sensitive to value of y. If the value of y is increased by a tiny amount h, the function will keep outputting value of x which is 4, hence the gradient is 0. If the value of y is changed by a large amount (in this case larger than 2) then output of f will change. Derivatives shows nothing about the large changes on the function inputs. They only inform about the infinitely tiny amount of changes on the inputs as indicated in eq. 3.

Backpropagation relies on the above explained rule in a more complex neural network. It computes the gradient of loss function with respect to each weight via chain rule based on the above mentioned examples. The computation is performed one layer at a time starting from last layer till the second layer of network. First layer is not included as it is the input layer. Also redundant calculations are avoided by computing derivative of each layer at a time. In other words, the weights are adjusted every time in such a way that loss is minimized and the model reaches at a checkpoint where adjusted weights are very close to target features, hence to get most accurate predictions.

2.5 Weights Initialization

Initially weights of a CNN before training are randomly chosen. The key rule to pick a random weight is not to initialize with a too large or too small value to avoid the well known problem of vanishing or exploding gradients. Some initialization methods like He Initialization and Xavier Initialization are the common practices to setup weights. Before training, the CNN is not able to make meaningful predictions because there is no relation between the input images and their annotated outputs with classes. This process is performed in training where weights are adjusted in a way so that difference between desired output and network output is minimized. In other words network is trained to look for the right features needed for classification.

2.6 Neural Network Overview

In supervised machine learning object detection task can be achieved using different approaches. There are various architectures based upon statistical functions. A neural network

is made up of neurons having learnable weights and biases. It means that during training process, weights and biases can be updated. A neuron in an artificial neural network is referred to a mathematical function. It receives inputs which can also be output of neurons from a previous layer, weighs each input and sum them up. A weight shows the strength of connection of one neuron to another neuron in next layer. Every neuron has a bias and it determines if the neuron is activated or not. Biases are added to the product of weight and value of neuron. Each neuron is fully connected to all the neurons of previous layer. These neurons do not share any connection within a single layer. The last fully connected layer is referred to as "output layer". The network takes an input and computes the output by taking a dot product. The entire network takes raw image pixels and presents an output in the form of a digit representing the class score for an object. Output layer represents the class scores. It expresses a single differentiable score function. There could be few to many hidden layers and a fully connected layer that has a loss function i.e support vector machine (SVM) or softmax. A typical neural network architecture is shown below:

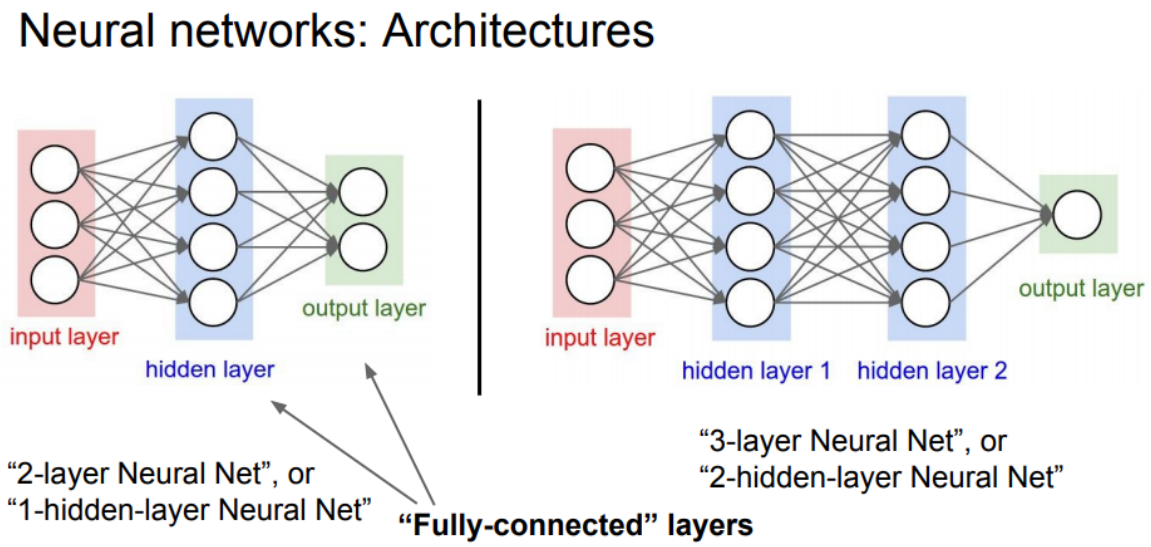


Figure 3: A typical Neural Network architecture

Artificial Neural Networks are very similar to convNets but they do not convolute on an image through a filter thus leading to many more parameters as compared to CNN i.e. in CIFAR-10 dataset, images are of 32x32x3 (width, height, 3 color channels). That means in an input image of size 32x32x3, an Artificial Neural Network would have 3072 weights. These weights tend to increase with size of image.

2.7 Convolutional Neural Network Overview

Just like Artificial Neural Networks as shown in Fig 3, convNets are also made up of neurons consisting of weights and biases which are learnable. Each neuron receives an input then performs a dot product and optionally follows it with a non-linearity. The entire network represents a single differentiable score from the raw image input to the output score. Last fully connected layer has a loss function which represents the error score. This entire process

is referred to as forward pass. In convNet, the architecture assume inputs as images which provides flexibility of encoding certain properties. This greatly reduce the amount of parameters to learn and make forward pass more efficient. The layers of convNets have neurons arranged in 3 dimensions (width, height, depth). The neurons in a layer are not connected to all of the neurons in a previous layer like in an Artificial Neural Networks, instead they are connected to only small region of previous layer. The output results into a single vector of class scores by reducing the full image. Following are the types of layers in a Convolutional Neural Network:

2.7.1 Convolutional layer

Lets suppose there is a 2D input image of size 6x6. A filter convolves through the image to extract an output image with desired features. The filter (which is also a matrix) could be of size lets say 3x3, extracts certain features of the image. This process is known as convolutional operation because filter convolve through the image. In example, it would look as shown in fig: 4

INPUT IMAGE

18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

FILTER

1	0	1
0	1	0
1	0	1

OUTPUT IMAGE

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

Figure 4: Convolutional Operation

In fig: 4 the output 429 in 4x4 matrix is obtained by addition of element wise multiplication of the filter with top left 3x3 portion of the input image. Then filter jumps to next pixel and other values are obtained (stride is taken as 1). Output size can be calculated using eq.

$$\frac{n+2p-f}{s}+1 \times \frac{n+2p-f}{s}+1 \tag{5}$$

In the above example n equals 6 as image is 6x6 and f is 3 because of 3x3 filter size, p is padding which equals 0 and s stands for stride which is 1, inserting values in this equation would give us the size of output which is a 4x4 matrix.

2.7.2 Pooling layer

An input image could be large which increases the amount of parameters and introducing a pooling layer helps to reduce the number of parameters. Most commonly used type of pooling is max pooling. Fig 5 shows an example of max pooling with stride and filter size of 2. The

idea is to keep the high values in each quadrant because the highest number represents a particular feature and in the example shown in Fig 5, number 6 is the highest value in this quadrant. It means that the most activated pixel in this quadrant is 6 and same goes for other quadrants. The high values are preserved and lower ones are dropped out which are not as activated. Another pooling layer type is average pooling where averaged output of all the pixels is preserved. As it can be seen in the example below that pooling has reduced a 4x4 matrix to just 2x2 matrix, significant amount of parameters are reduced, in addition pooling may also help in reducing overfitting. The resultant matrix after a pooling operation can be obtained from the eq.(1)

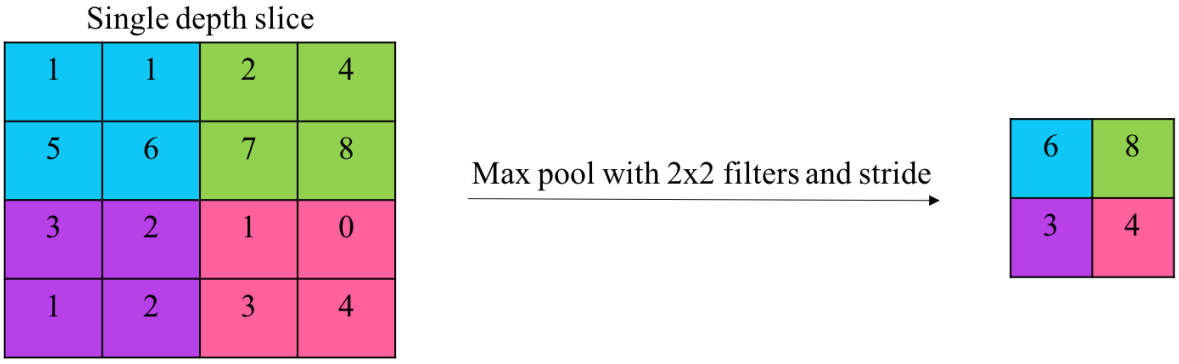


Figure 5: Pooling Operation

2.7.3 Fully Connected layer

After multiple convolution and pooling operations, finally an output can be generated in the form of a class. Convolution and pooling layers only extract the features and reduce amount of parameters from the input images. Fully connected layer (FC layer) is applied at the end to generate the required output equal to the number of classes. There can be multiple FC layers to further minimize the amount of parameters. In convolution the resultant is generation of 3D activation maps whereas the intention is to know whether the image belongs to a particular class or not. The output layer (which generates the class scores) has a loss function and once the forward pass is completed, backpropagation begins to update biases and weights for loss and error reduction. An overview of the architecture is shown below:

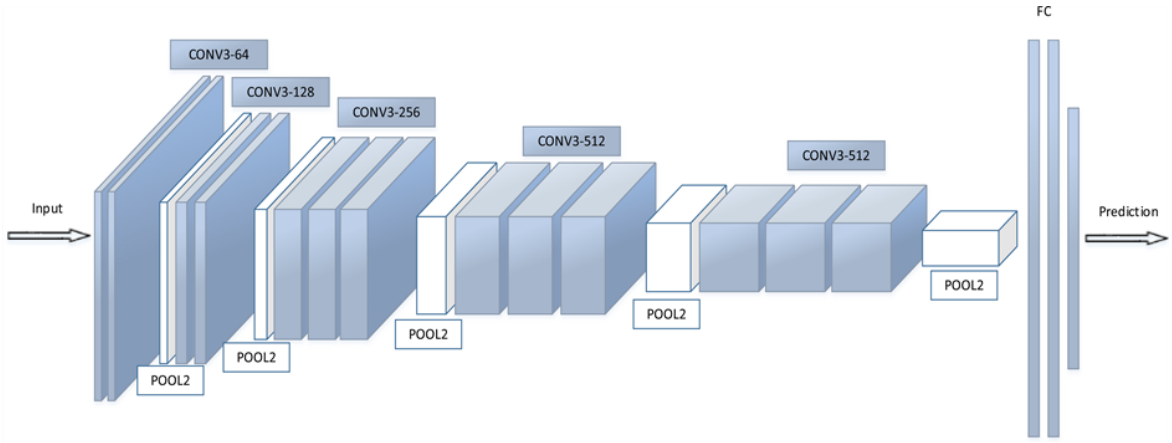


Figure 6: Convolutional Neural Network Architecture [El Khiyari and Wechsler, 2016]

2.8 Loss Function

It determines the amount of deviation from the groundtruth or labeled data. Higher loss means the actual outcome is very different than expected result. In Fig 2 instead of high cat score, dog score was the highest. This anomaly is measured by loss function. A high loss function indicates poor classification of training data and low means better classification. There are different ways to define a loss function. Finding craters is a classification problem (i.e crater or no crater). So the loss is measured in as a classification loss problem.

2.8.1 Cross Entropy Loss

It is also known as the log loss. It measures the performance of a classification model whose output is a probability value between zero and one. The cross entropy loss increases as the predicted probability diverges from the actual label so predicting a probability of i.e. 0.019 when the actual observation label is one would be bad and it would result in high loss value. Therefore, probability of observation as zero or one (positive or negative) is written as:

$$H(p, q) = -\sum_{x} p(x)\ \log q(x) \tag{6}$$

In the above equation p and q are the cross entropy of true distribution and predicted distribution. The calculation of above equation depends on following factors:

1. Type of layers used in neural network.
2. Type of activation function used. Many activations might not be compatible with calculation because of output value which is either greater than one, negative value or do not sum to one. Therefore, softmax function is often used for multi class classification as it guarantees a well behaved probability distribution function.

A machine learning, a common convention is to represent the ground truth (or labeled) data by vector $\mathbf{y}$ and $\hat{\mathbf{y}}$ is a vector containing the estimate. For a single example the equation is:

$$L = -\mathbf{y} \cdot \log(\hat{\mathbf{y}}) \tag{7}$$

If for example a true label is $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix}$ and predictions are $\begin{bmatrix} 0.1 & 0.5 & 0.1 & 0.1 & 0.2 \end{bmatrix}$ then in this specific example all the probability is given to the first value and remaining are zero so they can be ignored. Mathematically it will be calculated as:

$$L = -(1 \times \log(0.1) + 0 \times \log(0.5) + \ldots)$$

$$L = -\log(0.1) \approx 2.303$$

It is clear that loss would be same even if predictions are $\begin{bmatrix} 0.1 & 0.5 & 0.1 & 0.1 & 0.2 \end{bmatrix}$ or $\begin{bmatrix} 0.1 & 0.6 & 0.1 & 0.1 & 0.1 \end{bmatrix}$. This is a key feature of multi class cross entropy loss. The value does not depend on how probability is split between incorrect classes, it only penalizes the probability of correct class.

2.9 Activation Functions

These functions are an extremely important feature of a neural network. These functions decide which neuron (a neuron is noting but a mathematical function) would be activated. This means whether the information received by a neuron is relevant or should it be ignored. An activation function performs a non-linear transformation on the input signal and forward it as an input to the next layer of neurons.

Without activation functions weights and bias would be simply doing a linear transformation and a linear equation is easy to solve but very limited to the capacity of solving complex problems. Therefore, a neural network without having an activation function is nothing but a linear regression model which will not be capable of learning or performing complex tasks. Image classification or object detection is a complicated task and would require non-linear transformations. These functions make the process of back propagation possible because of the receiving gradients and error which are a measure to update weights and biases.

Most commonly used activation functions are discussed below:

2.9.1 Sigmoid Function

It is defines as:

$$a = \sigma(z) = \frac{1}{1 + e^{-z}} = \frac{e^z}{e^z + 1} \tag{8}$$

where

$$z = Wx_i + b \tag{9}$$

$W being weight vector and x_i is image vector, b stands for bias$

The maximum output of this function is 1 and minimum is 0. Output always lies between values 0 and 1. Plotting a graph of sigmoid function represents it output more clearly:

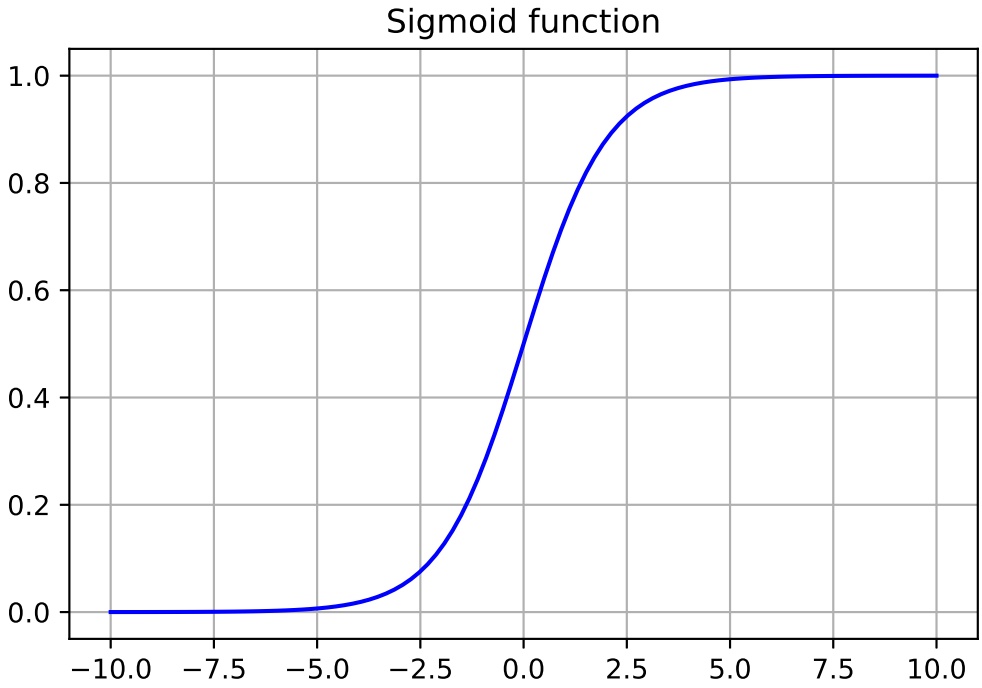


Figure 7: Sigmoid Function

From the graph it can be seen in the graph that z is 0 when curve is passing though 0.5. In convention a rule can be set if i.e. z is greater than 0.5 then output is always 1 and if less than 0.5 then it is 0. A notable behavior of this function is that if z is larger than the dotted region in graph then the derivative of this function is close to zero and same way if it is negative slope is again equal to or close to zero.

This means that it can slow down gradient descent. It also becomes a source of vanishing gradient problem. If the weights are initialized with either very large or very small values then these values saturate the input to sigmoid at very valued region (close to zero or close to one). Even if weights are initialized at a value i.e. 0.2, in a deep network with many layers it will also lead to vanishing gradient problem. In case of only 4 layer network $0.2^4 = 0.0016$ which is small and will get even smaller in next layers. Also the mean of data is 0.5 which meas that data is not centered for the next layer.

It is used for a problem where binary classification is required. In a neural network layer sigmoid must not be used in every layer because of the problems mentioned above. Anyhow, where binary classification is required, it can be useful in the last layer of the network to squash output such as $1 \geq \hat{y} \geq 0$.

2.9.2 ReLU Function

It stands for rectified linear unit and defined as:

$$a = \max(0, z) \tag{10}$$

The graphical form is given as:

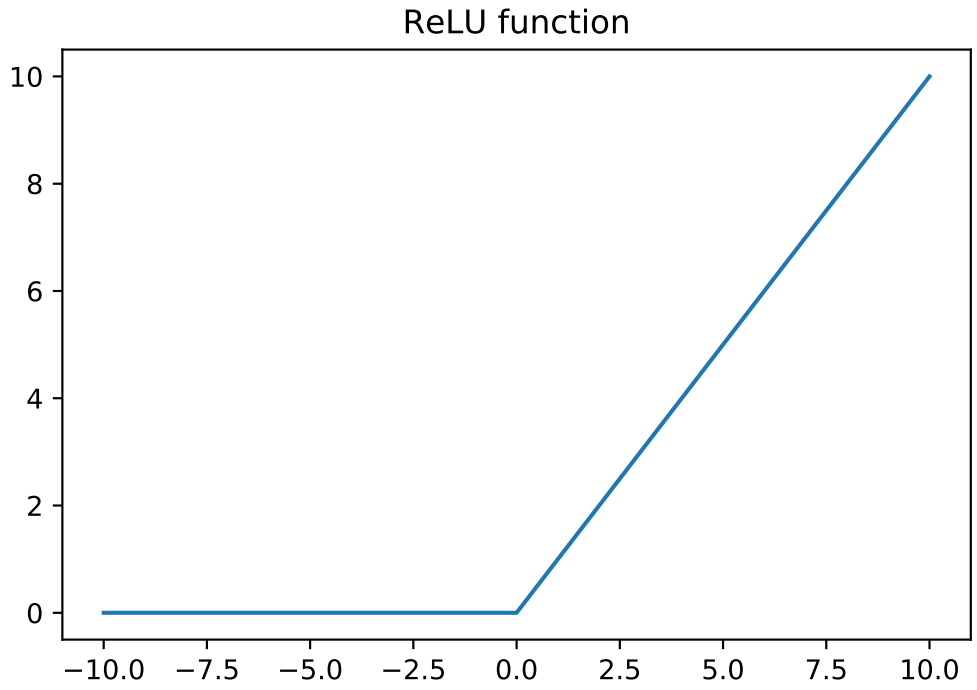


Figure 8: ReLU Function

As seen in the graph ReLU gives an output for a positive value and otherwise the output is 0. It is to be noted that in graph the line looks linear but ReLU is non-linear in nature and it is one of most popular functions used in neural networks because of its simplicity and ability to not let all the neurons fire at once. ReLU is computationally much faster than sigmoid or tanh (written as $a = \tanh(z)$), it does not have exponent computation in such as sigmoid activation function therefore reduces training time significantly in very deep neural networks. [Krizhevsky et al., 2012a] observed that training deep CNNs with ReLU is much faster as compared to sigmoid or tanh.

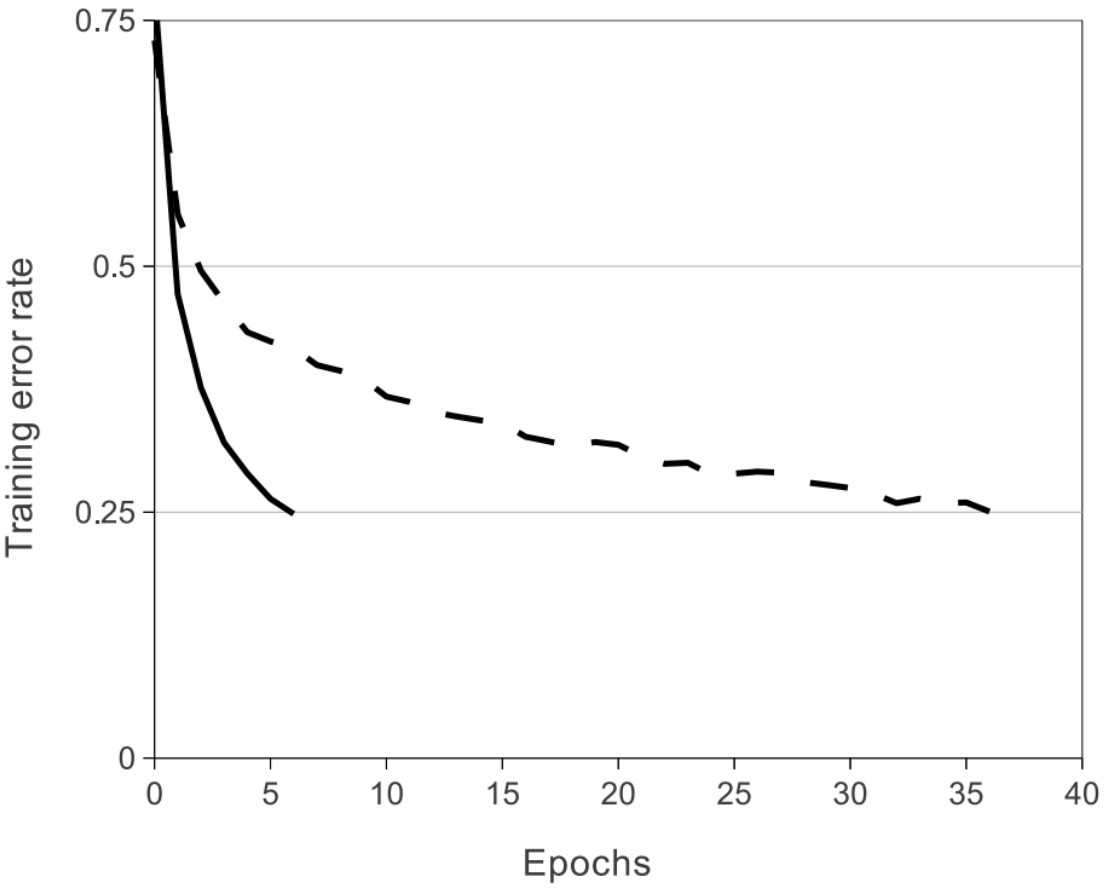


Figure 9: A CNN of four layers trained with ReLU in solid line and tanh in dashed line. It shows that ReLU reaches a 25% training rate on CIFAR-10 six times faster as compared to tanh.

Unlike sigmoid when the receiving input is at the right or left plateau i.e. less than -5 or greater than 5 in Fig 7 which makes it meaningless to pass a backward pass because of derivative being closer to 0, ReLU only saturate when the input is a negative value. Also in Fig 9 [Krizhevsky et al., 2012b] it can be seen that ReLU allows training of larger nets at much less computational costs which means more parameters can be trained at the same computational cost.

2.9.3 Softmax Classifier

A binary classifier is a function that determines the class of an object i.e. if there is a crater (which has an assigned class of 1), ideally it will recognize it as one. Softmax classifier is generalization of binary form of logistic regression. It interprets the scores as unnormalized log probabilities for each class and has the following form:

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \tag{11}$$

Loss function has to minimize the negative log likelihood of correct class. Here f is the mapping function (x_i, W) and f_j represents the j-th element of the class scores vector f. In order to represent softmax function for the entire dataset, one would need to take the mean of L_i over all training data and hence softmax function is represented as:

$$f_j(z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{12}$$

z is the vector of score of arbitrary real values and those are squashed to a vector of values ranging between zero and one. The sum of these values is one. The form $\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$ in eq.11 is the estimated distribution. Plugging it into eq.6 makes it clear that softmax is minizing the loss.

2.10 Lunar Production Function

Moon provides an ideal test site for studying crater records, particularly since almost all of the lunar endogenic activities ended more than around 3 Giga years (G.y.) with some exceptions [Hiesinger et al., 2000]. Therefore, in last 3 G.y. crater impacts have dominated to change lunar landscape. Space missions have also studied the Moon extensively and collected samples from Moon have provided a unique opportunity to assign age to craters and areas where accumulated crater are counted [Stöffler and Ryder, 2001]. Therefore, on the Moon it an be estimated that cratering rate is the number of craters of a given diameter that are accumulated at given surface during a given time interval.

Crater population is changed by obliteration processes. If it is assumed that a planetary surface is obliterated by some process and crater population starts to develop then crater SFD represents the production SFD of the projectiles. Many authors have tabulated and generalized large amount of crater-counts data in an attempt to understand this production function. Some of the most commonly known lunar crater SFD are proposed by W. Hartmann and G. Neukum.

2.10.1 Hartmann Production Function (HPF)

Hartmann uses a log-incremental SFD representation with a standard bin size for diameter to represent the crater SFD of terrestrial planets. The obtained results are referred as Hartmann production function or HPF. The number of craters per kilometers squared are calculated for a certain diameter range which is $D_L < D < D_R$, this range represents a bin where D_L and D_R are the left and right boundaries of diameter range respectively. The standard bin width is $D_R/D_L = 2^{1/2}$. HPF provides a resultant tabulated data for one specific moment of time, this is the average time of lunar mare surface formation. Findings from most of the lunar mare basalt samples suggests a narrow range of ages between 3.2 to 3.5 G.y. [Stöffler and Ryder, 2001] therefore, the condition of having a fresh surface is satisfied. Some lava flows on the surface could be younger [Hiesinger et al., 2000] therefore the age variation is represented by a factor of 1.1.

The tabulated HPF is considered reliable for the projectile of a production function because the crater counts from different areas of the Moon are combined and averaged. The incremental form of HPF takes form of a piece-wise three segment power law [Ivanov, 2002].

$$\log N_{2^{1/2}} = -2.616 - 3.82 \log D_L, (D < 1.41km) \tag{13}$$

$$\log N_{2^{1/2}} = -2.920 - 1.80 \log D_L, (1.41km < D < 64km) \tag{14}$$

$$\log N_{2^{1/2}} = -2.616 - 3.82 \log D_L, (D > 64km) \tag{15}$$

The function is represented in Fig 10. Hatmann chose power law segments in 1960s when this work started. Some of the selections were on the basis of historical reasoning that only the craters branch with diameter range between 1.41km and 64km was well established. At that time there were already existing laws of meteorites and asteroids and Hartmann's attempt was to relate those laws to lunar data.

2.10.2 Neukum Production Function (NPF)

Neukum proposed an analytical function describing the cumulative SFD of lunar impact craters. He wrote a series of publications in description of his function. For summaries, see [Neukum and Ivanov, 1994] and [Neukum, 1983]. Neukum showed that the production function is stable since 4 G.y. The time Neukum proposed this function, a full size crater spectrum was known. His approach was different from Hartmann in a way that he computed a polynomial fit to the cumulative number of craters, N per squared kilometers with diameters greater than the provided values of D. Where as Hartmann proposed a piece-wise exponential equations for his production function. For the time period of 1 G.y., Neukum's production function can be represented as

$$\log_{10}(N) = a_0 + \sum_{n=1}^{11} a_n [\log_{10}(D)]^n \tag{16}$$

In above equation D is in km, N is the number of craters with diameters greater than D per squared kilometers per Giga year and values of coefficients a_n are provided in table. The above equation is valid for crater diameters from 0.01km to 300km.

On the basis of age assumption NPF was fit to the crater count. It is notable that both HPF and NPF are a good match for the crater diameter (D) data under 1km range. However, with D > 1km, HPF is much higher than NPF and both functions meet again at diameter of approx. 40km. In Fig 10 it can be seen that the maximum variation between the two functions is a factor 3 around the diameter of approx. 6km. Note that below the diameter of 1km and between 30-100km, both of the functions are same. After 100km both started to decline but HPF declines more rapid as compared to NPF.

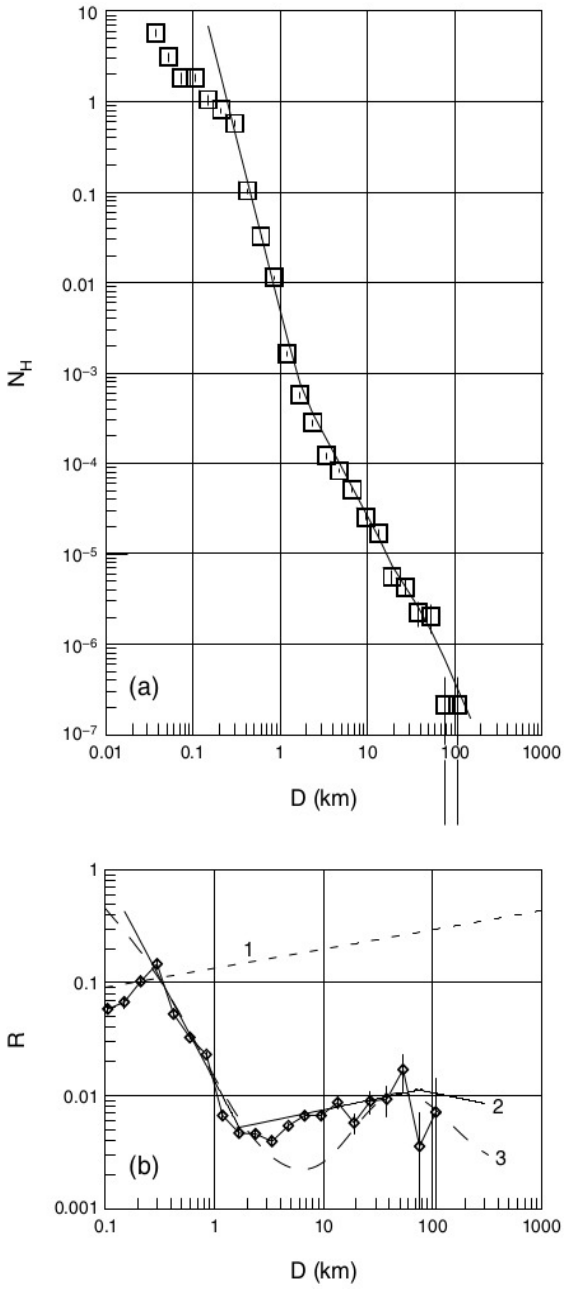


Figure 10: (a) The incremental representation of the Hartmann production function (HPF). The HPF, in a direct sense, is the set of points shown in the plot. Straight lines represent the piece-wise power law fitting to the data (equation (1)). (b) Comparison of pro- duction functions derived by Hartmann (HPF) and Neukum (NPF) in the R plot representation. The maximum discrepancy between HPF (2) and NPF (3) (roughly a factor of 3) is observed in the diameter bins around D 6 km. Below D 1 km and in the diam- eter range of 30–100 km, the HPF and NPF give the same or simi- lar results. Fitting the HPF to equation (3), we obtain a model age of 3.4 G.y. The NPF, which is fit to the wide range count of impact craters in the Orientale Basin, yields a model age of 3.7 G.y. The dashed line 1 represents the approximate saturation level estimated by Hartmann (1995).

3 Related Work

Most of the craters are formed as the result of meteoroid impacts. The relative formation age of local area of planet could be estimated when frequency of size distribution of meteoroids and its time variations are known. [Sawabe et al., 2006] proposed an algorithm that did not depend on cameras, spatial resolution or sun illumination. It also did not require to tune any parameters either. This algorithm was improvement to their own previously proposed algorithm. Their previous method did not include pyramid representation of the image which was included in later method and thus improving accuracy and reducing processing time. The algorithm was applied on images acquired by Clementine and Apollo under different solar elevation. The accuracy was more than 80% compared to the manual detection results [Sawabe et al., 2006]. Accuracy rate was validated by comparing with crater count results of [Neukum et al., 1975]. In 2009 [Martins et al., 2009] performed Viola and Jones (2004) algorithm on Mars dataset gathered by the Mars Orbiter Camera onboard Mars Global Surveyor probe. In this paper author claims that no such method existed that is satisfactory for craters detection.

There are many reasons for usage of CNN to crater detection problem. CNNs have proven record in computer vision problems and such datasets where a correlation lies between features [Long et al., 2015]. Not only in computer vision based on images but also CNNs have demonstrated their versatility in sounds and signals as well. Another major reason of using CNNs is the ability to learn from features hence they has the capability to engineer their own representation of features, thus removing human involvement in developing sophisticated pre-processing models and custom input features. CNNs are also able to classify objects in images which are in different scales even when on a single image. For lunar craters this case is similar where craters range from few meters in diameter to several hundreds of meters and have multiple instances in a single image.

A deep learning model (CNN) was introduced by [Silburt et al., 2019] for crater identification on lunar digital elevation map images also known as DEM. The author also applied transfer learning to craters on Mercury. In this paper random DEM images from LRO and Kaguya were taken as an input data. Thus it was a global gray scale map. The proposed CNN detects only half of the caters per target. The post-CNN recall is lower at $57\% \pm 20\%$ but detections for craters less than 15 pixels largely improved the post-CNN test recall to $83\% \pm 16\%$. The sample of DEM from [Silburt et al., 2019] is shown in Figure 11

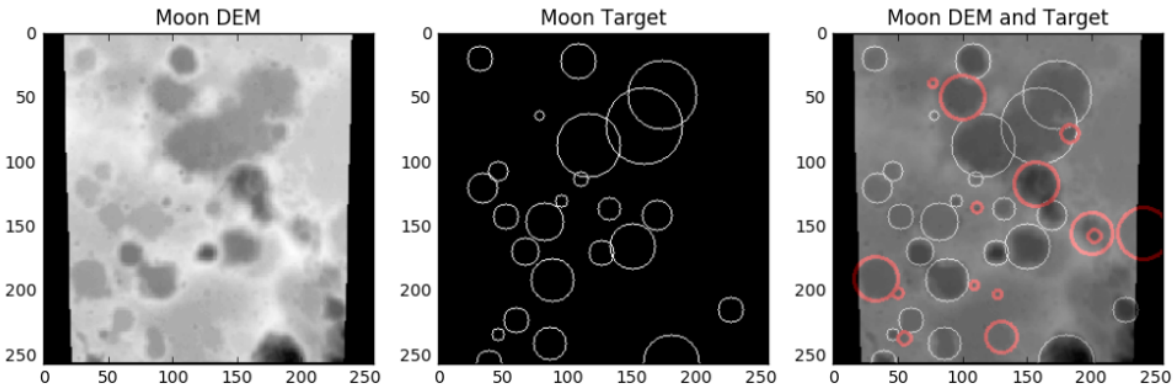


Figure 11: Most left is the Moon DEM sample and middle image shows prediction of craters and most right one shows the missing classifications which are marked in red circles

4 Model Architecture

The model architecture is composed of contracting and expansive paths as shown in the Figure 13. Like in a typical CNN, the image becomes smaller as a result of convolutional operations. U-net has the same effect and thus left part of this pipeline is referred to as contracting path. It consists of several convolutional operations such that each of two unpadded convolutional layers with an activation of rectified linear unit (ReLU) are followed by a layer of max pooling of the size 2x2. This results into downsampling of the image. During each downsampling step the number feature channels are doubled as seen in the Figure 13. This part ends up with a dropout function before it starts expanding. Dropout is nothing but regularization which reduces interdependent learning amongst neurons. This reduces the possibility of overfitting the model.

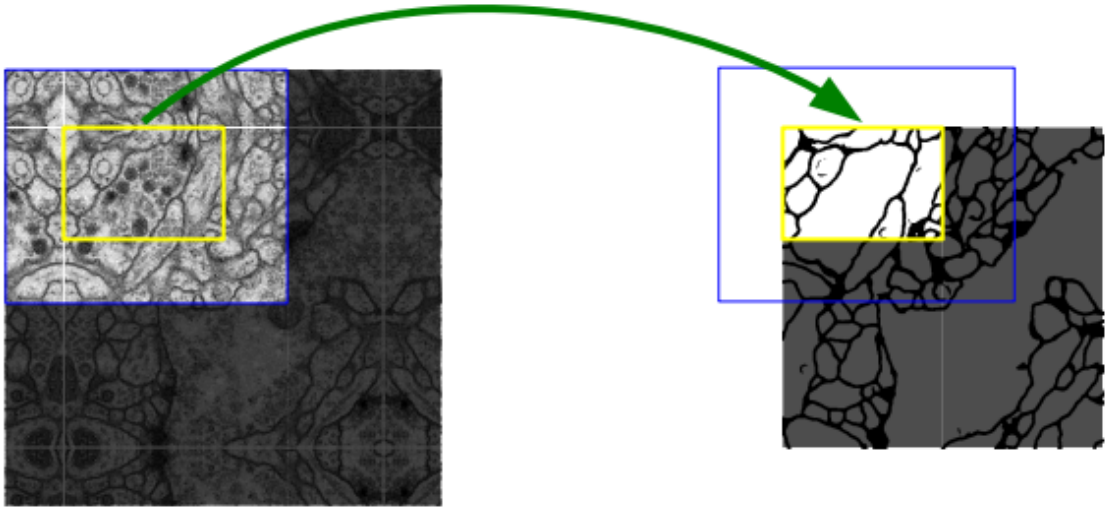


Figure 12: This is example of neuronal structures segmentation. It shows the overlap-tile strategy for seamless segmentation of large images. The yellow area is target for prediction and data inside the blue area is required as in input for prediction. Missing input data is exptrapolated by mirroring

The right part of Figure 13 illustrates the expansive path where every step consists of up-sampling of feature map which is followed by a 2x2 convolutional operation (also referred to as up-convolution). This halves the number of feature channels. The gray arrow shows the concatenation of corresponding feature map from contracting path where it is cropped from the image. It is then subjected to two 3x3 convolutions and each followed up by a ReLU activation function. The cropping is required because of loss of boarder pixels during convolutions. At the final layer a 1x1 convolution maps each 64 component feature vector to the target number of classes i.e. background class and crater class. The final layer follows up by sigmoid activation function which determines the output of a class score between 0 to 1. Finally, binary cross entropy determines the loss while training of algorithm. In this model the total number of convolutional layers amounts to 23. It is important to select the input image size so that max pooling operations are applied to a layer with the same x and y size. This allows seamless tiling of segmentation map in output as shown in Figure 12

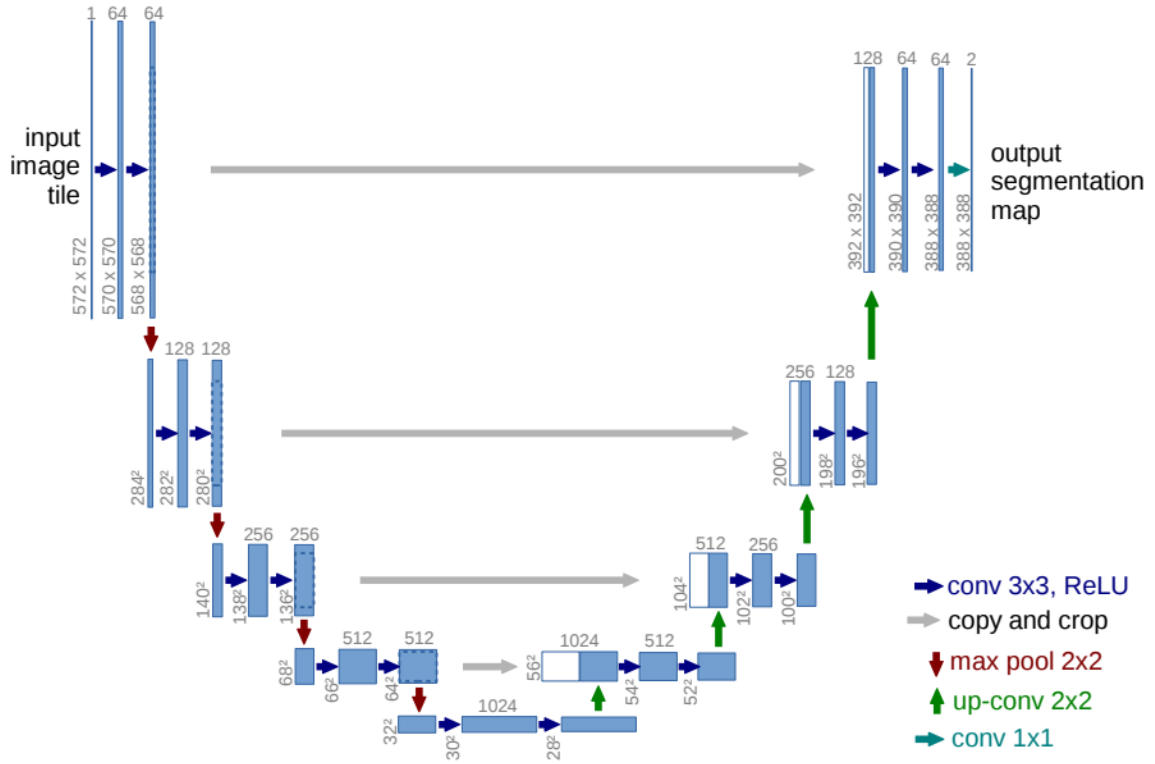


Figure 13: U-net Architecture

5 Methodology

This section is divided into following parts:

- Data Set
- Pre-processing
- Training of Network
- Postprocessing to get segmented images

5.1 Data Set

The dataset which is in the form of gray scale lunar images is taken from lunar reconnaissance orbiter camera (LROC) archive *http://wms.lroc.asu.edu/lroc/search*. Images from this data source are very large in size (5064x52224) with resolution of 1.009. Which means processing these images would require a competitive hardware. Also each image in this data has several thousands of craters with sizes ranging from very few meters to several hundred meters. Keeping in view the size of an image and amount of craters it contains, it was considered to take one of the image and crop it into several small images of equal size and width. This dimension was chosen based upon the ease for annotation process. Smaller image size would mean less craters to annotate in a single image and therefore more number of images

in training set which would make data handling such as pre-processing and post-processing simpler. The amount of cropped images are 300 each with size 350x350 and are gray scale images.

5.2 Pre-processing

In this step before training of Neural Network images are processed for annotation and extracting labeled masks for the preparation of input data set for training. It involves following step.

- Image Annotation
- Extraction of Binary Masks

5.2.1 Image Annotation

It is one of the most important tasks in computer vision problem related to deep learning. This task is done by human, images are annotated with labels. These are established by the annotator who can perform this task on the basis of simple instructions. No skilled labor is required to do this task. These labels carry information to the Neural Network about what are the target features in image. In this particular project there is only one label which is "crater" and has shape of a circle.

Annotation (labeling) could be laborious and time taking task. It is specially required when a network is not trained on the same type of class as required for predictions. In the case of lunar craters, there is not a great amount of work that has been done before in terms of machine learning and finding already annotated data set seems difficult to find. Therefore labeling was needed for training purpose. There are several tools to perform annotation, some are listed below.

- LabelMe: One of the most commonly used tools. It has been observed that it is slow on UI mainly when zooming into images.
- RectLabel: At the time of annotation there was no support for Linux, it only worked for Mac.
- LabelBox: It provides options for different labeling tasks and works well for larger annotation projects.
- VGG Image Annotator (VIA): Very simple design and easy to use for beginners as well as suitable for large projects as well.
- COCO UI: This tool for COCO dataset annotations.

After performing few annotation experiments from the above mentioned tools, VGG Image Annotator was found most simple and faster to annotate lunar crater dataset. This tool stores annotations in the form of a json format. Once the labeling is complete, a large json file represents the craters with a center point as x and y coordinates and radius respectively.

In Figure 14, images in the first row are part of the training dataset and below them are the annotations. Images marked with circles on craters are not intended to be used for the

training of algorithm, those are only visualizations to labeled data so that wrong annotations could be removed or missing ones could be added.

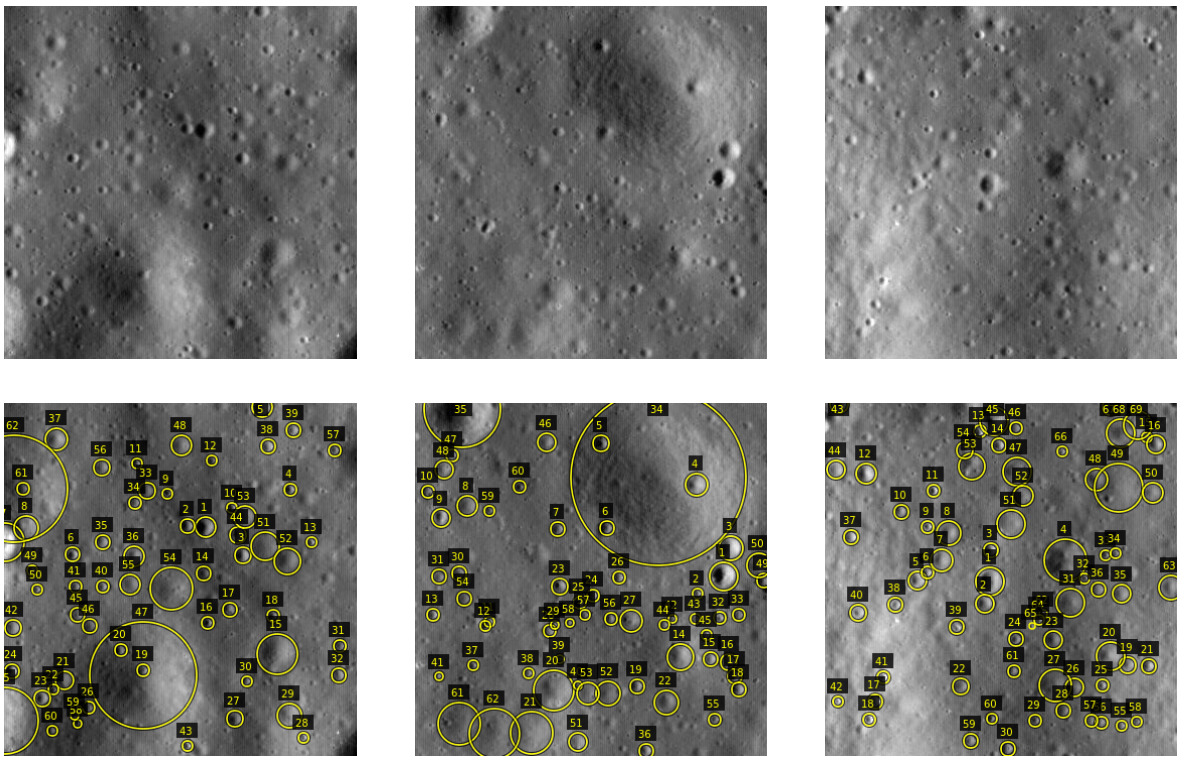


Figure 14: Example of cropped images and corresponding labeled images. Annotation is performed using VIA tool

5.2.2 Extraction of Binary Masks

Like mentioned before the result of these annotations (json) is projected to get the binary masks of craters. These masks are binary images that represents the black and white pixel values of images. White pixels means craters and black means background class as shown in the Figure 15. White circles are craters belonging to Figure 14. These provide learning objective to Neural Network that how the shape looks like and what features to learn. These binary masks have pixel values either 0 or 255 and are of the same shape along x and y as of the images. These masks are normalized in side the algorithm architecture to values 0 and 1.

5.2.3 Data Augmentation

To create larger dataset without additional annotations, data augmentation is necessary. Creation of a larger dataset is done to achieve better training for the algorithm. Augmentation creates multiple images out of one image which means larger dataset for training of algorithm and thus achieve better performance on test data. Therefore every training image is randomly augmented in following ways:

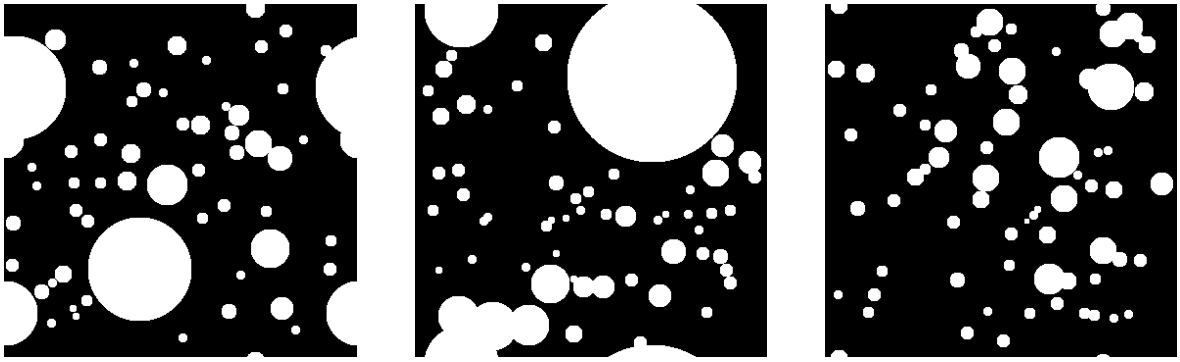


Figure 15: Visualization of binary masks after projection from annotated (json format) images that belong to Figure 13

1. Rotation: Random rotation of images from 0° to 360° angle.
2. Translation: Random change in shift in x and y direction of image between -4 to 4 pixels.
3. Flipping: Image is flipped with a factor of 0.5 as probability.

The images are randomly perturbed during training so that augmentation can have maximum effect. This way model have a low probability of seeing same type of training image more than once. In lunar crater dataset there are no RGB images hence color augmentation is opted out.

5.2.4 Contrast Limited Adaptive Histogram Equalization (CLAHE)

Histogram equilization considers the global contrast of an image. Therefore in cases where image has high pixel values, such regions tend to loose alot of information as those pixels are stretched further towards 255 and object becomes too bright.

By using CLAHE, the image is divided into small blocks also called tiles (function implemented in OpenCV has default tile size of 8x8). Then each one of these tiles histogram equalized. This makes histogram confined to a small area. It will be amplified if noise is there. Contrast limiting is applied to avoid this problem. The default contrast limit is 40 in OpenCV and pixels with larger values are clipped and distributed to other bins uniformly before application of histogram equilization. After that there are artifacts in tile boarders which are removed by the application of bilinear interpolation. By applying CLAHE the detection of craters increased up to 50%.

5.3 Training

The data composed of images and their respective masks is divided into training (80%) and validation (20%) data set. This percentage practice is typical in machine learning, however it does not have to be always at this rate. In case of large data set i.e. several thousands of images, 60% or even less can be set for training. In this particular case there is not test data

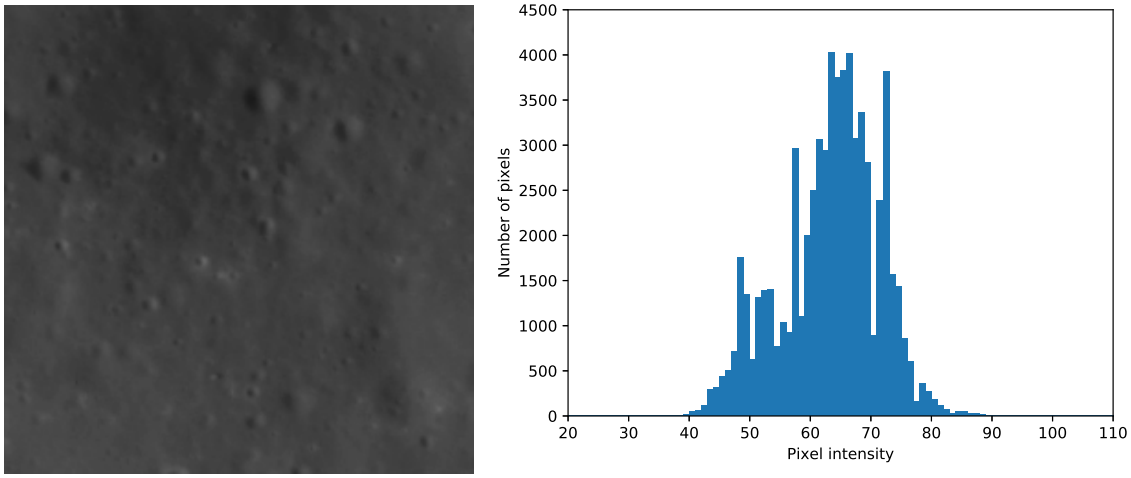


Figure 16: Histogram of an image before contrast limited adaptive histogram equilization

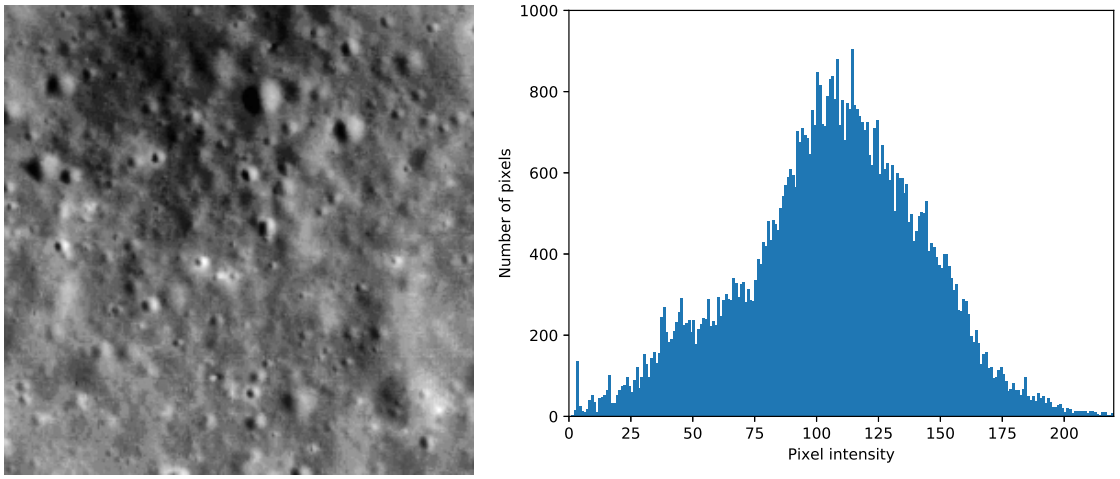


Figure 17: Histogram of an image after contrast limited adaptive histogram equilization

taken out from the data set because of limited annotations. As mentioned in introduction, test data is taken from another thesis student who has labeled all the craters manually.

The input images along with their corresponding segmentation maps are utilized to train network with stochastic gradient decent implementation of Keras (originally written in Cafe [Ronneberger et al., 2015]). Whereas Keras implementation can be found here: *https://github.com/zhixuhao/unet*. Inside training pipeline, the convolutions are unpadded which leads to a smaller output size after every layer. This decrease in size is of the factor of constant border width. The batch size is reduced to a single image to minimize the overhead and utilize maximum GPU memory. In this configuration momentum is kept high (0.99) such that a larger number of already seen training samples determine the update in ongoing optimization step. Momentum helps to accelerate gradients vectors in the right direction. A softmax function over the final feature map combined with cross entropy loss function determines the energy function. In softmax function as defined in eq.7, z_j denotes the activation in feature channel

j at the pixel position $z \in \Omega$ with $\Omega \subset \mathbb{Z}^2$. K is the number of classes and $f_j z$ is the approximated maximum function. This means that $f_j z$ will be close to 1 for j with maximum activation and vice versa for other values of j. The cross entropy function penalizes each deviation of $q(x)$ from 1 as defined in eq.1.

The weights are saved in a checkpoint when losses are less than previous saved checkpoint. Any of these saved weights could be utilized to apply on a test dataset but definitely saved weights with best performance on validation dataset is the best choice to be deployed on a test dataset.

5.4 Post-processing

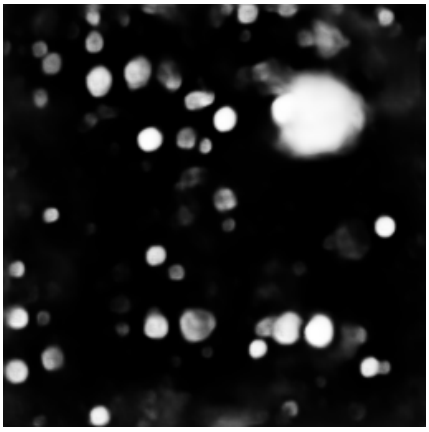
The resulting predictions from CNN model will be post processed in which noise from the image will be removed and the amount of detected craters will be counted. Then resultant craters will be projected on the original image to visually lookout for the missing ones. The test images will be of Apollo 17 landing site where number of craters are already annotated by another thesis student. The age of geographic site is also already known. Therefor it will also provide a good evaluation of how close CSFD and crater diameters are to the manually counted ones how different the final outcome (age estimation) is from the existing proposed age by the planetary science experts.

5.5 Calibration of Lunar Chronology Function

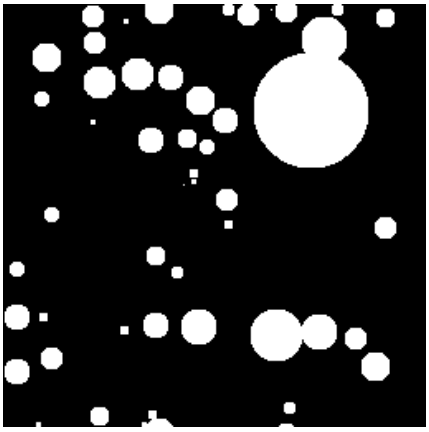
6 Results

6.1 Intersection Over Union (IoU)

(a) Crater image

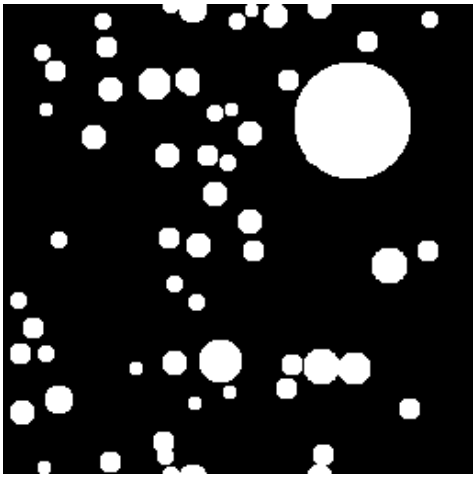


(a) Crater prediction



(b) Extracted craters

Figure 18: Model output of crater predictions



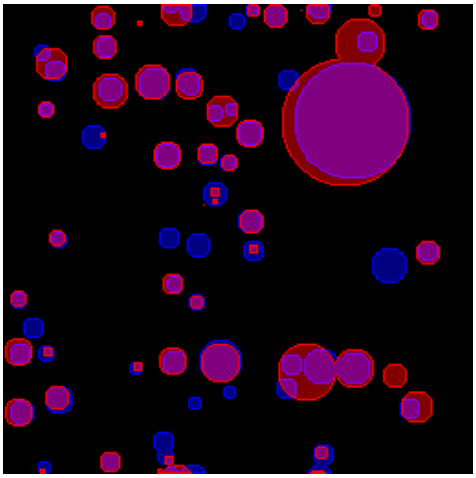


Figure 19: The image on left is ground truth of Figure 18 (a) and on right is the comparison of Figure 18 (b) with ground truth image. Red circles are extracted craters after post-processing and blue ones are the missed ones.

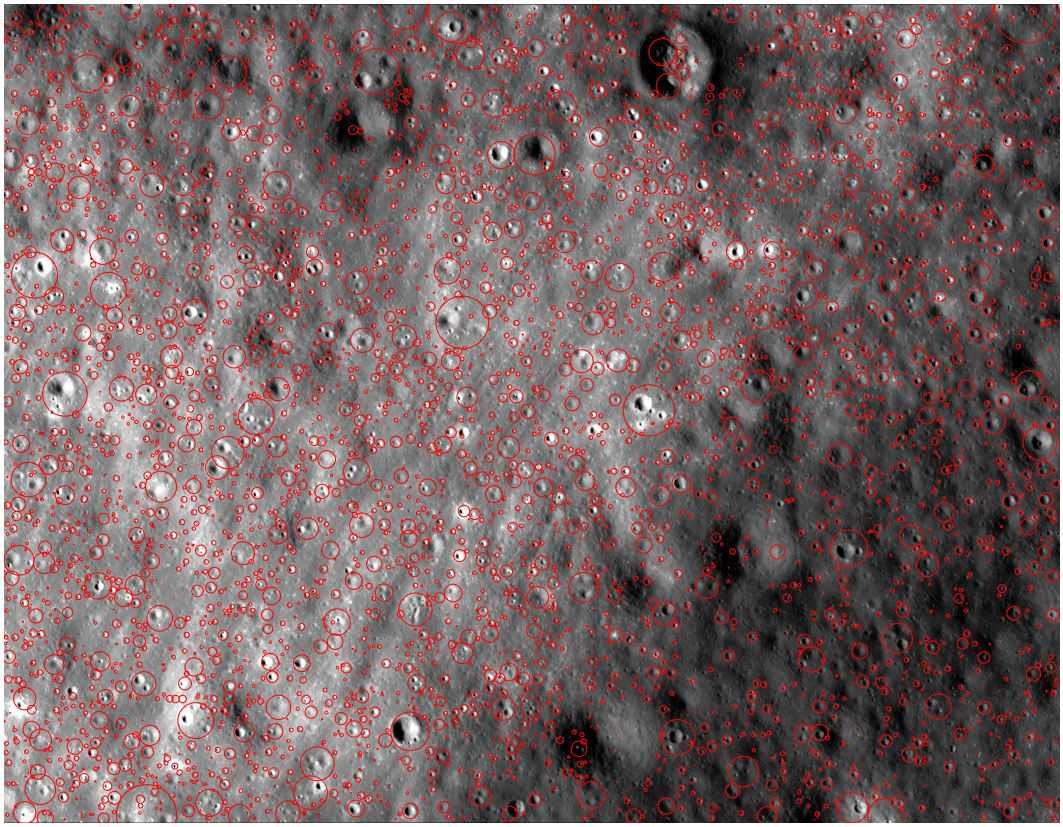


Figure 20: Projection of extracted craters on the test image.

References

[El Khiyari and Wechsler, 2016] El Khiyari, H. and Wechsler, H. (2016). Face recognition across time lapse using convolutional neural networks. *Journal of Information Security*, 7(03):141.

[Greeley and Gault, 1970] Greeley, R. and Gault, D. E. (1970). Precision size-frequency distributions of craters for 12 selected areas of the lunar surface. *The Moon*, 2(1):10–77.

[Hartmann, 1981] Hartmann, W. K. (1981). Chronology of planetary volcanism by comparative studies of planetary cratering. *Basaltic Volcanism on the Terrestrial Planets.*, pages 1049–1127.

[Hiesinger et al., 2000] Hiesinger, H., Jaumann, R., Neukum, G., and Head III, J. W. (2000). Ages of mare basalts on the lunar nearside. *Journal of Geophysical Research: Planets*, 105(E12):29239–29275.

[Ivanov, 2002] Ivanov, B. (2002). The comparison of size-frequency distributions of impact craters and asteroids and the planetary cratering rate. *Asteroids III*, 1:89–101.

[Koeberl, 1994] Koeberl, C. (1994). African meteorite impact craters: Characteristics and geological importance. *Journal of African Earth Sciences*, 18(4):263–295.

[Krizhevsky et al., 2012a] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012a). ImageNet classification with deep convolutional neural networks. 60(6):84–90.

[Krizhevsky et al., 2012b] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012b). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440.

[Martins et al., 2009] Martins, R., Pina, P., Marques, J. S., and Silveira, M. (2009). Crater detection by a boosting approach. *IEEE Geoscience and Remote Sensing Letters*, 6(1):127–131.

[Neukum, 1983] Neukum, G. (1983). Meteoritenbombardement und datierung planetarer oberflächen, habilitation dissertation for faculty membership. *Ludwig-Maximilians-University, Munich*, page 186.

[Neukum and Ivanov, 1994] Neukum, G. and Ivanov, B. (1994). Crater size distributions and impact probabilities on earth from lunar, terrestrial-planet, and asteroid cratering data. *Hazards due to Comets and Asteroids*, 1:359–416.

[Neukum et al., 1975] Neukum, G., König, B., Fechtig, H., and Storzer, D. (1975). Cratering in the earth-moon system-consequences for age determination by crater counting. In *Lunar and Planetary Science Conference Proceedings*, volume 6, pages 2597–2620.

[Opik, 1965] Opik, E. (1965). Mariner iv and craters on mars.

[Redmon et al.,] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. pages 779–788. IEEE.

[Robinson et al., 2010] Robinson, M., Brylow, S., Tschimmel, M., Humm, D., Lawrence, S., Thomas, P., Denevi, B., Bowman-Cisneros, E., Zerr, J., Ravine, M., et al. (2010). Lunar reconnaissance orbiter camera (lroc) instrument overview. *Space science reviews*, 150(1-4):81–124.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

[Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[Sawabe et al., 2006] Sawabe, Y., Matsunaga, T., and Rokugawa, S. (2006). Automated detection and classification of lunar craters using multiple approaches. 37(1):21–27.

[Silburt et al., 2019] Silburt, A., Ali-Dib, M., Zhu, C., Jackson, A., Valencia, D., Kissin, Y., Tamayo, D., and Menou, K. (2019). Lunar crater identification via deep learning. *Icarus*, 317:27–38.

[Soderblom, 1970] Soderblom, L. A. (1970). *The distribution and ages of regional lithologies in the lunar maria*. PhD thesis, California Institute of Technology.

[Stepinski et al., 2012] Stepinski, T., Ding, W., and Vilalta, R. (2012). Detecting impact craters in planetary images using machine learning. In *Intelligent data analysis for real-life applications: theory and practice*, pages 146–159. IGI Global.

[Stöffler and Ryder, 2001] Stöffler, D. and Ryder, G. (2001). Stratigraphy and isotope ages of lunar geologic units: Chronological standard for the inner solar system. In *Chronology and evolution of Mars*, pages 9–54. Springer.

[Williams et al., 2018] Williams, J.-P., van der Bogert, C. H., Pathare, A. V., Michael, G. G., Kirchoff, M. R., and Hiesinger, H. (2018). Dating very young planetary surfaces from crater statistics: A review of issues and challenges. *Meteoritics & Planetary Science*, 53(4):554–582.