SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

PROJECT SYNOPSIS
on

INTRACRANIAL HEMORRHAGE DETECTION

submitted by

NAME	SAP No.
RITVIK KHANDELWAL	**60002180081**
SATYAM UPADHYAY	**60002180101**
SHRENIK GANGULI	**60002180105**
UNMESH PHATERPEKAR	**60002180113**

under the guidance of

<Dr Sunil Karamchandani>

<Assistant Professor at Dwarakadas J Sanghvi College of Engineering>

DEPARTMENT OF

ELECTRONICS AND TELECOMMUNICATION ENGINEERING

Academic Year : 2021-2022

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

Shri Vile Parle Kelavani Mandal's

Dwarkadas J. Sanghvi College of Engineering

Plot no. U-15, JVPD Scheme, Bhaktivedanta Swami Marg,
Vile Parle (W), Mumbai – 400 056

Department of Electronics and Telecommunication Engineering

This is to certify that the Project Report Stage – II

"**Optimizing Bank's Financial Risk by Forecasting Future NPAs Using Machine Learning**"

Submitted by:

1. RITVIK KHANDELWAL
2. SATYAM UPADHYAY
3. SHRENIK GANGULI
4. UNMESH PHATERPEKAR

students of Department of **Electronics and Telecommunication Engineering** have successfully completed their **Project Stage – II** required for the fulfillment of **SEM VIII** as per the norms prescribed by the **University of Mumbai** during the First half of the year 2022. The project synopsis has been assessed and found to be satisfactory.

_____ _____
Internal Guide **External Guide**

_____ _____
Head of Department **Principal**

_____ _____
Internal Examiner **External Examiner**

TABLE OF CONTENTS

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature

Name

1. RITVIK KHANDELWAL
2. SATYAM UPADHYAY
3. SHRENIK GANGULI
4. UNMESH PHATERPEKAR

SAP ID:

1. 60002180081
2. 60002180101
3. 60002180105
4. 60002180113

Date: 3rd April 2022

ABSTRACT:

Hemorrhage in the brain (Intracranial Hemorrhage) is one of the top five fatal health problems. The hemorrhage causes bleeding inside the skull (typically known as cranium). It accounts for approximately 10% of strokes in the U.S thus diagnosing it quickly and efficiently is of utmost importance. Intracranial Hemorrhage can have many consequences depending on the location, size, type of bleed in the brain. The relationship is not that simple to interpret. A small hemorrhage might be equally fatal as a big one if it is in a critical location. If a patient shows acute hemorrhage symptoms such as loss of consciousness, then an immediate diagnosis is required to carry out the initial tests, but the procedure is very complicated and often time-consuming. The delay in diagnosis and treatment can have severe effects on patients, especially the ones in need of immediate attention. The solution proposed in this project aims to make this process efficient by automating the preliminary step to identify the location, type, and size of the bleed, the doctors can provide a better diagnosis and a faster one as the work of identification is done by the algorithm. Also, the algorithm treats every scan with the same scrutiny, reducing human error and learning more as it goes. Thus, the challenge is to build an algorithm to detect acute intracranial hemorrhage and its subtypes. This project will be a step towards identifying each of these relevant attributes for a better diagnosis. In our project, we aim to automate the process of detecting these bleeds in order to relieve overworked radiologists and flag patients in need of immediate attention.

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 (INTRODUCTION AND LITERATURE REVIEW):

INTRODUCTION:

An intracranial hemorrhage is a kind of bleeding which occurs within the brain. The symptoms may vary based on the location of the hemorrhage, it may include total or limited loss of consciousness, abrupt shivering, numbness on one side of the body, loss of motion, serious migraine, drowsiness, problems with speech and swallowing. It is important to get the individual to a hospital and this condition must be treated as a medical emergency. Normally during this emergency, a Computerized Tomographic (CT) scan will be taken by the radiologist and they will identify the location of the hemorrhage and its subtypes. This process is often time consuming and complicated. Through this paper we would like to present a model which uses rich image dataset provided by Radiological Society of North America (RSNA) in collaboration with members of the American Society of Neuroradiology and MD.ai. This will help the medical community identify the presence, location and type of hemorrhage in order to quickly and effectively treat affected patients. This algorithm can provide early detection of people at high risk of severe bleeding in the brain. The algorithm comprises of a neural network architecture designed to detect intracranial hemorrhage while tackling several challenges such as relatively small bleed size and high variance within the brain.

Hemorrhages that occur within the skull or brain generally happen from either external or internal causes. A hemorrhage can rapidly cause brain damage and can be life- threatening. This bleeding can be caused by a blood vessel in the brain that has leaked or ruptured (torn). Since the brain cannot store oxygen, it relies upon a series of blood vessels to supply oxygen and nutrients. When a hemorrhage occurs, oxygen will no

longer be able to reach brain tissue and when oxygen and nutrients are being deprived for more than three to four minutes the brain cells starts to die. The affected nerve cells and the related functions they control are damaged as well.

The different types of hemorrhage include:

• Intraparenchymal hemorrhage is when blood leak happens completely within the brain itself.

• Intraventricular or subarachnoid hemorrhage is when blood has leaked into the spaces of the brain that normally contain cerebrospinal fluid (the ventricles or subarachnoid cisterns).

• Extra-axial hemorrhage is when blood is collected in the tissue coverings that surround the brain (e.g. subdural or epidural subtypes figure).

Patients may exhibit more than one type of cerebral hemorrhage; which may appear on the same image. While small hemorrhages are less morbid than large hemorrhages typically, even a small hemorrhage can lead to death because it is an indicator of another type of serious abnormality (e.g. cerebral aneurysm).

A pictorial representation of the process involved in identifying and classifying various hemorrhages is given in Fig. 1.

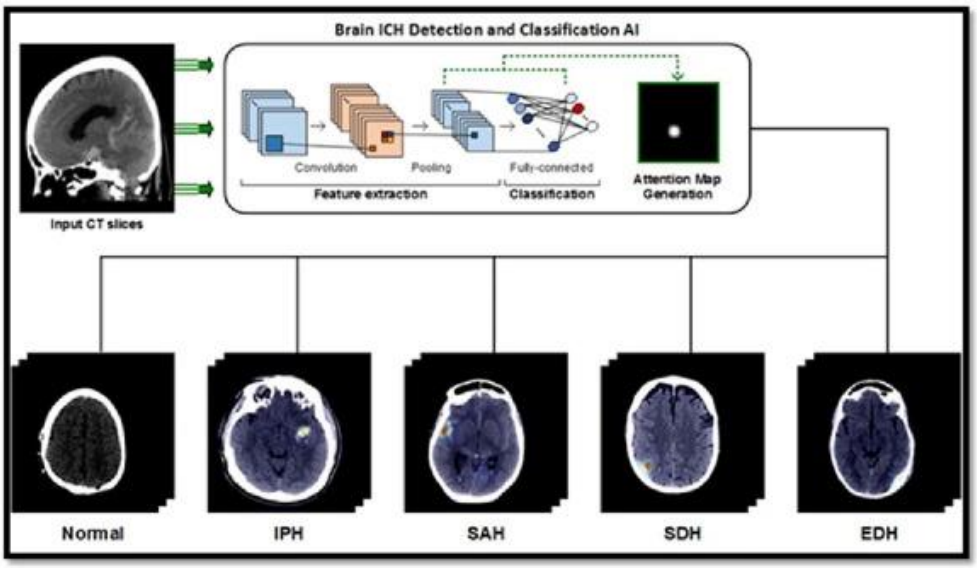


Figure 1: A pictorial representation of the process involved in identifying and classifying various hemorrhages

LITERATURE REVIEW:

1.1 ***Overview***

Intracranial hemorrhage image attenuation significantly over-lap with those of gray matter, meaning that simple thresholding is ineffective [2]. A variety of methods have been developed to detect intracranial hemorrhages or to measure hemorrhage volume using standard image processing approaches [1, 3-11]. These approaches typically take a sequential approach of detecting the head within the image, aligning the head, removing the skull, compensating for CT cupping artifacts, extracting handcrafted features from the imaged brain tissue, and classifying intracranial hemorrhage voxels based on the features. Most have used small datasets of 11-30 cases. In some approaches, manual intervention is sometimes needed, such as for aligning the head [1]. Comparing performance of these methods is difficult because of the different types of intracranial hemorrhages considered and because of the different performance metrics used. Some papers only consider large intracranial hemorrhages, which are the easiest to detect. Previous work has not reported on the ability to detect subtle hemorrhages, particularly along the falx cerebri. Often detection is assessed as a percentage of intracranial hemorrhage voxels, rather than considering the detection of each intracranial hemorrhage, which is more informative to a radiologist. One of the more sophisticated approaches [1] specifically focuses on detecting small acute intracranial hemorrhages. In addition to executing the processing steps listed above, this approach also registers the head to an anatomic model and adjusts detections based on anatomic regions. A follow-on study [11] determined that this computer-aided diagnosis system improved detection of these hemorrhages by emergency physicians and radiology residents, although not by radiology specialists.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alexander A Alemi [12]**:** This enlightened us about the convolutional network acting as core to the largest advancements in the image recognition performance in these years. One such example is about the inception architecture that has obtained an ultimate performance at relatively low computational costs. This paper also depicts how proper activation scaling stabilizes the training of inception Networks.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun [13]**:** This paper explains to us the difficulties faced while training extremely deep neural networks hence a learning framework was presented to ease out the training. The layers are reformulated as the residual function regarding the input from the layer. The output of these residual nets obtains only around 3.75% error on the ImageNet test set.

Mayank Chawla, Saurabh Sharma, Jayanthi Sivaswamy, L.T Kishore [14]**:** This paper explains to us about the importance of Computed Tomographic (CT) Images. The CT images are prominently used in the diagnosis of stroke. Here an advanced automated method is implemented to detect and categorize an abnormality into acute infarct, chronic infarct, and hemorrhage.

Monika Grewal, Muktabh Mayank Srivastava, Pulkit Kumar, Srikrishna Varadarajan [4]**:** This paper describes an informative learning approach about the process of brain hemorrhage detection from CT scans. The model that they obtained follows the procedures used by the radiologist in analysing a 3D CT Scan in real world.

Tong Duc Phong, Hieu N.Duong, Hien T. Nguyen, Tran van Hoa [15]**:** This paper proposes an approach to diagnose and analyse brain hemorrhage with the help of deep learning. There are three types of convolutional Neural network LeNet, GoogLeNet, and Inception Resnet were employed. It was confirmed that among the three LeNet was the most time-consuming model.

Marcus Badgeley, Javin Schefflein, Margaret Pain, Andres Su, and **Michael Cai** [16]**:** This paper determines that rapid diagnosis and treatment of severe neurological illness such as stroke Hemorrhage and Hydrocephalus are critical to achieving positive outcomes. However, these illnesses are often recognizable by their initial symptoms. Their critical means of diagnosis is done via rapid imaging.

Mohammad R. Arbabshirani, Brandon K. Fornwalt, and Gino J. Mongelluzzo [17]**:** This paper describes the fact that Intracranial Hemorrhage (ICH) requires instant diagnosis to obtain a positive outcome. This also states the hypothesis that algorithms used in, machine learning could automatically analyse CT of the head, prioritize the worklist and hence reduce the time used in diagnosing the ICH.

Sasank Chilamkurthy, Rohit Ghosh, Swetha Tanamala, Mustafa Biviji, Prashant Warier [18]**:** This paper describes the importance of CT scan images and how these images have become the current standard as the initial step in the process of diagnosing patients with trauma or strokes symptoms. The objective was to develop, validate a set of algorithms to find if the abnormality would be classified under ICH, IPH, IVH, SDH, EDH, and SAH hemorrhages.

1.2 *Development and Validation of Deep Learning Algorithms for CT Scans*

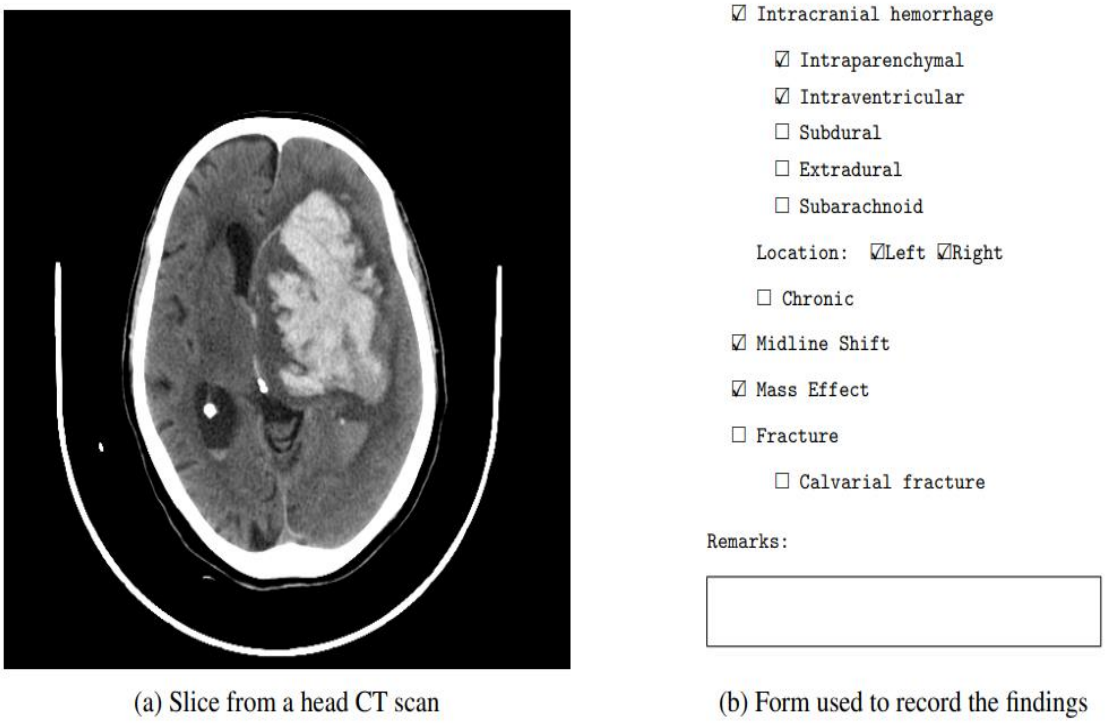
Deep learning has been used extensively in medical imaging in the last decade whether it is detecting diabetes using retinopathy or pneumonia using Chest X-ray. Image segmentation has been used for colour-coding scans which are then used for applications like automatic measurement of organs, cell counting, or simulations based

on the extracted boundary information. A recent paper by the Qure.ai team [22] shows the use of deep learning for classifying MRIs. Their objective was to develop and validate a set of deep learning algorithms for automated detection of following key findings from non-contrast head CT scans: intracranial hemorrhage (ICH) and its types, intraparenchymal (IPH), intraventricular (IVH), subdural (SDH), extradural (EDH) and subarachnoid (SAH) hemorrhages, calvarial fractures, midline shift and mass effect.

They retrospectively collected a dataset containing 313,318 head CT scans along with their clinical reports from various centers. A part of this dataset (Qure25k dataset) was used to validate and the rest to develop algorithms. Additionally, a dataset (CQ500 dataset) was collected from different centers in two batches B1 & B2 to clinically validate the algorithms.

Original clinical radiology report and consensus of three independent radiologists were considered as gold standard for Qure25k and CQ500 datasets respectively. Area under receiver operating characteristics curve (AUC) for each finding was primarily used to evaluate the algorithms.

Qure25k dataset contained 21,095 scans (mean age 43.31; 42.87% female) while batches B1 and B2 of CQ500 dataset consisted of 214 (mean age 43.40; 43.92% female) and 277 (mean age 51.70; 30.31% female) scans respectively. On Qure25k dataset, the algorithms achieved AUCs of 0.9194, 0.8977, 0.9559, 0.9161, 0.9288 and 0.9044 for detecting ICH, IPH, IVH, SDH, EDH and SAH respectively. AUCs for the same on CQ500 dataset were 0.9419, 0.9544, 0.9310, 0.9521, 0.9731 and 0.9574 respectively. For detecting calvarial fractures, midline shift and mass effect, AUCs on Qure25k dataset were 0.9244, 0.9276 and 0.8583 respectively, while AUCs on CQ500 dataset were 0.9624, 0.9697 and 0.9216 respectively.



(a) Slice from a head CT scan

(b) Form used to record the findings

Figure 2: Schematic of the reading process of the CQ500 Dataset

This study demonstrates that deep learning algorithms can accurately identify head CT scan abnormalities requiring urgent attention. This opens up the possibility to use these algorithms to automate the triage process. They may also provide a lower bound for quality and consistency of radiological interpretation.

Characteristic	Qure25k dataset	CQ500 dataset batch B1	CQ500 dataset batch B2
No. of scans	21095	214	277
No. of readers per scan	1	3	3
PATIENT DEMOGRAPHICS			
Age			
No. of scans for which age was known	21095	189	251
Mean	43.31	43.40	51.70
Standard deviation	22.39	22.43	20.31
Range	7 – 99	7 – 95	10 – 95
No. of females / No. of scans for which sex was known (percentage)	9030/21064 (42.87%)	94/214 (43.92%)	84/277 (30.31%)
PREVALENCE			
No. of scans (percentage) with			
Intracranial hemorrhage	2494 (11.82)	35 (16.36)	170 (61.37)
Intraparenchymal	2013 (9.54)	29 (13.55)	105 (37.91)
Intraventricular	436 (2.07)	7 (3.27)	21 (7.58)
Subdural	554 (2.63)	9 (4.21)	44 (15.88)
Extradural	290 (1.37)	2 (0.93)	11 (3.97)
Subarachnoid	611 (2.90)	9 (4.21)	51 (18.41)
Fractures	1653 (7.84)	8 (3.74)	31 (11.19)
Calvarial Fractures	992 (4.70)	6 (2.80)	28 (10.11)
Midline Shift	666 (3.16)	18 (8.41)	47 (16.97)
Mass effect	1517 (7.19)	28 (13.08)	99 (35.74)

Table 1: Dataset characteristics of CQ500 and Qure25k

In summary their approach included the use of CNNs as well as NLP techniques on the corresponding medical reports. The dataset included 313,318 anonymous CT scans from several centers in India. One major problem they faced was that of 3D scans. The mean age of the dataset was 43.4 years and 42.87% of the population was female. They managed to achieve an AUC of 0.94 ± 0.3 on all the sub categories.

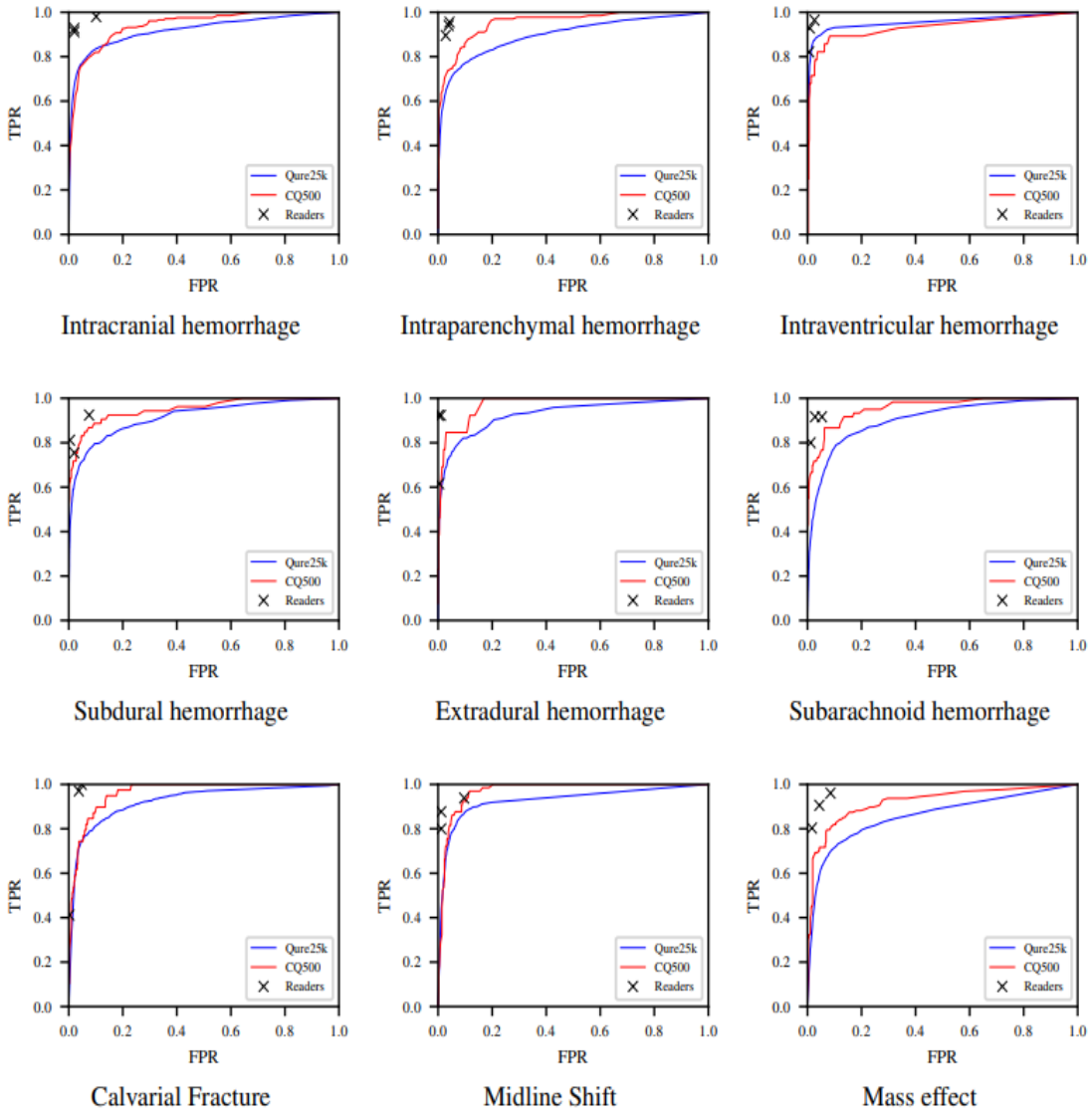


Figure 3: Receiver operating characteristic (ROC) curves for the algorithms on Qure25k and CQ500 datasets. Blue lines are for the Qure25k dataset and Red lines are for the CQ500 dataset. Readers' TPR and FPR against consensus on CQ500 dataset are plotted along with the ROCs for comparison

1.3 *Detecting Intracranial Hemorrhage with Deep Learning*

In [28], Initial results are reported on automated detection of intracranial hemorrhage from CT, which would be valuable in a computer-aided diagnosis system to help the radiologist detect subtle hemorrhages. They have also taken a classic approach involving multiple steps of alignment, image processing, image corrections, handcrafted feature extraction, and classification. Their work work also uses a deep convolutional neural network to simultaneously learn features and classification, eliminating the multiple hand-tuned steps. Performance is improved by computing the mean output for rotations of the input image. Postprocessing is additionally applied to the CNN output to significantly improve specificity. Their database consists of 134 CT cases (4,300 images) which is highly small to convert its applicability to real world scenarios. It is divided into 60, 5, and 69 cases for training, validation, and test. Each case typically includes multiple hemorrhages. Performance on the test set was 81% sensitivity per lesion (34/42 lesions) and 98% specificity per case (45/46 cases). The sensitivity is comparable to previous results (on different datasets), but with a significantly higher specificity. In addition, insights are shared to improve performance as the database is expanded which is still not enough compared to our work.

Their network architecture is composed of 9 convolutional blocks and operates on 2D slices. Each block contains k convolutional layers, where a layer is defined as a 3x3 convolution followed by batch normalization and a rectified linear unit (RELU). They experiment with $k \in \{1,2\}$ and find that both models perform similarly. Following the first 4 blocks, features are down sampled using 2x2 max-pooling. After the next 4 blocks, features are up sampled using 2x2 nearest neighbor expansion followed by a 3x3 convolution, batch normalization, and RELU. Following [19], features from blocks 1, 2, 3, and 4 are concatenated with the up sampled outputs from blocks 8, 7, 6, and 5, respectively. Finally, the output of the 9th convolutional block is processed by a 1x1 convolutional layer to produce pixel-level detections. Two forms of data augmentation were used to improve the performance of the model: random left-right flipping and random rotation of ±10 degrees. In thir study it was found that data augmentation was beneficial during both training and testing therefore we took inspiration from the same and extended it to our work. Accordingly, outputs for evaluation are produced by averaging the outputs from multiple random augmentations of the same slice.

The results obtained were them are shown in the figure below.

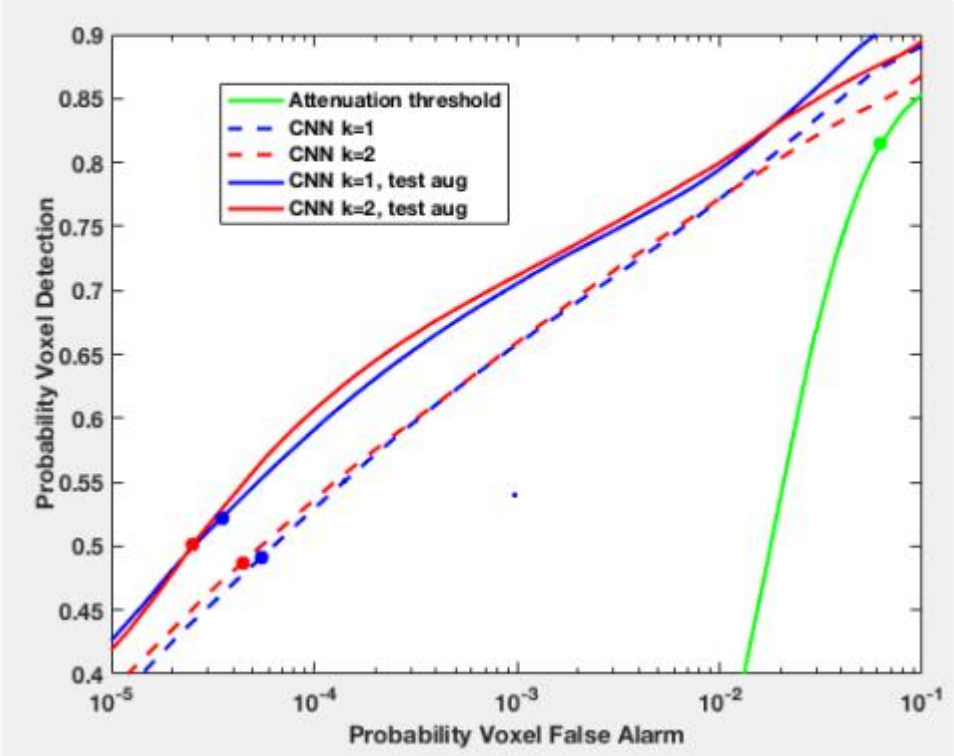


Figure 4: Voxel-wise ROC. Solid circles plot operating points for attenuation thresholding at 40 and 80 HU and for CNN output threshold at 0.5.

Other interesting papers include Expert-level detection of acute intracranial hemorrhage on head computed tomography using deep learning.[2], Extracting 2D weak labels from volume labels using multiple instances learning in CT hemorrhage detection. Computed tomography (CT) of the head is used worldwide to diagnose neurologic emergencies. However, expertise is required to interpret these scans, and even highly trained experts may miss subtle life-threatening findings. For head CT, a unique challenge is to identify, with perfect or near-perfect sensitivity and very high specificity, often small subtle abnormalities on a multi-slice cross-sectional (three-dimensional [3D]) imaging modality that is characterized by poor soft tissue contrast, low signal-to-noise using current low radiation-dose protocols, and a high incidence of artifacts. In this paper they trained a fully convolutional neural network with 4,396 head CT scans performed at the University of California at San Francisco and affiliated hospitals and compared the algorithm's performance to that of 4 American Board of Radiology (ABR) certified radiologists on an independent test set of 200 randomly selected head CT scans. Their algorithm demonstrated the highest accuracy to date for this clinical application, with a receiver operating characteristic (ROC) area under the curve (AUC) of 0.991 ± 0.006 for

identification of examinations positive for acute intracranial hemorrhage, and also exceeded the performance of 2 of 4 radiologists. However, the dataset they are working on is again very small. But the use of deep neural networks and their utilization to achieve maximum convergence is something we were inspired from.

1.4 *Intracranial Hemorrhage Detection*

In [27], the figure below fig no, the x-axis of the graphs denotes the epochs (or, the number of the times the model is trained on the training data or validation data). The y-axis of the graphs denote the accuracy or losses of the model with respect to the increasing number of epochs. From the graph, it was observed that they have achieved an accuracy (acc) and validation accuracy (value acc) between 0.95 (95%) and 0.96 (96%). Losses are scores and do not have units, because it is the average of the difference between the predicted and true values. From the graph, it was inferred that they had achieved a loss of 0.12 – 0.14 at the end of 10 epochs. These losses occur due to lesser size of data as well as lesser training periods, with increased training periods, we can improve to a greater extent.

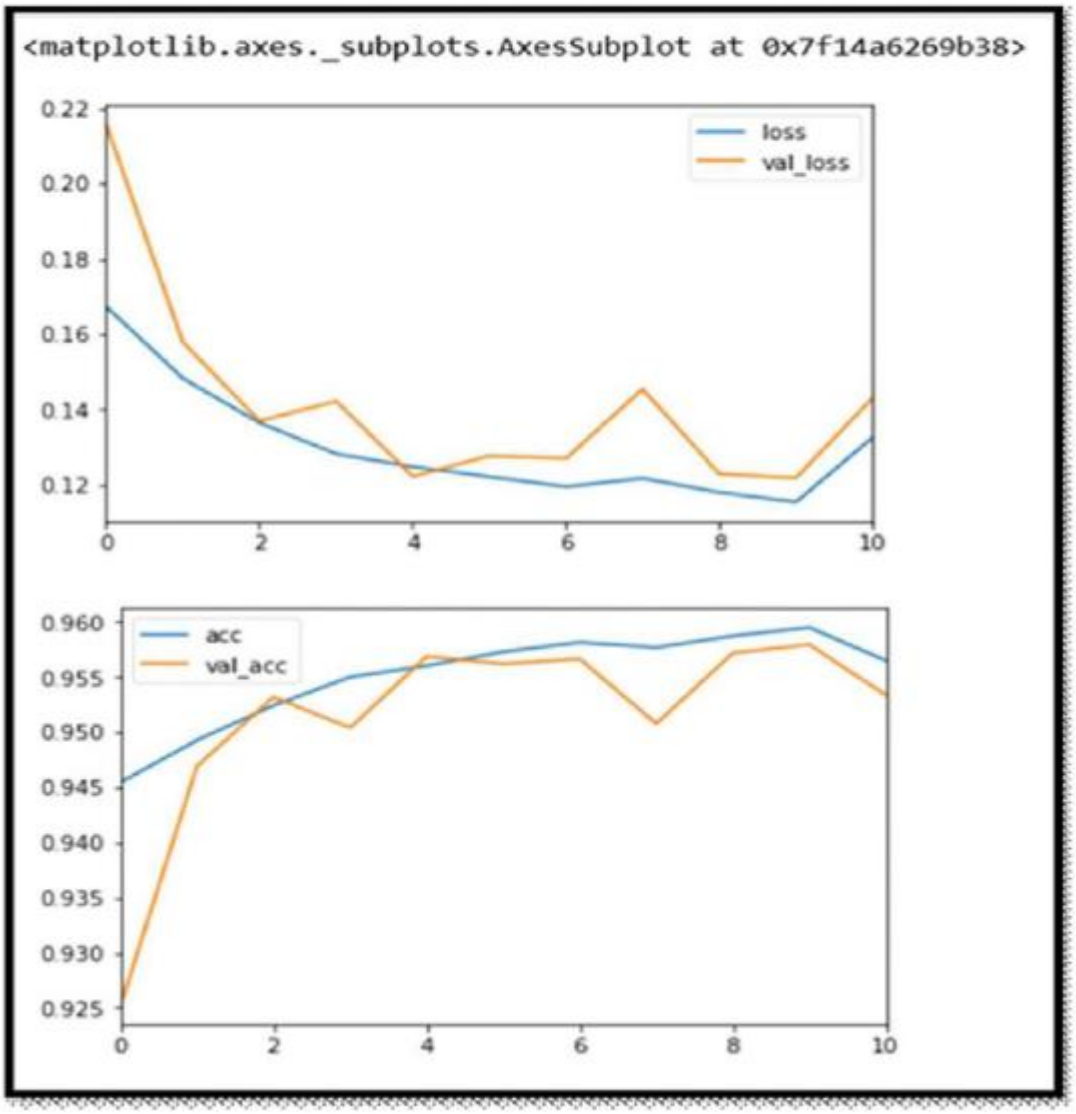


Figure *5*

Improvements that can be made, to further minimize errors

1) Changes in learning rate and learning rate schedule.

2) Larger input size.

3) Longer training period.

4)Addition of more dense layers and regularizing these layers using the Dropout () method, so as to prevent

overfitting.

5)Trying some more optimal windowing technique so as to expose the model to some better and clearer hemorrhage contrast images.

In our project and paper, we have addressed the parameters 1), 2), 3) and 4) by leveraging the power of fast.ai library and using differential learning rates. The idea of decreasing the learning rate during training has been around forever. It's just called learning rate annealing. But the idea of gradually increasing it at the start is much more recent and comes from Leslie Smith [5]

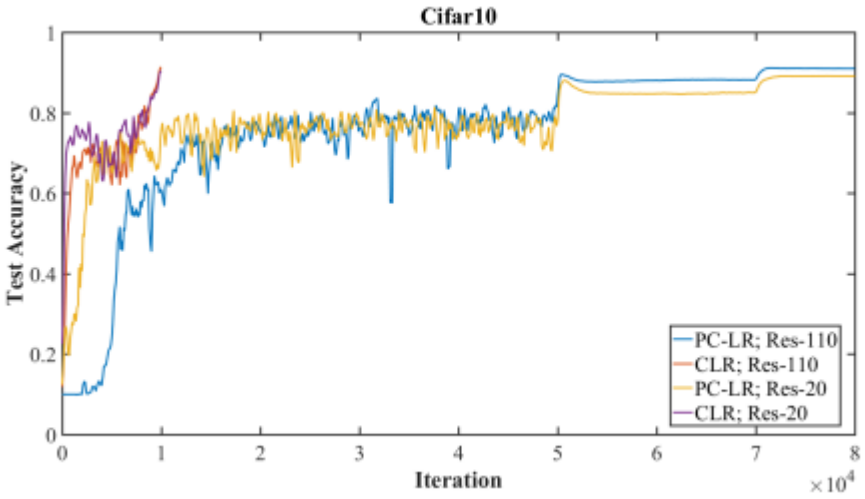
1.5 ***Super-Convergence: Fast Training of Neural Networks Using Large Learning Rates***

In [26], The contributions of this paper include:

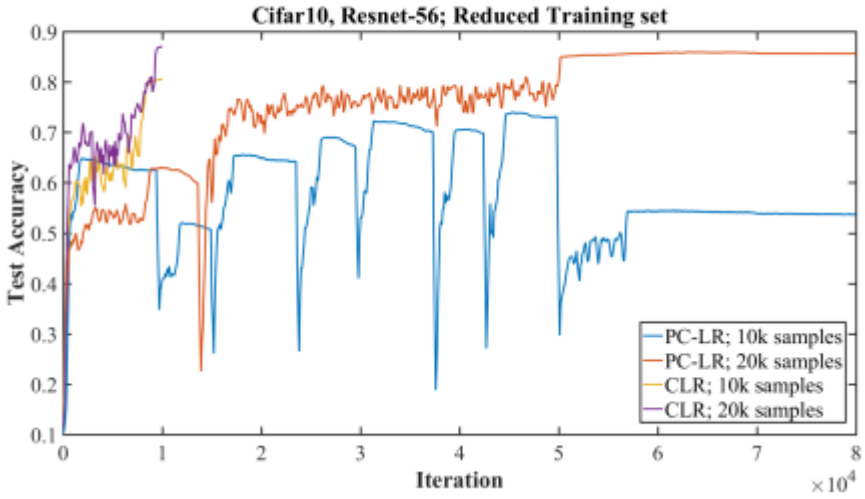
1. Systematically investigates a new training methodology with improved speed and performance.
2. Demonstrates that large learning rates regularize training and other forms of regularization must be reduced to maintain an optimal balance of regularization.
3. Derives a simplification of the second order, Hessian-free optimization method to estimate optimal learning rates which demonstrates that large learning rates find wide, flat minima.
4. Demonstrates that the effects of super-convergence are increasingly dramatic when less labelled training data is available.

Experiments with Resnet was run with a number of layers in the range of 20 to 110 layers; that is, they ran experiments on residual networks with l layers, where l = 20 + 9n; for n = 0, 1, ...10. Figure 4b illustrates the results for Resnet-20 and Resnet-110, for both a typical (piecewise constant) training regime with a standard initial learning rate of 0.35 and for CLR with a step size of 10,000 iterations. The accuracy increased due to super-convergence is greater for the shallower architectures (Resnet20: CLR 90.4% versus piecewise constant LR schedule 88.6%) than for the deeper architectures (Resnet-110: CLR 92.1% versus piecewise constant LR schedule 91.0%). There are discussions in the deep learning literature on the effects of larger batch size and the generalization gap [Keskar et al., 2016, Jastrz¸ebski et al., 2017, Chaudhari and Soatto, 2017, Hoffer et al., 2017]. Hence, they investigated the effects total mini-batch size used in super-convergence training and found a small improvement in performance with larger batch sizes, as can be seen in Figure 5a. In addition, Figure 5b shows that the generalization gap (the difference between the training and test accuracies) are approximately equivalent for small and large mini-batch sizes. This result differs than results reported elsewhere [Keskar et al., 2016] and illustrates that a consequence of training with large batch

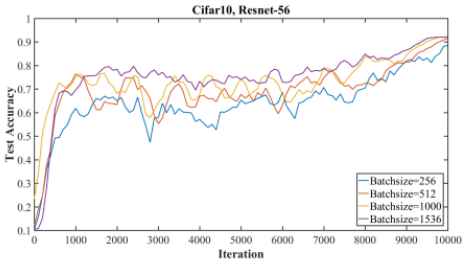
sizes is the ability to use large learning rates



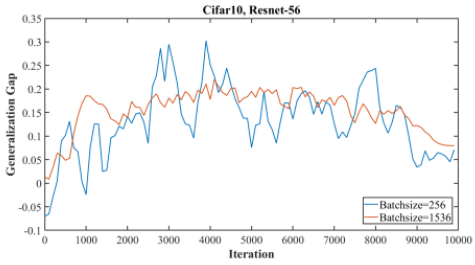
(a) Comparison of test accuracies for Cifar10/Resnet-56 with limited training samples.



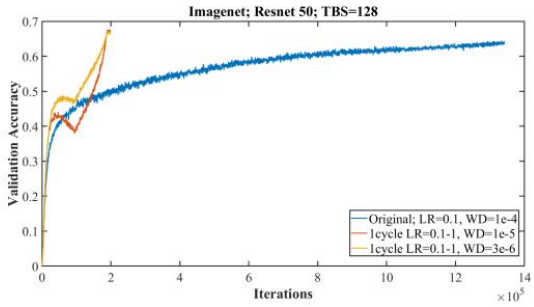
(b) Comparison of test accuracies for Resnet-20 and Resnet-110.



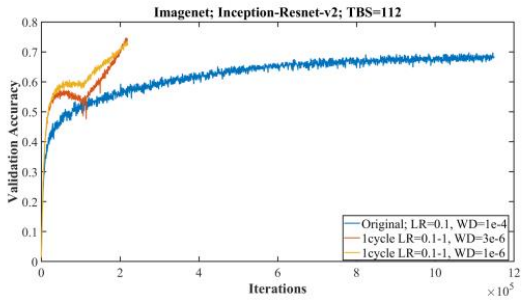
(a) Comparison of test accuracies for Cifar-10, Resnet-56 for various total batch sizes.



(b) Comparison of the generalization gap (training - test accuracy) for a small and a large batch size.



(a) Resnet-50



(b) Inception-resnet-v2

Figure 6: Training resnet and inception architectures on the imagenet dataset with the standard learning rate policy (blue curve) versus a 1cycle policy that displays super-convergence. Illustrates that deep neural networks can be trained much faster (20 versus 100 epochs) than by using the standard training methods.

To achieve maximum convergence and fast training of neural networks a thorough understanding of this paper was required so our group examined and used some of the state-of-the-art methodologies to achieve the best results.

[3] In the latter, they used bags of 2D slices and calculated the losses per slice. These losses were then max pooled together to calculate the gradient and update parameters of the model. However, they used only about 600 images for this project.

CHAPTER 2 (THEORY):

2.1 *Data Cleaning:*

Data cleaning is the procedure of preparing data for the required analysis by removing/modifying data that is improper, incorrect, incomplete, irrelevant, duplicated, or which is not properly formatted.

This data is usually not that important or needy when it comes to analyzing the data because it leads to hinderance of the process and thus providing results which are inaccurate. There are a number of methods for cleaning the data depending on how it is stored along with the answers which are to be found. Basically, data cleaning is not simply about erasing information to make space for new data, but rather finding a way to maximize a data set's accuracy without necessarily deleting information.

For one, data cleaning includes a number of actions than removing data, such as fixing spelling and syntax errors, standardizing data sets, and correcting mistakes such as empty fields, missing codes, and identifying duplicate data points. Data cleaning is considered a foundational element of the data science basics, as it plays an important role in the analytical process and uncovering reliable answers.
Data cleaning is also known as data scrubbing. Data cleaning is a process which ensures the set of data is correct and accurate. Data accuracy and consistency, data integration is checked during data cleaning. Data cleaning can be applied for a set of records or multiple sets of data which need to be submerged.

Data cleaning is performed by reading all records in a set and verifying their accuracy. Typing and spelling errors are to be rectified. Mis-labelled data if available is labelled and filed. Incomplete or missing entries are completed. Unrecoverable records are to be get rid of, for not to take space and inefficient operations. Data cleaning is a long-standing problem that has attracted much research interest in the past years in the databases community.

Data cleaning is a critically important step in any machine learning project.
In tabular data, there are many different statistical analysis and data visualization techniques you can use to explore your data in order to identify data cleaning operations you may want to perform.Before jumping to the sophisticated methods, there are some very basic data cleaning operations that you probably should perform on every single machine learning project. These are so basic that they are often overlooked by seasoned machine

learning practitioners, yet are so critical that if skipped, models may break or report overly optimistic performance results.

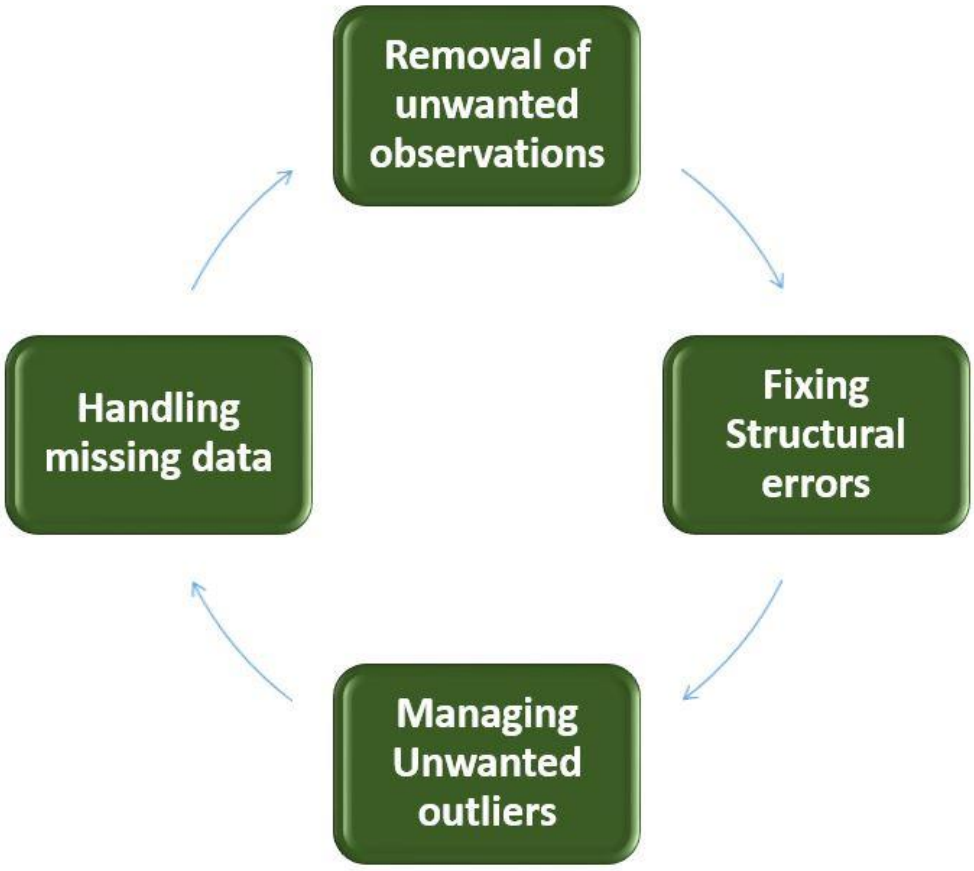


Figure 7

Why is data cleaning important?

Data cleansing ensures you only have the most recent files and important documents, so when you need to, you can find them with ease. It also helps ensure that you do not have significant amounts of personal information on your computer, which can be a security risk.

2.2 ***Data Exploration:***

At times some researchers tend to spend a specific amount of time on model architecture designing and tuning of parameters, the importance of data exploration cannot be easily ignored. For example, imagine having developed a perfect model. However, if the data breaks the assumption of required model or the data contains errors, we will not be able to get the craved results from our perfect model. Without the help of data exploration, we may even spend most of the time checking our model without even realizing what is the problem in the dataset.

Some of the tools for interactive data exploration significantly help to increase the chance that users discover false discoveries. They allow the users to visually examine many hypothesis and make connections with simple interactions, thus incurring the issue frequently known in statistics as the "multiple hypothesis testing error". A proposition of a solution to integrate the control of multiple hypothesis testing into interactive data exploration systems is mandatory. A key perception is that the existing methods for controlling the false discovery rate (such as FDR) are not directly applicable to the interactive data exploration.

Most of the research on visualization has focused on improving the rendering quality and speed, and thus enhancing the changes in features of the data. Recently, a significant amount of emphasis has been placed on focus + context (F+C) techniques for data exploration in addition to viewing transformation and hierarchical navigation. However, most of the existing data exploration techniques rely on the manipulation of viewing attributes of the rendering system or optical attributes of the data objects, with users being passive viewers. We propose a more active approach to data exploration, which attempts to mimic how we would explore data if we were able to hold it and interact with it in our hands. This involves allowing the users to physically or actively manipulate the geometry of a data object. While this approach has been traditionally used in applications, such as surgical simulation, where the original geometry of the data objects is well understood by the users, there are several challenges when this approach is generalized for applications, such as flow and information visualization, where there is no common perception as to the normal or natural geometry of a data object.

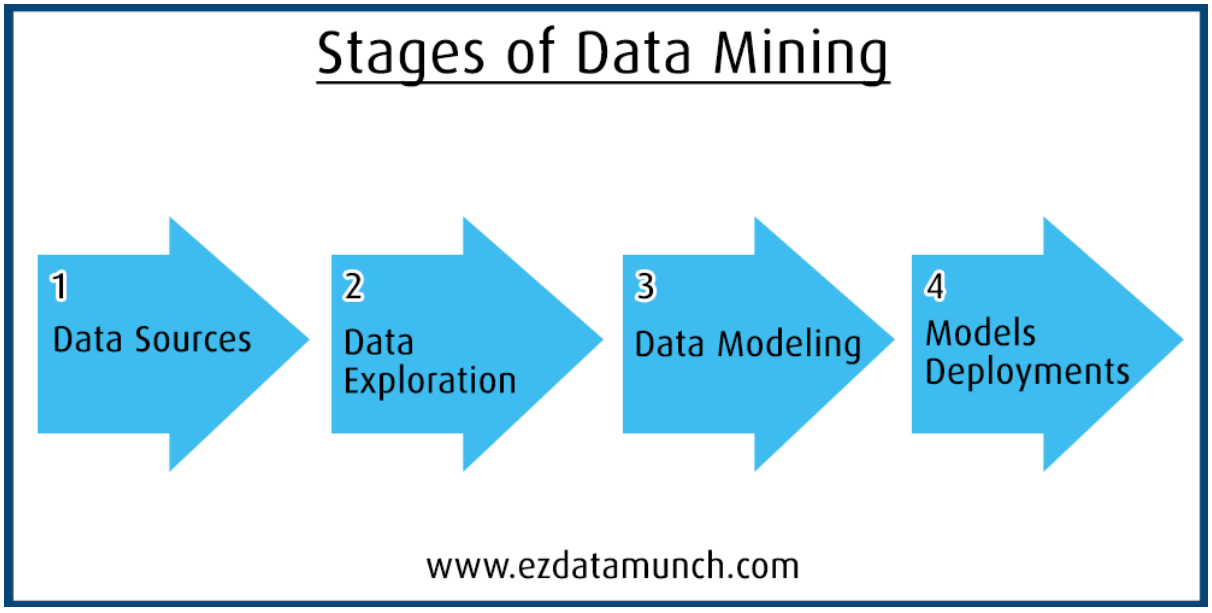


Figure 8

The art of data exploration is not easy and has no shortcuts to it. Times may arise when it is difficult to improve a model's accuracy, and knowledge of various data exploration techniques will be a saving grace. In the field of machine learning, the area of data exploration has become quite an area of interest. It might be still evolving; however, by employing machine learning and understanding common patterns, it is indeed possible to identify relationships, patterns or algorithms in a given set of data.

Employment of machine learning is crucial as it mitigates manual labour and time involved in data exploration and reduces the errors that may occur in the employment of manual inspection, trial and or error, or other traditional exploration techniques. Data exploration is a methodology that is very much like initial data analysis. A data analyst utilizes visual exploration to comprehend the contents of a dataset and its attributes, instead of using data management systems. These attributes could include the size, culmination, accuracy, potential connection between different data components or data files/ tables. Both manual (drill down or filtering of data to understand similar patterns in data) and automated (data profiling or visualization) methods may be used for data exploration. Essentially, Data exploration is pruning of data to remove unusable parts and identify potential relationships between different types of data.

2.3 ***Data Augmentation:***

Depending on the performance of a number of Machine Learning models, and particularly deep learning models, depends on the quality, the quantity and relevancy of training data. However, insufficient data is one of the most common challenges in implementing machine learning in the enterprise. This is because collecting such data can be costly and time-consuming in many cases.

Companies can leverage data augmentation to reduce reliance on training data collection and preparation and to build more accurate machine learning models faster. We'll often want to increase the size and diversity of our training data split through data augmentation. It involves using the existing samples to generate synthetic, yet realistic, examples.

Split the dataset. We want to split our dataset first because many augmentation techniques will cause a form of data leak if we allow the generated samples to be placed across different data splits.

For example, some augmentation involves generating synonyms for certain key tokens in a sentence. If we allow the generated sentences from the same origin sentence to go into different splits, we could be potentially leaking samples with nearly identical embedding representations across our different splits.

Augment the training split. We want to apply data augmentation on only the training set because our validation and testing splits should be used to provide an accurate estimate on actual data points.

Inspect and validate. It's useless to augment just for the same of increasing our training sample size if the augmented data samples are not probable inputs that our model could encounter in production.

Deep convolutional neural networks have performed remarkably well on many Computer Vision tasks. However, these networks are heavily reliant on big data to avoid overfitting. Overfitting refers to the phenomenon when a network learns a function with very high variance such as to perfectly model the training data. Unfortunately, many application domains do not have access to big data, such as medical image analysis. This survey focuses on Data Augmentation, a data-space solution to the problem of limited data. Data Augmentation encompasses a suite of techniques that enhance the size and quality of training datasets such that better Deep Learning models can be built using them. The image augmentation algorithms discussed in this

survey include geometric transformations, color space augmentations, kernel filters, mixing images, random erasing, feature space augmentation, adversarial training, generative adversarial networks, neural style transfer, and meta-learning. The application of augmentation methods based on GANs are heavily covered in this survey. In addition to augmentation techniques, this paper will briefly discuss other characteristics of Data Augmentation such as test-time augmentation, resolution impact, final dataset size, and curriculum learning.

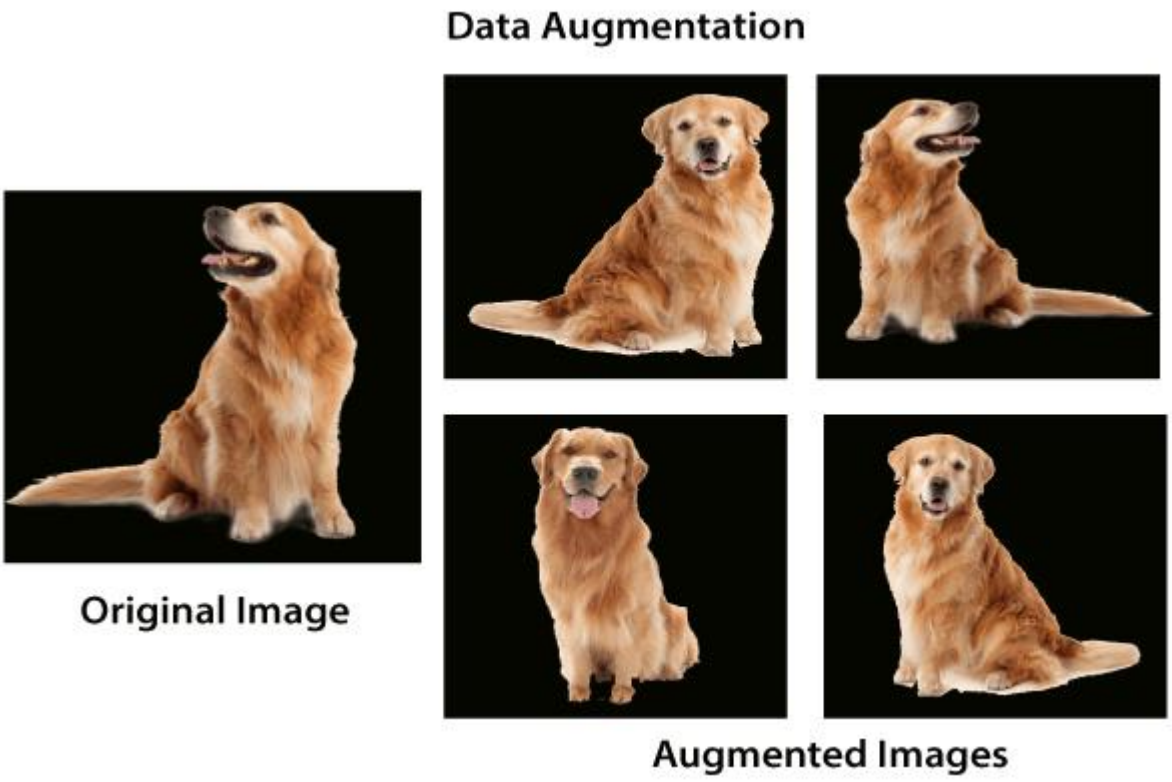


Figure 9

2.4 *Binary Classification:*

Binary classification is a form of classification — the process of predicting categorical variables — where the output is restricted to two classes. Deep learning can be used for binary classification, too. In fact, building a neural network that acts as a binary classifier is little different than building one that acts as a regressor. In this post, you'll learn how to use Keras to build binary classifiers. My next post will describe how to create deep-learning models that perform multiclass classification.

Classification is the process of dividing a set of data into distinct classes. It may be applied to both organized and unstructured data. Predicting the class of data points is the first step in the procedure. Target, label, and categories are common terms for the classes.

Approximating the mapping function from discrete input variables to discrete output variables is the problem of classification predictive modelling. The basic objective is to figure out which category or class the new data belongs in.

This is what we'll discuss a bit more in-depth here. Classification problems with two class labels are referred to as binary classification. In most binary classification problems, one class represents the normal condition and the other represents the aberrant condition.

For example, the normal class label would be that a patient has the disease, and the abnormal class label would be that they do not, or vice-versa.

As is with every other type of classification, it is only as good as the binary classification dataset that it has – or, in other words, the more training and data it has, the better it is.

There are quite a few different algorithms used in binary classification. The two that are designed with only binary classification in mind (meaning they do not support more than two class labels) are Logistic Regression and Support Vector Machines. A few other algorithms are: Nearest Neighbours, Decision Trees, and Naive Bayes.

Binary classification problems can be solved by a variety of machine learning algorithms ranging from Naive Bayes to deep learning networks. Which solution performs best in terms of runtime and accuracy depends on the data volume (number of samples and features) and data quality (outliers, imbalanced data).

The actual output of many binary classification algorithms is a prediction score. The score indicates the system's certainty that the given observation belongs to the positive class. To make the decision about whether the observation should be classified as positive or negative, as a consumer of this score, you will interpret the score by picking a classification threshold (cut-off) and compare the score against it. Any observations with scores higher than the threshold are then predicted as the positive class and scores lower than the threshold are predicted as the negative class.

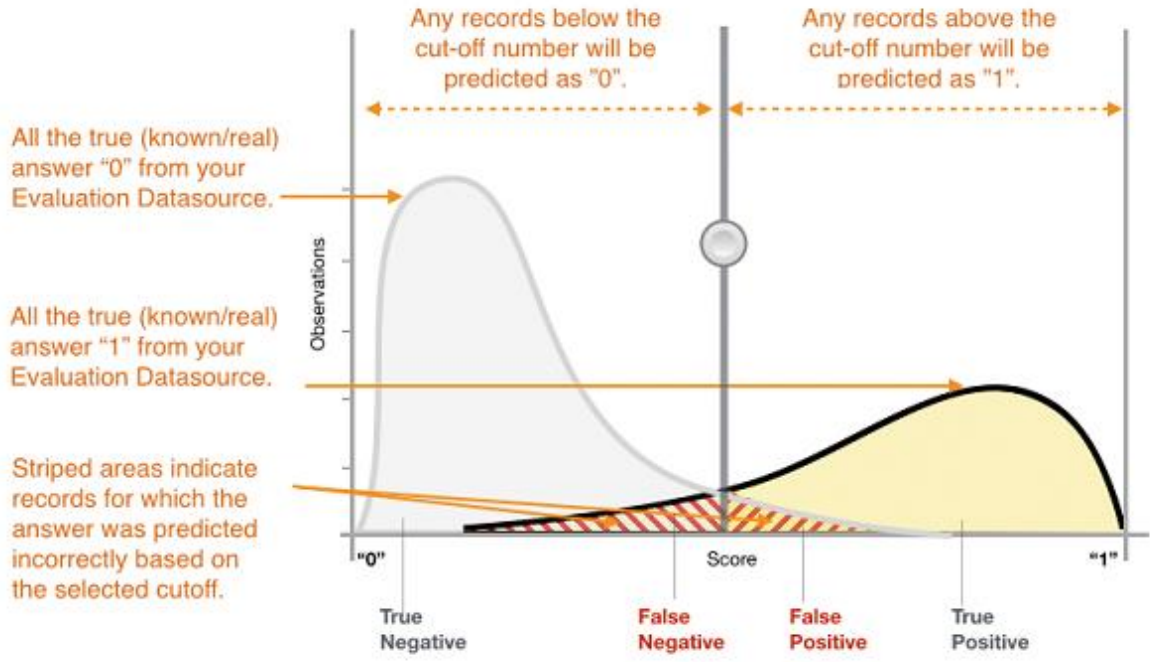


Figure 10

Binary classification means there are two classes to work with that relate to one another as true and false. Imagine you have a huge lug box in front of you with yellow and red tomatoes. But, your fancy Italian pasta recipe says that you only need the red ones.

2.5 *Multilabel Classification:*

Multi-label classification involves predicting zero or more class labels.

Unlike normal classification tasks where class labels are mutually exclusive, multi-label classification requires specialized machine learning algorithms that support predicting multiple mutually non-exclusive classes or "labels."

Deep learning neural networks are an example of an algorithm that natively supports multi-label classification problems. Neural network models for multi-label classification tasks can be easily defined and evaluated using the Keras deep learning library.

Most classification problems associate a single class to each example or instance. However, there are many classification tasks where each instance can be associated with one or more classes. This group of problems represents an area known as multi-label classification. One typical example of multi-label classification problems is the classification of documents, where each document can be assigned to more than one class.'

Classification problems with two or more class labels, where one or more class labels may be anticipated for each case, are referred to as multi-label classification. It differs from binary and multi-class classification, which predict a single class label for each case.

Some classification tasks require predicting more than one class label. This means that class labels or class membership are not mutually exclusive. These tasks are referred to as multiple label classification, or multi-label classification for short.

In multi-label classification, zero or more labels are required as output for each input sample, and the outputs are required simultaneously. The assumption is that the output labels are a function of the inputs.

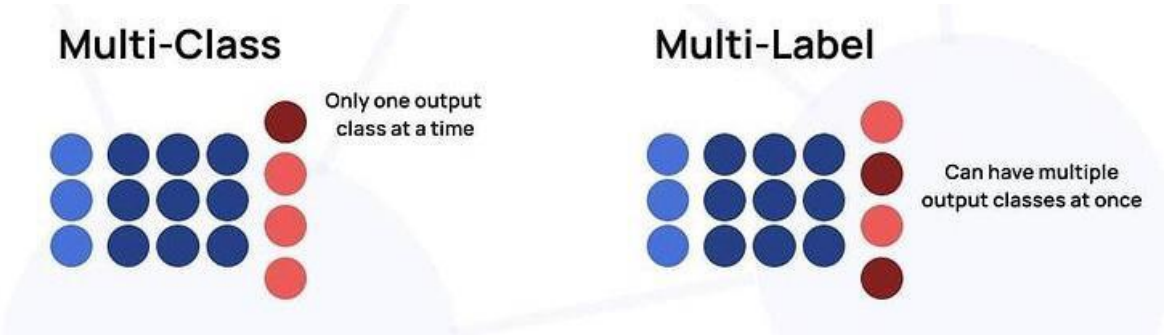


Figure 11

To understand multi-label classification, firstly we will understand what is meant by multi-label, and find the difference between multi-label and binary-label.

Multi-label vs. single-label is the matter of how many classes an object or example can belong to. In neural networks, when single-label is required, we use a single soft-max layer as the last layer, learning a single probability distribution that ranges over all classes. In the case where multi-label classification is needed, we use multiple patterns on the last layer and thus learn a separate distribution for each class.

In certain problems, each input can have multiple, or even none, of the designated output classes. In these cases, we go for the multi-label classification problem approach.
Therefore, for a given dataset, any of the samples that come from the dataset takes more than one label out of the number of available classes. For example, the given movies and how they're classified into different genre.

movie	Adventure	Comedy	Fantasy	Crime	children
Jumanji (1995)	1	0	1	0	1
Puccini for Beginners (2006)	0	1	0	0	0
How the Grinch Stole Christmas! (1966)	0	1	1	0	1

Table 2

2.6 *Convolutional Neural Networks:*

A convolutional neural network is a feed-forward neural network that is generally used to analyze visual images by processing data with grid-like topology. It's also known as a ConvNet. A convolutional neural network is used to detect and classify objects in an image.

In CNN, every image is represented in the form of an array of pixel values.

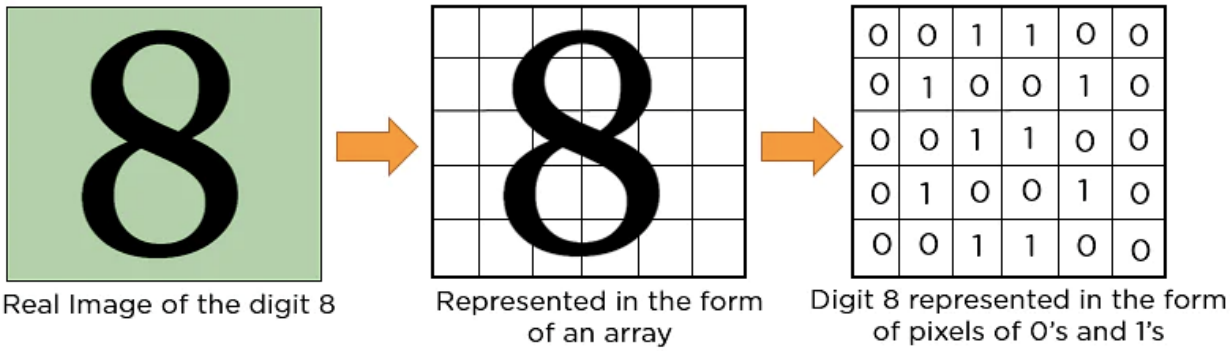


Figure 12

The convolution operation forms the basis of any convolutional neural network. Let's understand the convolution operation using two matrices, a and b, of 1 dimension.

a = [5,3,7,5,9,7]

b = [1,2,3]

In convolution operation, the arrays are multiplied element-wise, and the product is summed to create a new array, which represents a*b.

The first three elements of the matrix a are multiplied with the elements of matrix b. The product is summed to get the result.

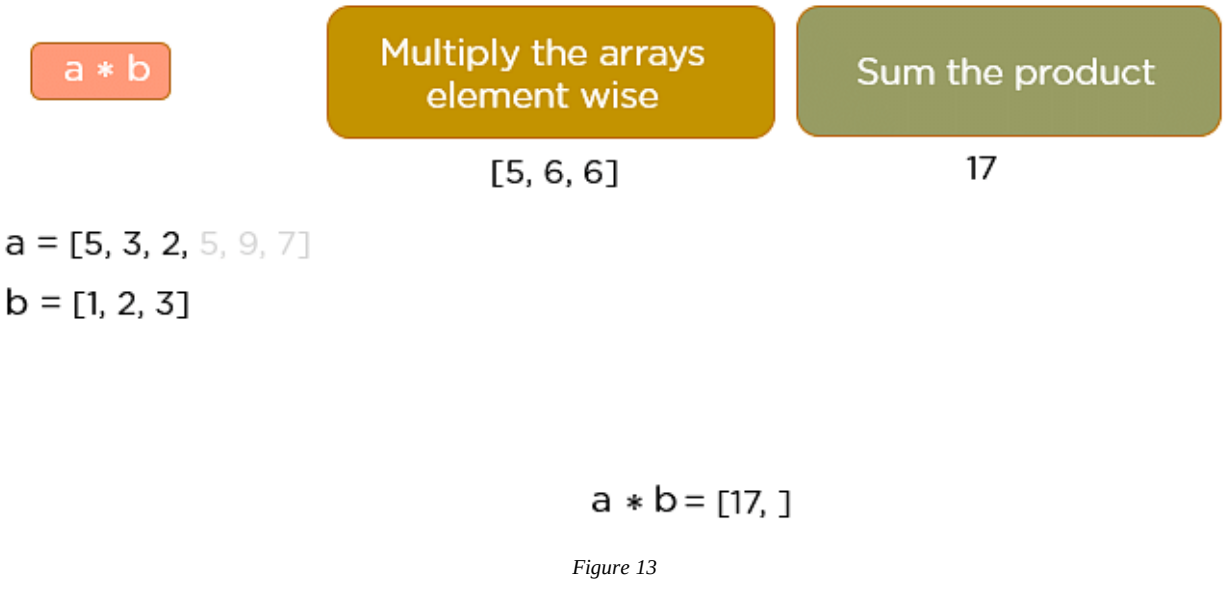


Figure 13

The next three elements from the matrix a are multiplied by the elements in matrix b, and the product is summed up

This process continues until the convolution operation is complete.

2.7 How does CNN recognize images

Consider the following images:

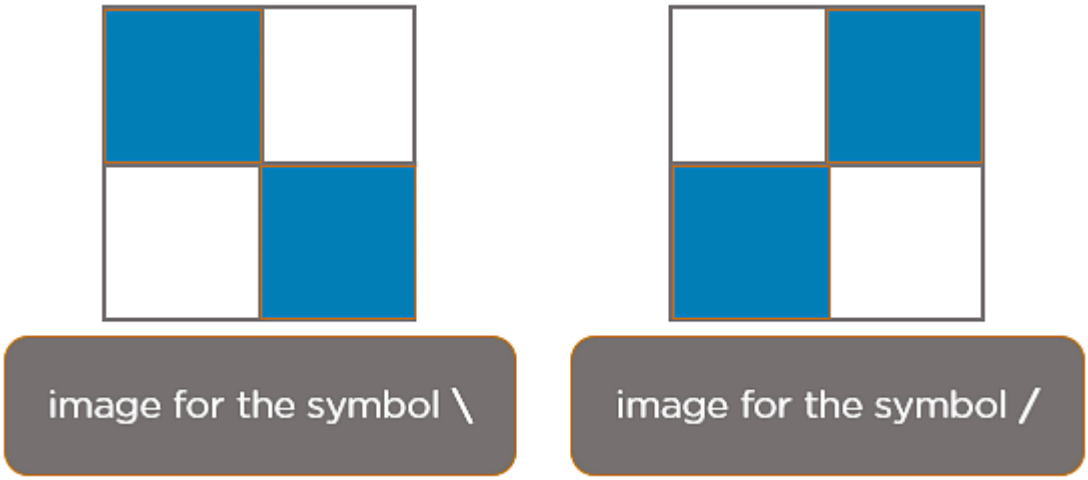


Figure 14

The boxes that are colored represent a pixel value of 1, and 0 if not colored. When you press backslash (\), the below image gets processed.

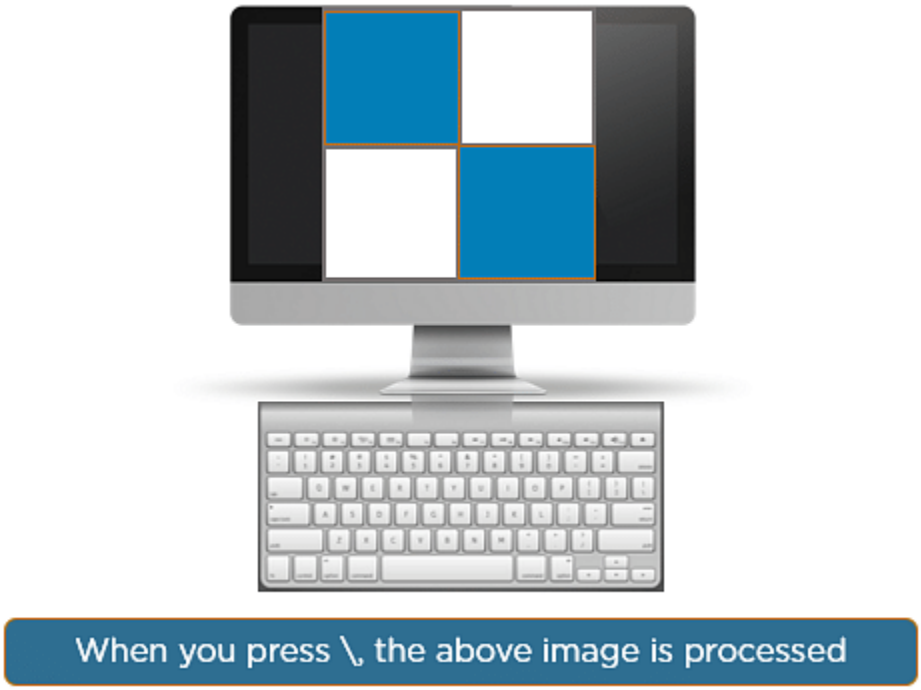


Figure 15

When you press forward-slash (/), the below image is processed

Here is another example to depict how CNN recognizes an image:

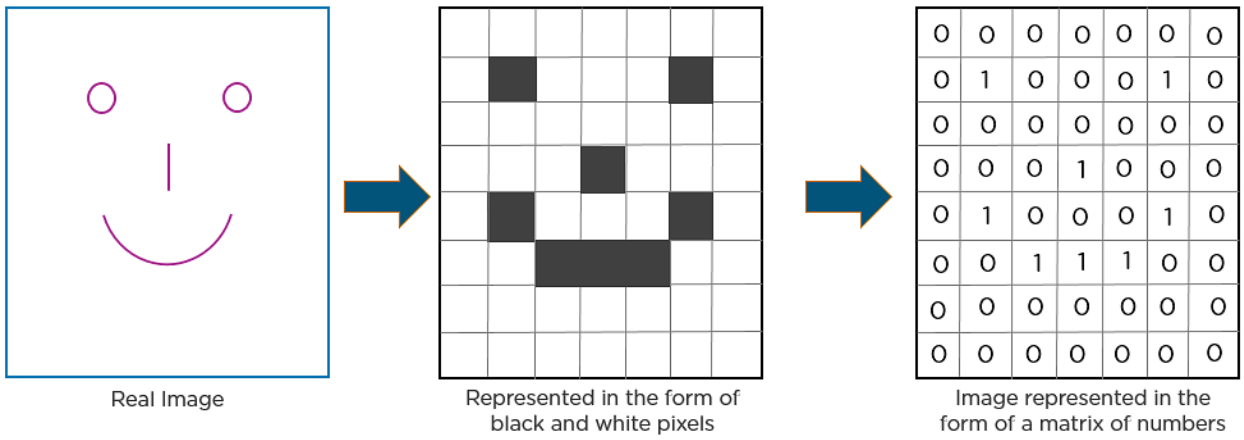


Figure 16

As you can see from the above diagram, only those values are lit that have a value of 1.

A convolution neural network has multiple hidden layers that help in extracting information from an image. The three important layers in CNN are:

1. Convolution layer

This is the first step in the process of extracting valuable features from an image. A convolution layer has several filters that perform the convolution operation. Every image is considered as a matrix of pixel values.

2. ReLU layer

ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is to move them to a ReLU layer.

ReLU performs an element-wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map.

3. Pooling layer

Pooling is a down-sampling operation that reduces the dimensionality of the feature map. The rectified feature map now goes through a pooling layer to generate a pooled feature map.

2.8 *ResNet 101:*

ResNet, short for Residual Networks, is a classic neural network used as a backbone for many computer vision tasks. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training, very deep neural networks were difficult due to the problem of vanishing gradients. It is a convolutional neural network that is 101 layers deep. One can load a pre-trained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224. One can use classify to classify new images using the ResNet-101 model. Replacing VGG-16 layers in Faster R-CNN with ResNet-101, they observed a relative improvement of 28%. Basically, this solves the problem when a deeper network starts converging; a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated and then degrades rapidly. Whenever one wants to increase the accuracy of a classification model, the easiest way to do so is to increase the number of layers. Hence, we change the pre-trained model from ResNet50 to ResNet101. This increases our accuracy from 91.3% to 91.7%.

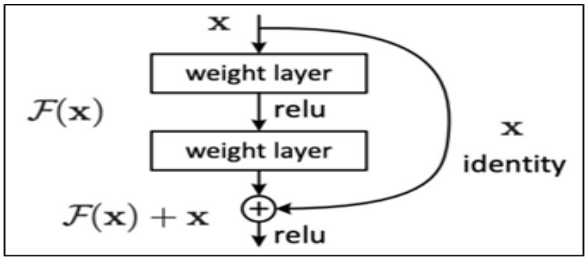


Figure 17

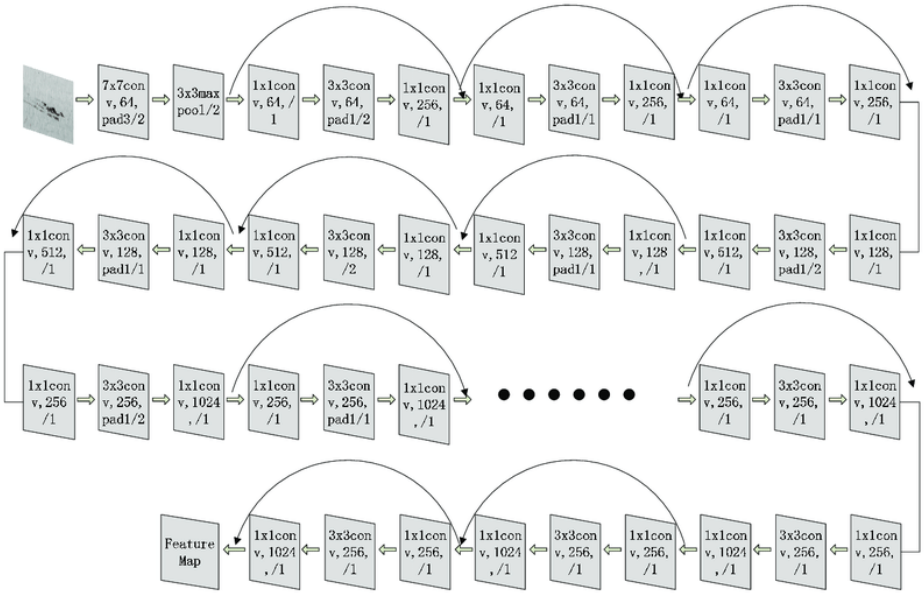

7x7con v, 64, pad3/2
3x3max pool/2
1x1con v, 64, / 1
3x3con v, 64, pad1/2
1x1con v, 256, /1
1x1con v, 64, /1
3x3con v, 64, pad1/1
1x1con v, 256, /1
1x1con v, 64, /1
3x3con v, 64, pad1/1
1x1con v, 256, /1
1x1con v, 512, /1
3x3con v, 128, pad1/1
1x1con v, 128, /1
1x1con v, 512, /1
3x3con v, 128, /2
1x1con v, 128, /1
1x1con v, 512 /1
3x3con v, 128, pad1/1
1x1con v, 128 , /1
1x1con v, 512, /1
3x3con v, 128, pad1/2
1x1con v, 128, /1
1x1con v, 256 /1
3x3con v, 256, pad1/2
1x1con v, 1024 , /1
1x1con v, 256, /1
3x3con v, 256, pad1/1
1x1con v, 1024 , /1
1x1con v, 256, /1
3x3con v, 256, /1
1x1con v, 1024 , /1
Feature Map
1x1con v, 1024 , /1
3x3con v, 256, /1
1x1con v, 256, /1
1x1con v, 1024 , /1
3x3con v, 256, /1
1x1con v, 256, /1
1x1con v, 1024 , /1
3x3con v, 256, /1
1x1con v, 256, /1

Figure 18

2.9 *DenseNet121*

In a traditional feed-forward Convolutional Neural Network (CNN), each convolutional layer except the first one (which takes in the input), receives the output of the previous convolutional layer and produces an output feature map that is then passed on to the next convolutional layer. Therefore, for 'L' layers, there are 'L' direct connections; one between each layer and the next layer. However, as the number of layers in the CNN increases, i.e. as they get deeper, the 'vanishing gradient' problem arises. This means that as the path for information from the input to the output layers increases, it can cause certain information to 'vanish' or get lost which reduces the ability of the network to train effectively. DenseNets resolve this problem by modifying the standard CNN architecture and simplifying the connectivity pattern between layers. In a DenseNet architecture, each layer is connected directly with every other layer, hence the name Densely Connected Convolutional Network. For 'L' layers, there are L(L+1)/2 direct connections. DenseNet-121 has the following layers:

1 7x7 Convolution

58 3x3 Convolution

61 1x1 Convolution

4 AvgPool

1 Fully Connected Layer

In short, DenseNet-121 has 120 Convolutions and 4 AvgPool.

All layers i.e., those within the same dense block and transition layers, spread their weights over multiple inputs which allows deeper layers to use features extracted early on. DenseNet require fewer parameters and allow feature reuse, they result in more compact models and have achieved state-of-the-art performances and better results across competitive datasets, as compared to their standard CNN or ResNet counterparts. Thus, the next architecture we try is the DenseNet121 architecture. In DenseNet, each layer receives collective knowledge from all preceding layers. For this, the output of a convolution is concatenated with the input. This concatenated feature-map is passed to the next layer. Since this increases the number of parameters, we reduce the batch size by half. With the DenseNet, we get the accuracy of 91.8%.

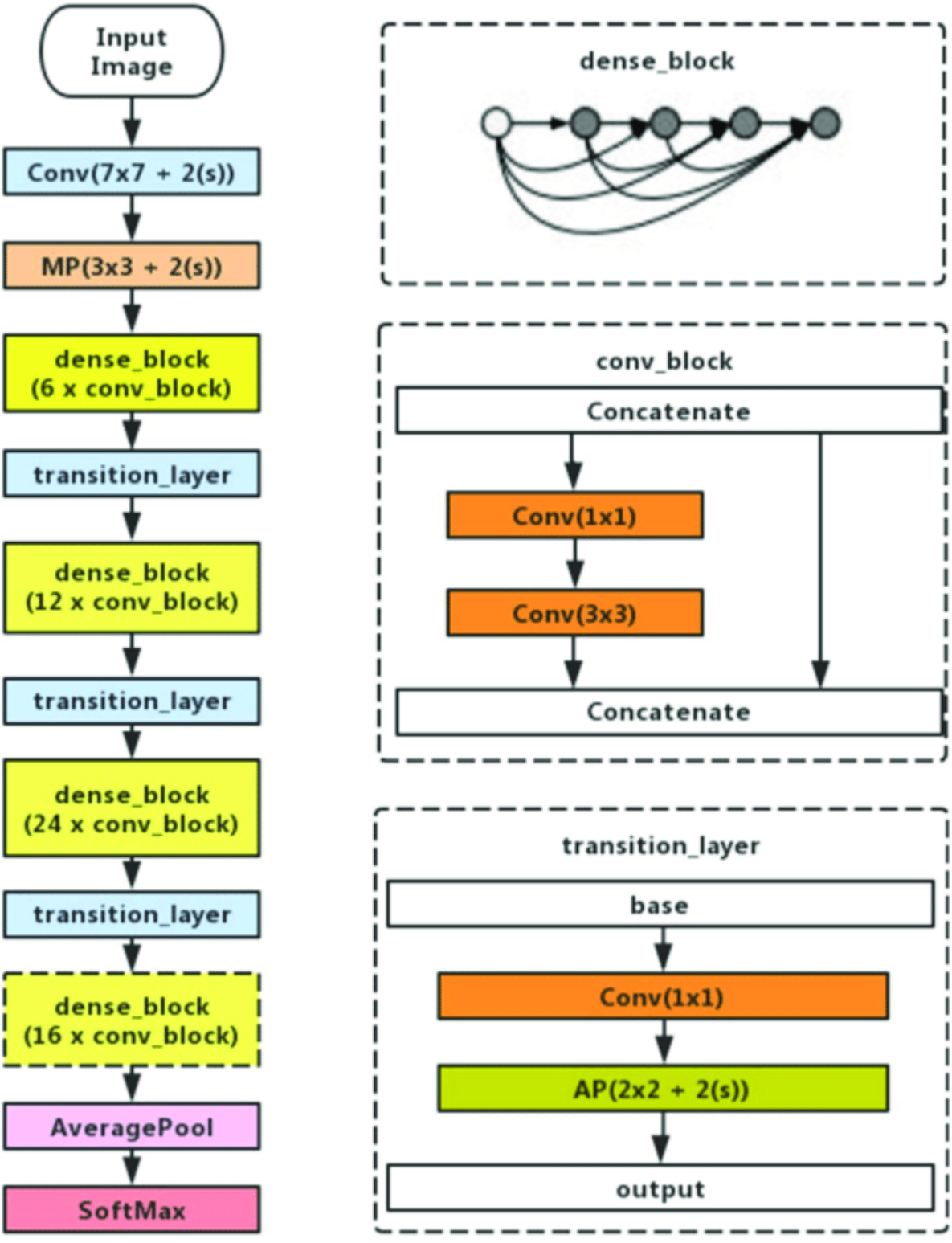

Input Image
Conv(7x7 + 2(s))
MP(3x3 + 2(s))
dense_block (6 x conv_block)
transition_layer
dense_block (12 x conv_block)
transition_layer
dense_block (24 x conv_block)
transition_layer
dense_block (16 x conv_block)
AveragePool
SoftMax
dense_block
conv_block
Concatenate
Conv(1x1)
Conv(3x3)
Concatenate
transition_layer
base
Conv(1x1)
AP(2x2 + 2(s))
output

Figure 19

2.10 *AlexNet:*

A major breakthrough came when Alex Krizhevsky and Ilya Sutskever implemented a deep CNN that could run on GPU hardware. They realized that the computational bottlenecks in CNNs, convolutions and matrix multiplications, are all operations that could be parallelized in hardware. Using two NVIDIA GTX 580s with 3GB of memory, they implemented fast convolutions. The code cuda-convnet was good enough that for several years it was the industry standard and powered the first couple years of the deep learning boom. AlexNet, which employed an 8-layer CNN, won the ImageNet Large Scale Visual Recognition Challenge 2012 by a phenomenally large margin. This network showed, for the first time, that the features obtained by learning can transcend manually-designed features, breaking the previous paradigm in computer vision. The final architecture we try is the AlexNet. This is another famous architecture for image classification. It won the ImageNet competition in 2012. It uses overlap pooling to reduce the size of the network and trains quite fast. Although the training is much faster than the other architectures we used, the accuracy was not better. It came to 89%.The architecture consists of eight layers: five convolutional layers and three fully-connected layers. But this isn't what makes AlexNet special; these are some of the features used that are new approaches to convolutional neural networks:

- ReLU Nonlinearity. AlexNet uses Rectified Linear Units (ReLU) instead of the tanh function, which was standard at the time. ReLU's advantage is in training time; a CNN using ReLU was able to reach a 25% error on the CIFAR-10 dataset six times faster than a CNN using tanh.

- Multiple GPUs. Back in the day, GPUs were still rolling around with 3 gigabytes of memory (nowadays those kinds of memory would be rookie numbers). This was especially bad because the training set had 1.2 million images. AlexNet allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU. Not only does this mean that a bigger model can be trained, but it also cuts down on the training time.

- Overlapping Pooling. CNNs traditionally "pool" outputs of neighboring groups of neurons with no overlapping. However, when the authors introduced overlap, they saw a reduction in error by about 0.5% and found that models with overlapping pooling generally find it harder to overfit

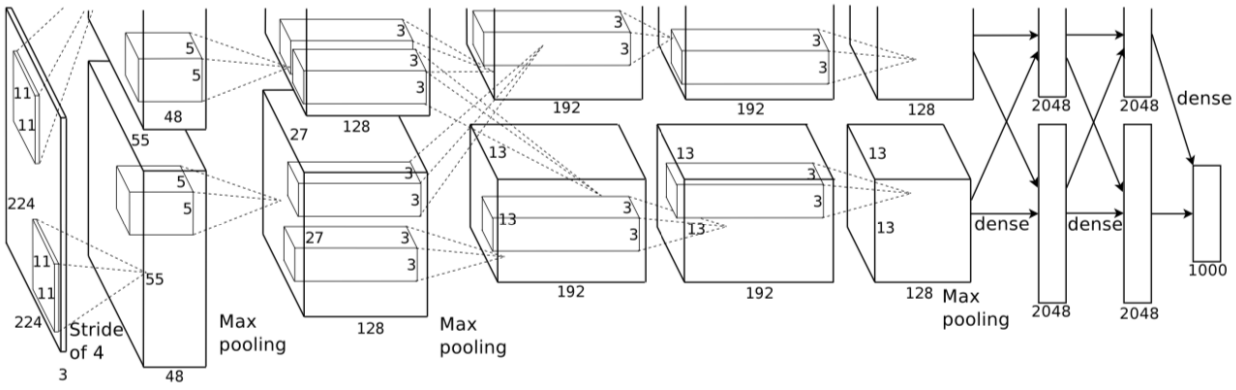


Figure 20

The Overfitting Problem. AlexNet had 60 million parameters, a major issue in terms of overfitting. Two methods were employed to reduce overfitting:

- Data Augmentation. The authors used label-preserving transformation to make their data more varied. Specifically, they generated image translations and horizontal reflections, which increased the training set by a factor of 2048. They also performed Principle Component Analysis (PCA) on the RGB pixel values to change the intensities of RGB channels, which reduced the top-1 error rate by more than 1%.

- Dropout. This technique consists of "turning off" neurons with a predetermined probability (e.g. 50%). This means that every iteration uses a different sample of the model's parameters, which forces each neuron to have more robust features that can be used with other random neurons. However, dropout also increases the training time needed for the model's convergence.

AlexNet is an incredibly powerful model capable of achieving high accuracies on very challenging datasets. However, removing any of the convolutional layers will drastically degrade AlexNet's performance. AlexNet is a leading architecture for any object-detection task and may have huge applications in the computer vision sector of artificial intelligence problems. In the future, AlexNet may be adopted more than CNNs for image tasks.

As a milestone in making deep learning more widely-applicable, AlexNet can also be credited with bringing deep learning to adjacent fields such as natural language processing and medical image analysis.

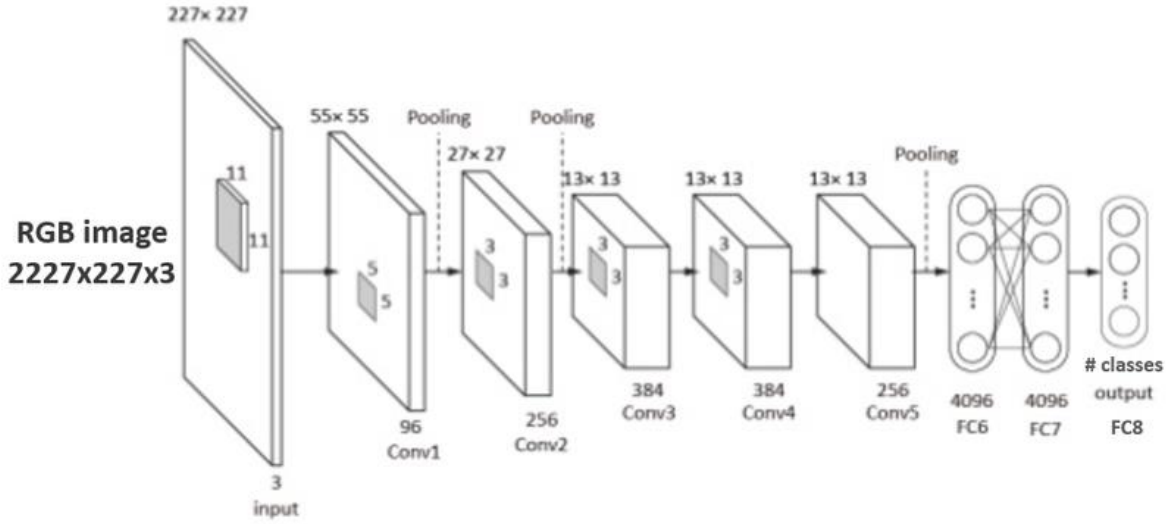

227× 227
RGB image
2227x227x3
11
11
3
input
55× 55
5
5
96
Conv1
Pooling
27× 27
3
3
256
Conv2
Pooling
13× 13
3
3
384
Conv3
13× 13
3
3
384
Conv4
13× 13
256
Conv5
Pooling
4096
FC6
4096
FC7
classes
output
FC8

Figure 21

2.11 *Weight Decay:*

Weight Decay provides an approach to reduce the overfitting of a deep learning neural network model on the training data and improve the performance of the model on new data, such as the holdout test set. It prevents our model from getting too complex by penalizing complexity. It does so by adding the sum of squares of all parameters to the loss function. However, this can make the loss so huge that the best model would set all the parameters to 0. To prevent this from happening, we multiply the sum of squares with another small number. This number is called weight decay (wd).

$$loss = crossen(yy) + wd * \sum parameter^2$$

When we update weights using gradient descent we do the following:

w(t) = w(t-1) – lr*dLoss / dw

Now since our loss function has 2 terms in it, the derivative of the 2nd term w.r.t. w would be

d(wd * w^2) / dw = 2 * wd * w (similar to d(x^2)/dx = 2x)

We use weight decay value = 0.01. The reason to choose this value is because if you have too much weight decay, then no matter how much you train, the model never quite fits well enough whereas if you have too little weight decay, you can still train well, you just have to stop a little bit early. This helps the model learn much better.

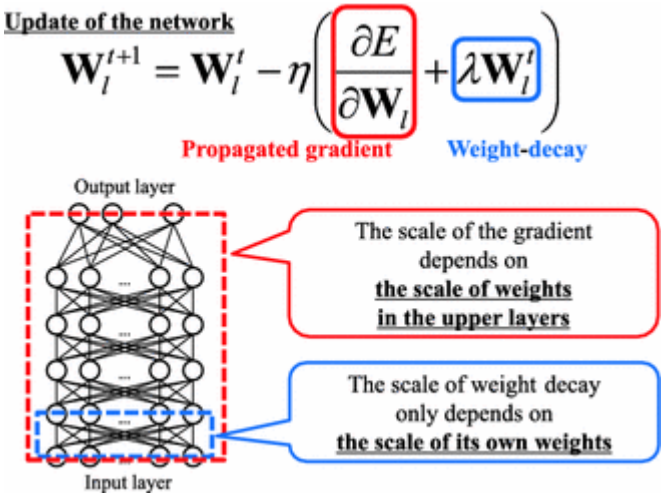


Figure 22

2.12 *Mixed Precision Training*

In neural networks, inputs, parameters and activations are stored using 32 bits. This gives us a high amount of precision. However, these high precision computations require more time and memory. Hence, using 32-bit precision for all our calculations, we perform some of them in 16 bits. By leveraging the half-precision floating-point format (FP16) well supported by recent GPUs, mixed precision training (MPT) enables us to train larger models under the same or even smaller budget. Weight updates need to be as precise as possible hence, we haven't reduced the bits for weight updates. Thus Mixed precision training offers significant computational speedup by performing operations in half-precision format, while storing minimal information in single-precision to retain as much information as possible in critical parts of the network. Since the introduction of Tensor Cores in the Volta and Turing architectures, significant training speedups are experienced by switching to mixed precision -- up to 3x overall speedup on the most arithmetically intense model architectures. Using mixed precision training requires two steps:

1. Porting the model to use the FP16 data type where appropriate.
2. Adding loss scaling to preserve small gradient values.

In theory, mixed precision training should reduce the training time by half. However, in practice we see only a slight decrease in the training time. But, in our implementation, the time did not reduce much.

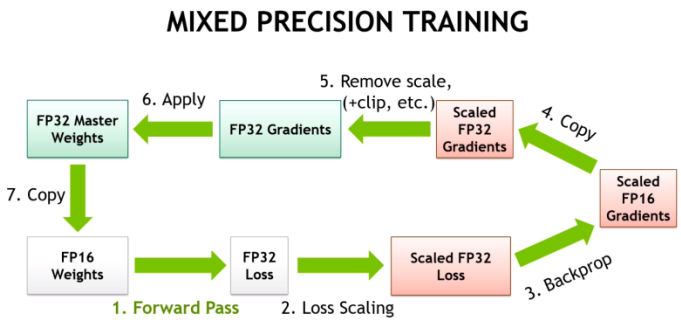


Figure 23

2.13 *Progressive Image Resizing:*

It is the technique to sequentially resize all the images while training the CNNs on smaller to bigger image sizes. A great way to use this technique is to train a model with smaller image size say 64x64, then use the weights of this model to train another model on images of size 128x128 and so on. Each larger-scale model incorporates the previous smaller-scale model layers and weights in its architecture. Since neural networks receive inputs of the same size, all images need to be resized to a fixed size before inputting them to the CNN. The larger the fixed size, the less shrinking required. Less shrinking means less deformation of features and patterns inside the image. This will mitigate the classification accuracy degradation due to deformations. However, large images not only occupy more space in the memory but also result in a larger neural network. Thus, increasing both the space and time complexity. It is obvious now that choosing this fixed size for images is a matter of tradeoff between computational efficiency and accuracy.

Images larger than the fixed size (in one dimension or both) could be resized down to the desired fixed size using two approaches: cropping their border pixels or scaling them down using interpolation. Both approaches are lossy. While cropping poses the risk of missing the features or patterns that appear in border areas, scaling poses the risk of deforming features or patterns across the image. Since deforming patterns is less risky than losing them, scaling is the reasonable choice to resize larger images down to the desired fixed size. Resizing smaller images up to the fixed size is the focus of this study. Zero-padding is proposed for this purpose and compared with the conventional approach of scaling images up (zooming in) using interpolation.

2.14 *Loss Function - Categorical cross entropy loss:*

Cross entropy loss function is an optimization function which is used in case of training a classification model which classifies the data by predicting the probability of whether the data belongs to one class or the other class. One of the examples where Cross entropy loss function is used is Logistic Regression.

Keras – Categorical Cross Entropy Loss Function

October 28, 2020 by Ajitesh Kumar · Leave a comment

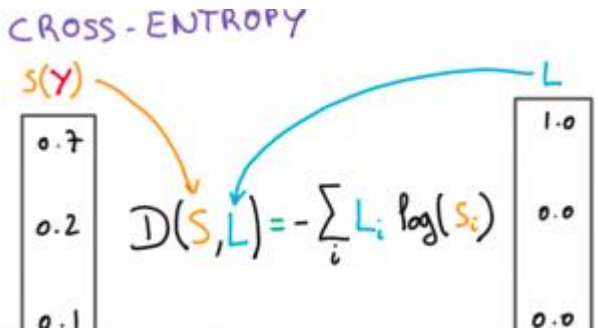


Figure 24

In this post, you will learn about when to use **categorical cross entropy loss function** when training **neural network** using Python **Keras**. Generally speaking, the loss function is used to compute the quantity that the model should seek to minimize during training. For regression models, the commonly used loss function used is mean squared error function while for classification models predicting the probability, the loss function most commonly used is cross entropy. In this post, you will learn about different types of cross entropy loss function which is used to train the Keras neural network model.

Cross entropy loss function is an optimization function which is used in case of training a classification model which classifies the data by predicting the probability of whether the data belongs to one class or the other class. One of the examples where Cross entropy loss function is used is Logistic Regression. Check my post on the related topic – Cross entropy loss function explained with Python examples.

When fitting a neural network for classification, Keras provide the following three different types of cross entropy loss function:

- **Binary cross entropy**: Used as a loss function for binary classification model. The binary cross entropy function computes the cross-entropy loss between true labels and predicted labels.
- **Categorical cross entropy**: Used as a loss function for multi-class classification model where there are two or more output labels. The output label is assigned one-hot category encoding value in form of 0s and 1. The output label, if present in integer form, is converted into categorical encoding using keras. Utils to categorical method.
- **Sparse categorical cross entropy**: Used as a loss function for multi-class classification model where the output label is assigned integer value (0, 1, 2, 3…). This loss function is mathematically same as the categorical cross entropy. It just has a different interface.

Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. Cross-entropy and log loss are slightly different depending on context, but in machine learning when calculating error rates between 0 and 1 they resolve to the same thing.

Cross-entropy is a measure of the difference between two probability distributions for a given random variable or set of events.

You might recall that information quantifies the number of bits required to encode and transmit an event. Lower probability events have more information, higher probability events have less information.

In information theory, we like to describe the "surprise" of an event. An event is more surprising the less likely it is, meaning it contains more information.

Low Probability Event (surprising): More information.
Higher Probability Event (unsurprising): Less information.
Information h(x) can be calculated for an event x, given the probability of the event P(x) as follows:
$h(x) = -\log(P(x))$

2.15 *Benefits of fast.ai:*

Fast.ai library goal is to make the training of deep neural networks as easy as possible, and, at the same time, make it fast and accurate using modern best practices. The library includes out-of-the-box support for computer vision tasks, text, and natural language processing, tabular/structured data classification or regression, and collaborative filtering models.

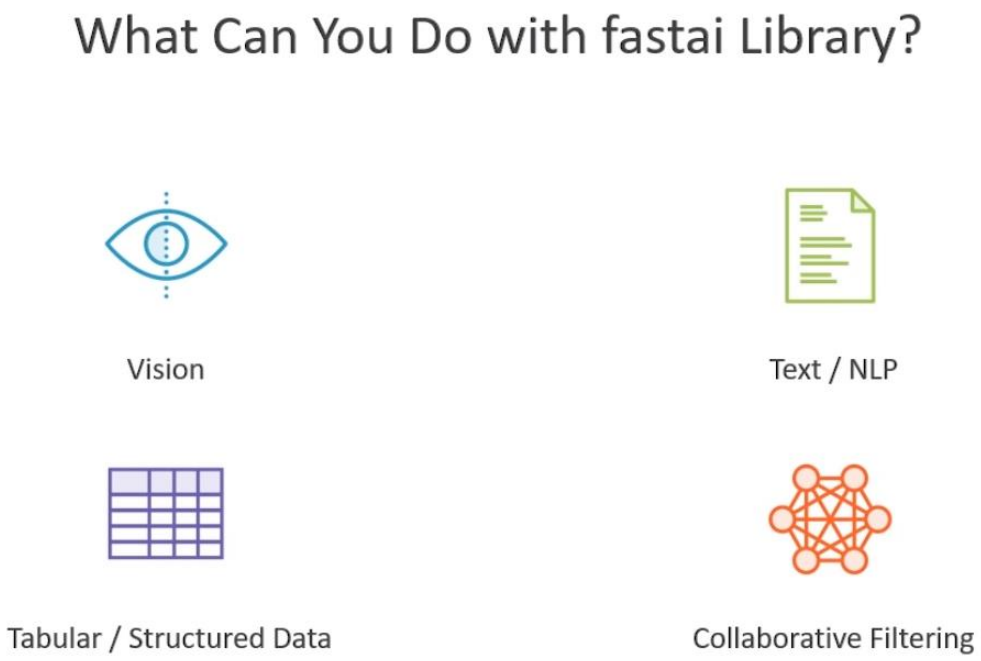


Figure 25

This model regroups the tools the library provides to help us preprocess and group our data, being it an image, a text, or a recurrent tabular dataset, and train a model out of it using the best practices, neural network architectures, and hyperparameters known so far, all built-in and without the need to know them in details. At the same time, if one wants to split the bits, tweak, or customize something, we can do it thanks to its flexible and fine-grained hooks and callbacks. And if that's not enough, we have access to its full-source code and PyTorch statically so we can shape everything according to our needs.

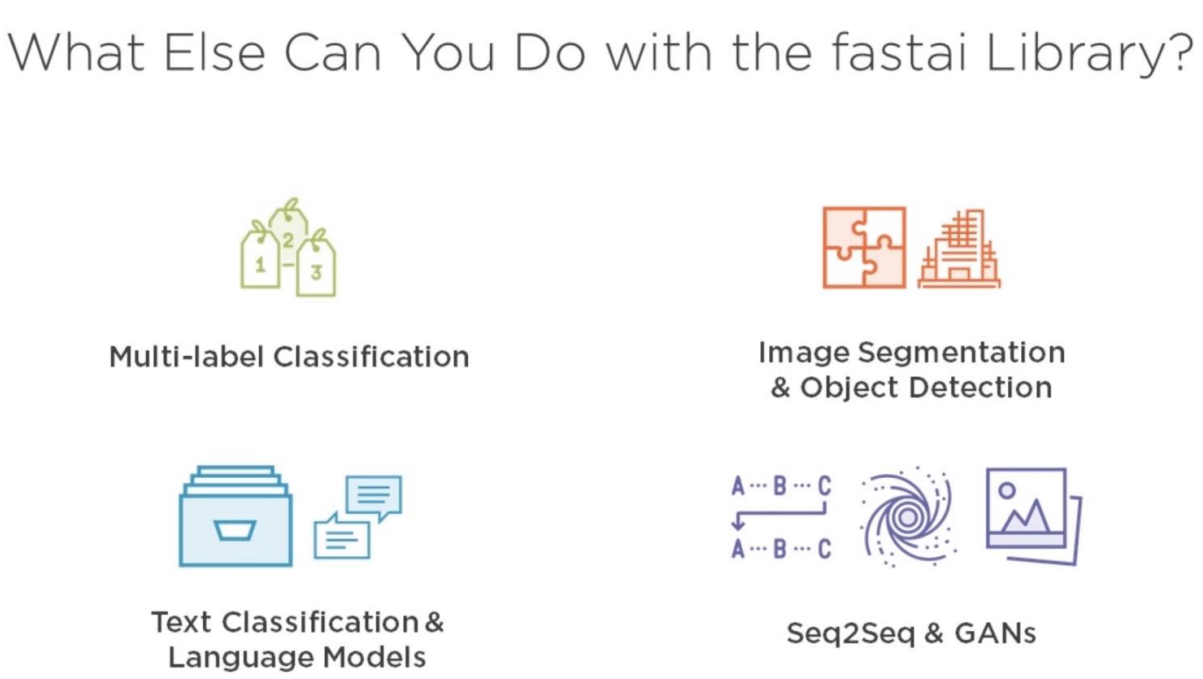


Figure 26

Image segmentation, image regression, or object detection models helpful to classify shapes and objects in pictures of videos.

FastAI delivers flexibility, speed, and ease of use. It offers many features and functionality, which makes developers customize the high-level API without engaging in low-level API parts. One example of this customization is Data Block, which allows you to load the data in detail. As FastAI explains, Data Loader class loads both the training and the validation data classes. Besides, the process of using validation data sets while training the data would make the job easier. Beginners working with this library, therefore, use available functions and start customizing models. The figure shows four fields of applications, including vision, text, tabular, and collaborative filtering, each of which is used for different purposes.

The FastAI library also implements the learning rate finder that provides the best value for the learning rate parameter after a sample training session.

CHAPTER 3 (IMPLEMENTATION)

3.1 Data Cleaning

The data is given in the form of DICOM files. A DICOM file consists of metadata and image data packed into a single file. The information in the metadata consists of various tags. It also removes confidential information about patients to maintain the integrity. We have 194082 images in the dataset. We start by converting the metadata into a data frame since the DICOM format is slow to use. We then view some of the images whichare in the form of slices from top of the brain to the bottom of the brain as shown.

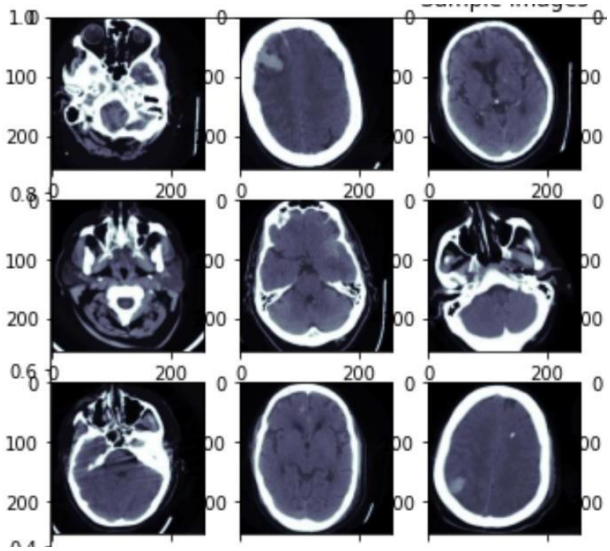


Figure 27: Sample Images

Sometimes the slice can be completely blank (a slice before or after the brain). To remove such slices, we use a column called "img_pct_window". It tells us the percentage of brain pixels in the slice. We remove images with less than 20\% value in that column.

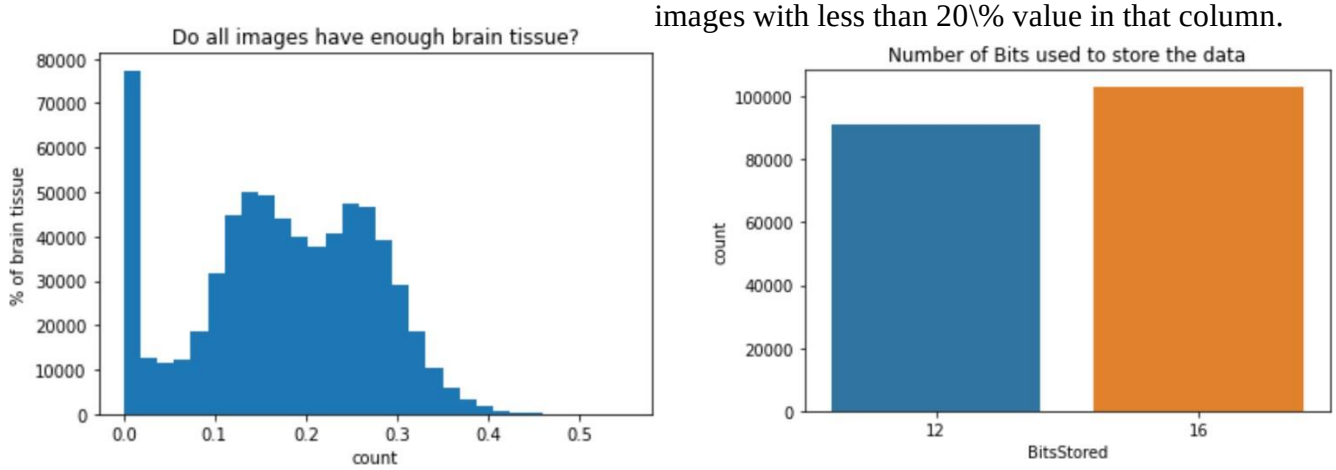


Figure 28

We also fix the column Rescale_Intercept, resize and crop the images to (256,256) and save them as ".jpg".

3.2 *Data Exploration*

We look at the distribution of the data.

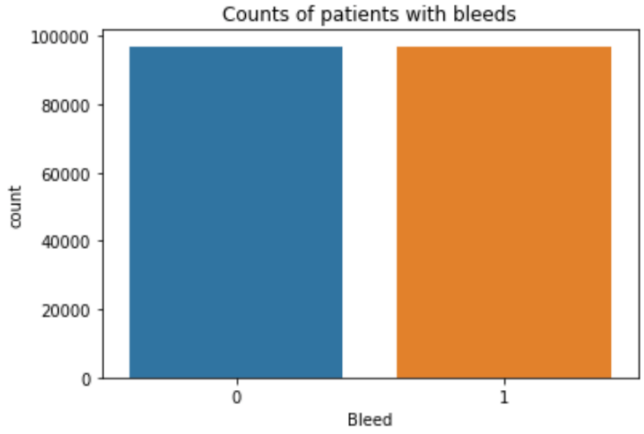


Figure 29

We see that we have a pretty balanced dataset when it comes to images with and without bleeds. We also check the counts of thedifferent subtypes of bleeds.

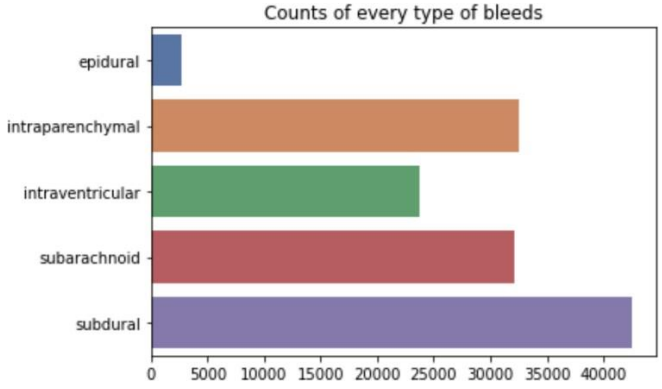


Figure 30

We see that "subdural" is the most common type of bleed present. We then move onto analyzing the metadata. One interesting column in the "bits stored" column which indicatesthe number of bits used to store the data. It has two distinct values: 12 and 16.

This might indicate that the data might have come form two different organizations. It is usually better for deep learning models that the data comes from same distribution but, we willmove forward with this data.

Finally, the images can be viewed in many different windows.

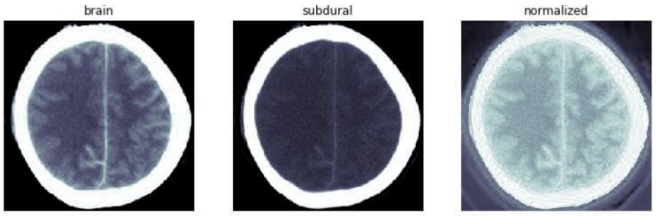


Figure 31

However, this is for the human perception. A neural network accepts floating point data and does not require any windowing.

3.3 *Data Augmentation*

The first step to train a good deep learning model is to have lotsof data. However, this is not always possible. Hence, we do small random transformations on the data that do not change what is inside the image (for the human eye) but changes the pixel values. Models trained with Data Augmentation generalize better.

Below are some of the transformations which can be used for our data.

Flip: The bleed can occur on any side of the brain. Hence, flipping gives the model more cases to consider.

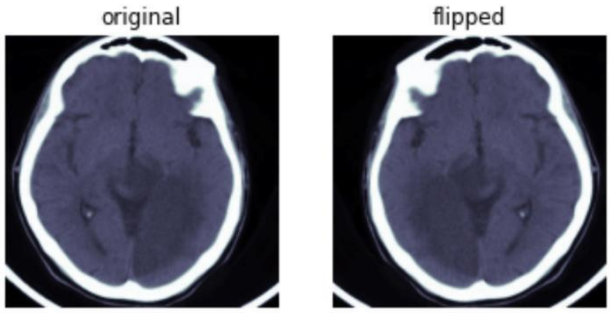


Figure 32

Rotation: The slices do not always appear straight. They might have different orientations. Hence, we randomly rotate few images in range of (-15, 15) degrees.

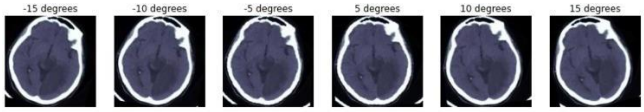


Figure 33

Blur: During an MRI, if the patient moves, the MRI might become blur. Hence, we apply blurring effect on some images.The blur effect is increased sequentially left to right.

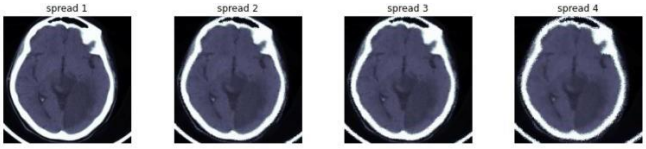


Figure 34

3.4 *Dataset and analysis*

The dataset which was used to develop the model was provided by Radiological Society of North America (RSNA) in collaboration with members of the American Society of Neuroradiology and MD.ai. All the images are in DICOM format and DICOM images contain a blend of header metadata just as basic crude picture clusters for pixel information. In Python, one mainstream library to get to and control DICOM documents is the pydicom. read file () strategy. The associated metadata also includes PatientID, SOPInstanceUID, and other features. The features each DICOM image contains is shown in the Fig. 2 Fig. 3 Fig. 4 Fig. 5 Fig. 6 Fig. 7. Hemorrhage Sub-Types: In this section, we grouped the different types of hemorrhage in a new file and then separated them and found the count for each subtype of hemorrhage images present. Fig. 1. Process Overview. H. Kishan Das Menon and V. Janardhan Materials Today: Proceedings 43 (2021) 3706–3714 3707 Train Dataset: In this section, we look at the train dataset which has two labels 0 and labels 1 where label 0 indicates that the image contains a hemorrhage, and label 1 indicates the number of patients count without hemorrhage. Hemorrhage Sub-Types (DICOM Representation): Here, we exemplate 2 of the subtypes of the hemorrhage. It is also to be noted that the dataset was a compiled record of many CT scans covering a wide range of patients of which the focus of the model was to spot the type of hemorrhage in the patient irrespective of the age and sex, as hemorrhage types are largely clustered or differentiated according to the location of the clot or the damage in the brain. Also this data was generated by the Radiological Society of North America, wherein experts annotated the images as normal or abnormal and that the abnormal ones were further classified into various sub sections. Also, the maximum information concealed about the patients was the patient_ID, to help map the output of the model with that of the patient.

```
In [21]: dataset = pydicom.dcmread(train_dir + "ID_c5c23af94.dcm")
         print(dataset)
```

```
(0008, 0018) SOP Instance UID                   UI: ID_c5c23af94
(0008, 0060) Modality                           CS: 'CT'
(0010, 0020) Patient ID                         LO: 'ID_9630cc49'
(0020, 000d) Study Instance UID                 UI: ID_c5409f3ace
(0020, 000e) Series Instance UID                UI: ID_5db30227b2
(0020, 0010) Study ID                           SH: ''
(0020, 0032) Image Position (Patient)           DS: ['-125.000', '-118.882', '33.678']
(0020, 0037) Image Orientation (Patient)        DS: ['1.000000', '0.000000', '0.000000',
'0.000000', '0.951057', '-0.309017']
(0028, 0002) Samples per Pixel                  US: 1
(0028, 0004) Photometric Interpretation         CS: 'MONOCHROME2'
(0028, 0010) Rows                               US: 512
(0028, 0011) Columns                            US: 512
(0028, 0030) Pixel Spacing                      DS: ['0.488281', '0.488281']
(0028, 0100) Bits Allocated                     US: 16
(0028, 0101) Bits Stored                        US: 16
(0028, 0102) High Bit                           US: 15
(0028, 0103) Pixel Representation               US: 1
(0028, 1050) Window Center                      DS: "40"
(0028, 1051) Window Width                       DS: "150"
(0028, 1052) Rescale Intercept                  DS: "-1024"
(0028, 1053) Rescale Slope                      DS: "1"
(7fe0, 0010) Pixel Data                         OW: Array of 524288 elements
```

Figure 35 : DICOM Image metadata for one image.

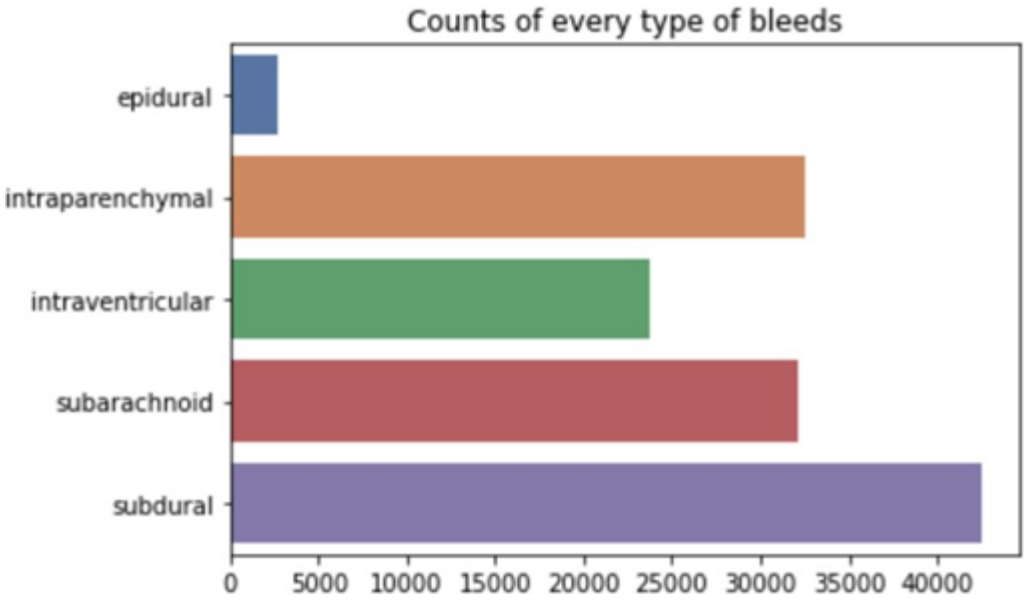


Figure 36 : Image Count (by Sub-Types) – 1 (This image shows the types of hemorrhages that the dataset comprises of and the frequency of the sub-types of hemorrhages).

```
show_batch(trn_imgs)
```

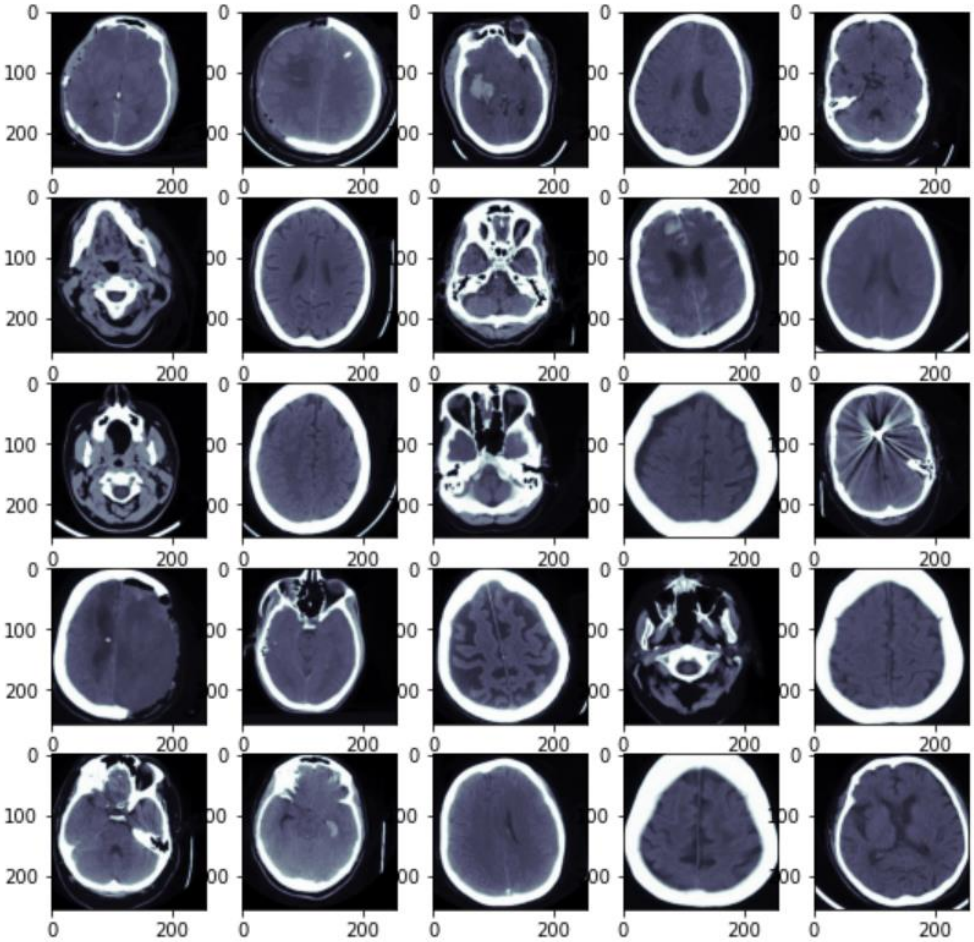


Figure 37: Displaying images and we transpose and print the head to be able to see all the columns.

CHAPTER 4 (RESULTS)

4.1 Binary Classification

Recent research has shown that metadata can prove useful in image classification. Our initial goal was to use a combinationof metadata and images for the task. We try to classify the hemorrhages solely using the metadata to gauge its usefulness.We find that even after using a robust model like Random Forest, we get an accuracy of 50%. Hence, we decide to discardthe metadata and focus on the images itself.

For our baseline model, we start with a ResNet18 model. Transfer learning has shown good results when it comes to image classification. It also reduces training time and resourceconsumption significantly.

We use Leslie Smith's learning rate finder [5] to find a good learning rate.

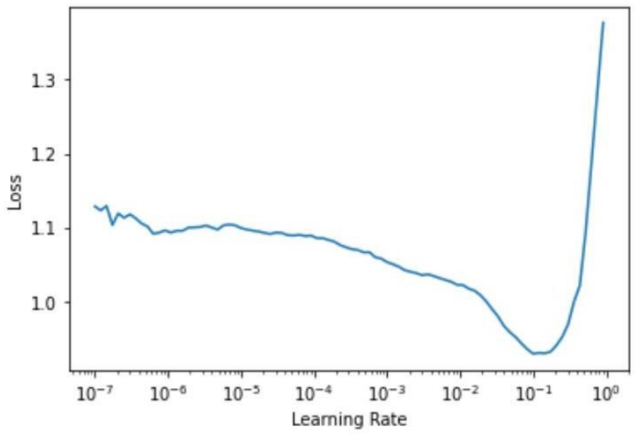


Figure 38

In this technique, we run a mock training loop on our data varying our learning rate from a very low value to a value as high as 10. We then calculate the loss and plot it against the learning rates. The best learning rate would then be the one where the loss decreases the most.

The batch size has been set to 256. We randomly sample 20% of the data as validation set. For hyperparameter tuning, we useLeslie Smith's one cycle training policy [4].

In this method, we initially freeze the pretrained part of the model and only train the new head. However, we do update theBatch Norm layers of the ResNet to maintain mean = 0 and std deviation = 1 of each layer. We have achieved ~89% accuracywith and without pretraining in just 5 epochs.

epoch	train_loss	valid_loss	error_rate	accuracy	time
0	0.268112	0.266647	0.111629	0.888371	11:24
1	0.264794	0.265100	0.110753	0.889247	11:26
2	0.262578	0.265090	0.110959	0.889041	11:20

Table 3

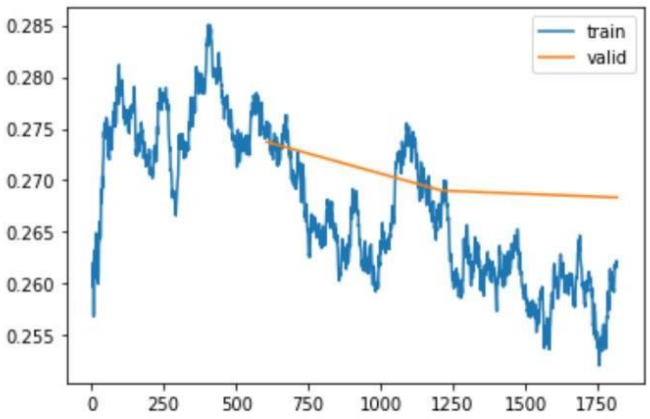


Figure 39

4.2 Multilabel Classification

For Multilabel Classification, we only include the images that have bleeds. We also create a new column in our data frame, which will indicate all the subcategories of bleeds present in an image, separated by semicolons. We use ResNet50 as ourpretrained model. The other hyper parameters are the same as the ones used in Binary Classification except for accuracy. The first step is to decide a threshold. We have used threshold = 0.5 for our implementation. Then, instead of taking the maximum value from our sigmoid outputs, we take all the classes with a value greater than our threshold. This way we get multiple classes per image. We can then compare them with the ground truths and get the accuracy. Our new accuracy formula looks asfollows:

Figure 40

```
def accuracy_multi(inp, targ, thresh=0.5, sigmoid=True):
    "Compute accuracy when `inp` and `targ` are the same size."
    if sigmoid: inp = inp.sigmoid()
    return ((inp>thresh)==targ.bool()).float().mean()
```

This architecture gives us an accuracy of about 91.3%. A ResNet pretrained model includes a number of res blocks which are also known as identity or skip connections. ResNets have been revolutionary for image classification and the reason theywork so well is because they solve the problem of gradient diffusion. A

because they solve the problem of gradient diffusion. A typical ResBlock architecture is shown below:

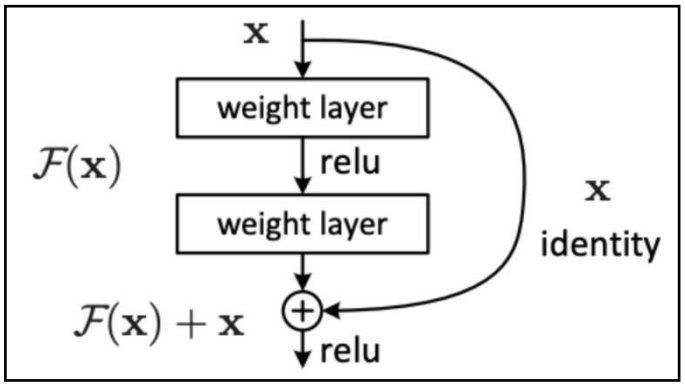
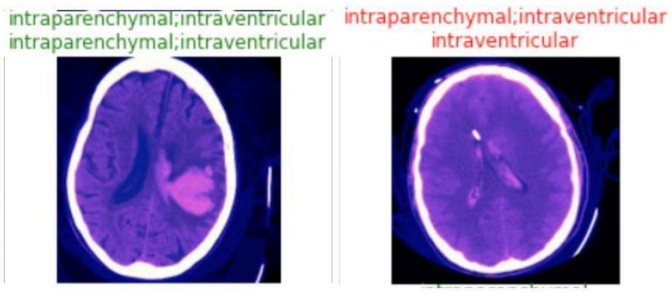


Figure 41: typical ResBlock architecture is shown above

What it does it, for every 2 convolutions, it adds the input to those convolutions to their output. If we consider our convolutions as functions, then the output goes from:



$$c2(c1(x)) \quad => \quad x + c2(c1(x))$$

Figure 42: Some results shown above

The output has two lines above every image where the upperline is the prediction and lower line is the actual value. The output in green implies correct prediction and red implies a wrong prediction.

4.3 *Different Architectures*

(i) ResNet101

Whenever one wants to increase the accuracy of a classificationmodel, the easiest way to do so is to increase the number of layers. Hence, we change the pretrained model from ResNet50 to ResNet101. This increases our accuracy from 91.3% to 91.7%.

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.177960	0.209554	0.915849	11:43

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.179900	0.211554	0.916622	15:04
1	0.173641	0.209837	0.916436	15:04
2	0.180747	0.210466	0.916230	15:03
3	0.177221	0.208074	0.916405	15:04

Table 4: ResNet101 results

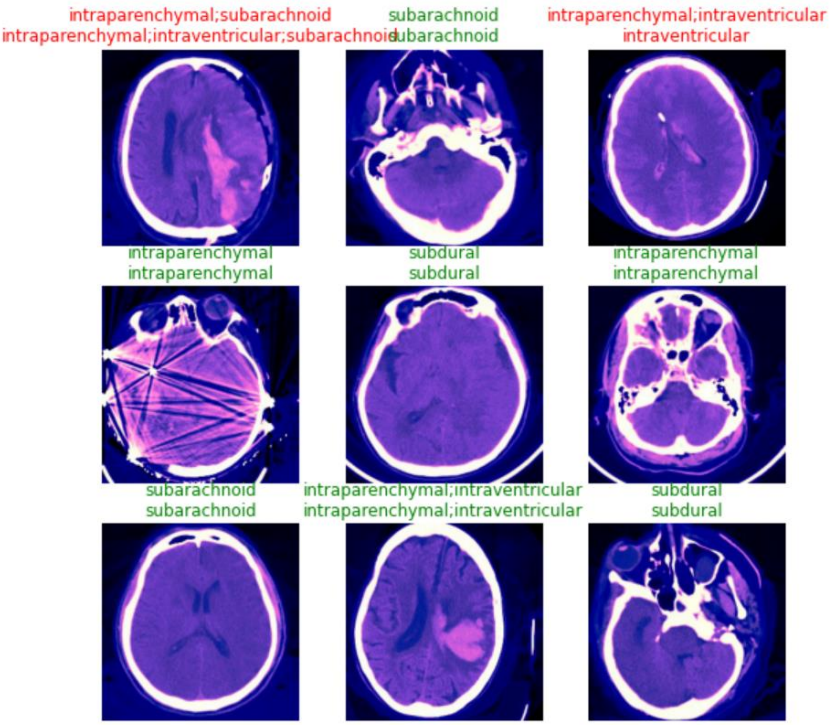


Figure 43: ResNet 101 output visualization

(ii)DenseNet121

The next architecture we try is the DenseNet121 architecture. In DenseNet, each layer receives collective knowledge from all preceding layers. For this, the output of a convolution is concatenated with the input. This concatenated feature-map is passed to the next layer. Since this increases the number of parameters, we reduce the batch size by half. With the DenseNet, we get the accuracy of 91.8%.

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.176242	0.204657	0.917158	10:54
1	0.176698	0.204080	0.917900	10:54
2	0.180510	0.204096	0.917993	10:54
3	0.176864	0.206739	0.917477	10:54

Table 5: DenseNet121 results

(iii) AlexNet

The final architecture we try is the AlexNet. This is another famous architecture for image classification. It won the ImageNet competition in 2012. It uses overlap pooling toreduce the size of the network and trains quite fast.

Although, the training is much faster than the other architectures we used, the accuracy was not better. It came to 89%.

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.274129	0.281940	0.887325	05:36
1	0.287285	1.125553	0.886407	05:37
2	0.261937	0.817691	0.889500	05:38
3	0.267892	1.084262	0.890046	05:37

Table 6: AlexNet121 results

(iv) Weight Decay

It is one of the regularization techniques in Deep Learning. It prevents our model from getting too complex by penalizing complexity. It does so by adding the sum of squares of all parameters to the loss function. However, this can make the loss so huge that the best model would set all the parameters to 0. To prevent this from happening, we multiply the sum ofsquares with another small number. This number is called weight decay (wd).

$$loss = crossent(\bar{y}y) + wd * \sum parameter^2$$

We use weight decay value = 0.01 This helps the model learnmuch better.

4.4 Mixed Precision Training

In neural networks, inputs, parameters and activations are stored using 32 bits. This gives us a high amount of precision. However, these high precision computations, require more time and memory. Hence, using 32-bit precision for all our calculations, we perform some of them in 16 bits. Weight updates need to be as precise as possible hence, we haven't reduced the bits for weight updates.

In theory, mixed precision training should reduce the training time by half. However, in practice we see only a slight decreasein the training time. But, in our implementation, the time did not reduce much.

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.115613	0.215237	0.918137	09:53
1	0.103331	0.216677	0.917869	09:40
2	0.104649	0.216569	0.919054	09:40

Table 7: Mixed precision training results

4.5 Progressive Image Resizing

Research has shown that training a model on smaller sized images and then gradually increasing the size during the train process helps the model learn better. Hence, we start by training a model with images of size (64,64), we then increase it to (128,128) and finally to (256,256). Progressive image resizing does not give good results immediately but only after a lot of epochs.

epoch	train_loss	valid_loss	accuracy_multi	time
0	0.185606	0.218689	0.912490	09:53
1	0.181458	0.216660	0.913221	09:51
2	0.176293	0.216696	0.913417	09:51

Table 8: Progressive image resizing results

4.6 *REFERENCES*

[1] T. Chan, "Computer aided detection of small acute intracranial hemorrhage on computer tomography of brain," *Comput Med Imaging Graph*, 2007;31(4–5):285–298.
[2] W.L Nowinski et al., "Characterization of interventricular and intracerebral intracranial hemorrhages in non-contrast CT," *Neuroradiology J.*, 27: 299-315, 2014.
[3] P. Maduskar, M. Acharyya, "Automatic identification of intracranial hemorrhage in non-contrast CT with large slice thickness for trauma cases," *Proc SPIE*, vol. 7260, 2009.
[4] C. Liao et al., "Computer-aided diagnosis of intracranial intracranial hemorrhage with brain deformation on computed tomography," *Computerized Medical Imaging and Graphics*, 34 (2010) 563-571.
[5] J. Kosior, "Quantomo: validation of a computer-assisted methodology for the volumetric analysis of intracerebral hemorrhage," *Int J Stroke*, vol. 6, 2011, 302-305.
[6] H.-L. Tong et al., "Automated hemorrhage slices detection for CT brain images," *Proc IVIC*, 268-279, 2011.
[7] D. Dowlashahi, "Planimetric intracranial hemorrhage measurement in patients with intraventricular hemorrhage," *Stroke*, 43, 2012, 1961- 1963.
[8] K. Prakesh et al., "Segmentation and quantification of intra- ventricular/cerebral hemorrhage in CT scans by modified distance regularized level set evolution technique," *Int J Computer Assisted Radiological Surgery*, 7(5), 2012, 785-798.
[9] S. Soroushmehr, "CT image segmentation in traumatic brain injury," *Proc IEEE EMBC*, 2015, 2973-2976.
[10] M. Sun et al., "Intracranial hemorrhage detection by 3D voxel segmentation on brain CT images," *Proc 2015 IEEE Int Conf on Wireless Communications and Signal Proc*, 2015.
[11] T. Chan, H. Kuang, "Effect of a computer-aided diagnosis system on clinician's performance in detection of small acute intracranial hemorrhage on computed tomography," *Acad Radiol*, 2008, 15:290- 299.
[12] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alexander A. Alemi; Google Inc.; 2017 Inception-v4, Inception-ResNet and the Impact of Residual H. Kishan Das Menon and V. Janardhan Materials Today: Proceedings 43 (2021) 3706–3714 3713 Connections on Learning; Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17).
[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; Very Deep Convolutional Networks for Large-Scale Image Recognition; 2014; Cornell University; arXiv:1409.10556 [cs.CV].
[14] Mayank Chawla, Saurabh Sharma, Jayanthi Sivaswamy, L. T Kishore, A method for automatic detection and classification of stroke from brain CT images; 2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society.
[15] Monika Grewal, Muktabh Mayank Srivastava, Pulkit Kumar, Srikrishna Varadarajan; 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018).
[16] T.D. Phong, H.N. Duong, H.T. Nguyen, Tran van Hoa, Brain hemorrhage diagnosis by using deep learning, Proceedings of the 2017 International Conference on Machine

Learning and Soft Computing, 2017.

[17] Marcus Badgeley, Javin Schefflein, Margaret Pain, Andres Su, and Michael Cai; Automated deep-neural-network surveillance of cranial images for acute neurologic events, 2018 nature medicine.

[18] M.R. Arbabshirani, B.K. Fornwalt, G.J. Mongelluzzo, Advanced machine learning in action: identification of intracranial hemorrhage on computed tomography scans of the head with clinical workflow integration, Springer Nature, 2018.

[19] Sasank Chilamkurthy, Rohit Ghosh, Swetha Tanamala, Mustafa Biviji, Norbert G. Campeau, Vasantha Kumar Venugopal, Vidur Mahajan, Pooja Rao, Prashant Warier; Development and Validation of Deep Learning Algorithms for Detection of Critical Findings in Head CT Scans; 2018 April, Cornell University; arXiv: 1803.05854.

[20] Hyunkwang Lee Sehyo, Yune Mohammad, Mansouri Myeongchan Kim, Shahein H. Tajmir, Claude E. Guerrier, A Ebert Sarah, R Pomerantz Stuart, Javier M. Romero, Shahmir Kamalian, Ramon G. Gonzalez, Michael H. Lev, Synho Do, An explainable deep-learning algorithm for the detection of acute intracranial hemorrhage from small datasets; 2018 Springer Nature, Nature Biomedical Engineering 3 (2019) 173–182.

[21] SaifengLiu, DavidUtriainen, ChaoChai, YongshengChen, LinWang, Sean K., Sethi, ShuangXia, E. MarkHaacke; Cerebral microbleed detection using Susceptibility Weighted Imaging and deep learning; ELSEVIER NeuroImage Volume 198, September 2019, Pages 271-282.

[22] Sasank Chilamkurthy, Rohit Ghosh, Swetha Tanamala, Pooja Rao and Qure.ai team. Development and Validation of Deep Learning Algorithms for Detection of Critical Findings in Head CT Scans.

[23] Weicheng Kuo, Christian Häne, Pratik Mukherjee, Jitendra Malik, and Esther L. Yuh, Expert-level detection of acute intracranial hemorrhage on head computed tomography using deep learning.

[24] Samuel W. Remedios, Zihao Wu, Camilo Bermudez, Cailey I. Kerley, Snehashis Roy, Mayur B. Patel, John A. Butman, Bennett A. Landman, and Dzung L. Pham, "Extracting 2D weak labels from volume labels using multiple instance learning in CT hemorrhage detection".

[25] Leslie N. Smith, " A disciplined approach to neural network hyper- parameters: Part 1 -- learning rate, batch size, momentum, and weight decay".

[26] Leslie N. Smith, "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates".

[27] Intracranial hemorrhage detection

Kishan Das Menon, H.JanardhanV.

Department of Information Technology, School of Engineering and Technology, CMR University, Bengaluru, India

[28] Detecting Intracranial Hemorrhage with Deep Learning

Arjun Majumdar et al. Annu Int Conf IEEE Eng Med Biol Soc. 2018 Jul.