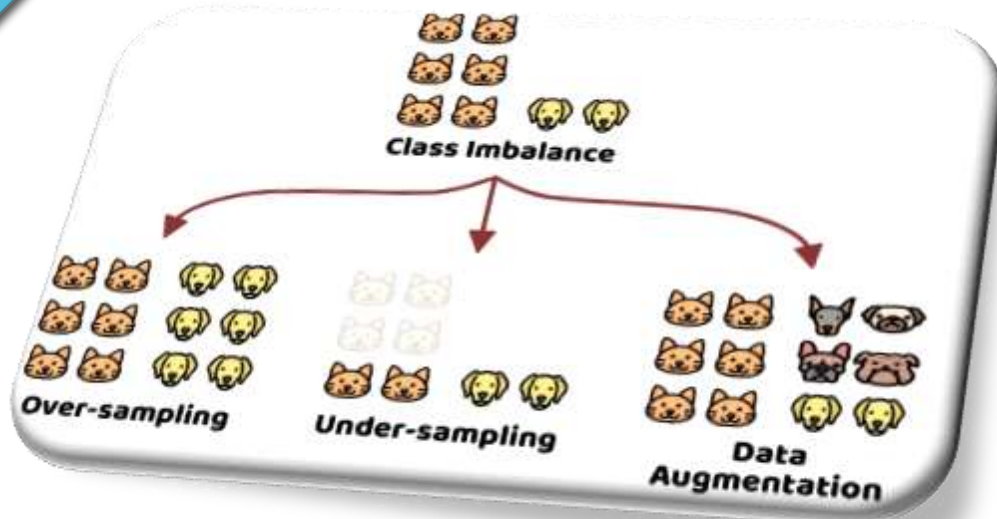


# IMPROVING MODEL PERFORMANCE & REMOVING THE CLASS IMBALANCE PROBLEM USING AUGMENTATION



Allena Venkata  
Sai Abhishek

MTech Data Science

GITAM University, Visakhapatnam, India

# **What is Class Imbalance Problem?**

- ▶ Unequally distributed (Major & Minor Class)

## **Types of Class Imbalance**

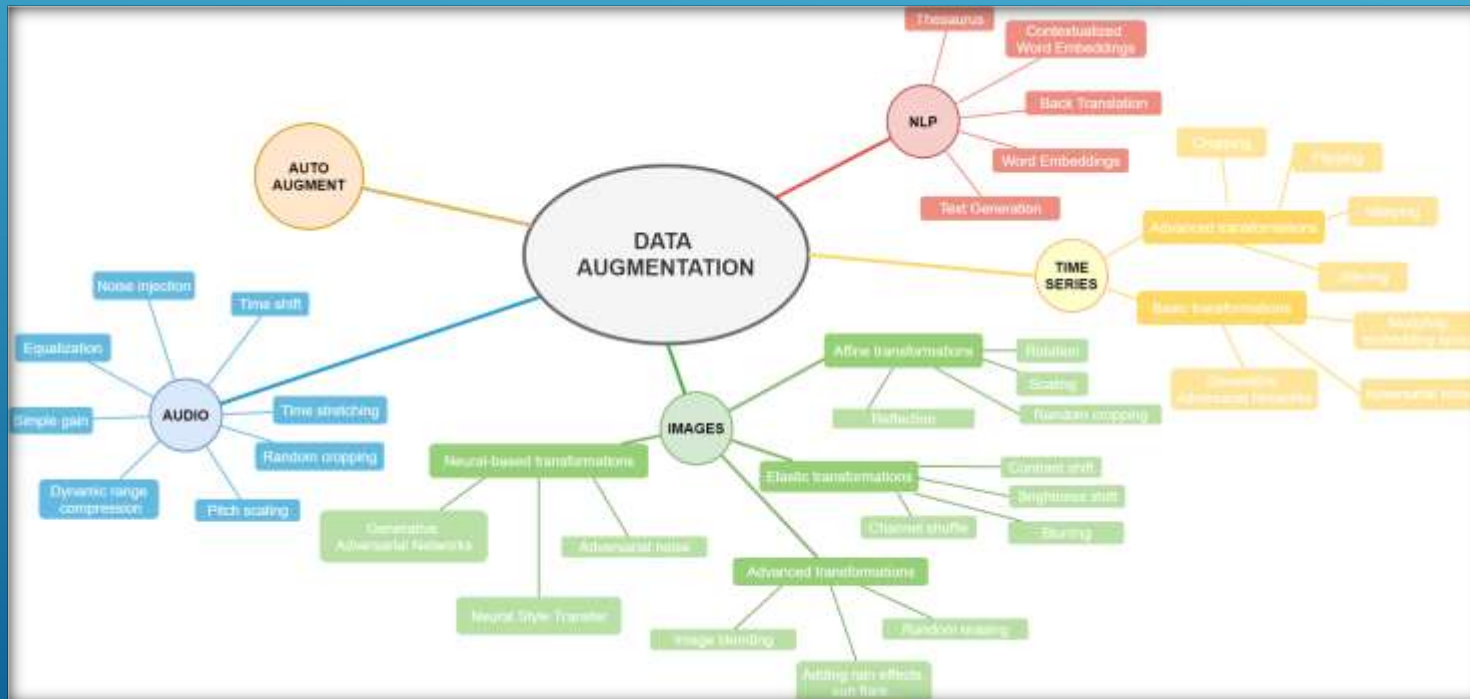
- ▶ Slight Imbalance ( $<1:10$ )
- ▶ Severe Imbalance ( $>1:10$ )

## **Solution for removing Class Imbalance Problem**

- ▶ Over-Sampling
- ▶ Under-Sampling

# What is Augmentation?

- ▶ Data augmentation is a technique to artificially create new training data from existing training data.
- ▶ Data augmentation is a strategy that enables to significantly increase the diversity of data available for training models, without actually collecting new data.



# **Motivation Behind the Topic**

The class imbalance problem is a painful feature in real world data

Many augmentations supported by various augmentation libraries

Several thin, parallel white lines are drawn diagonally across the bottom right corner of the slide, extending from the right edge towards the bottom left.

# LITERATURE SURVEY

	Journal Type and year	Authors	Title	Summary
1	IEEE, 2017	Mateusz Buda, Atsuto Maki, Maciej A. Mazurowski	A systematic study of the class imbalance problem in convolutional neural networks	As opposed to some classical machine learning models, oversampling does not cause overfitting of CNNs
2	IEEE, 2015	Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun	Deep Residual Learning for Image Recognition	residual networks are easier to optimize, and can gain accuracy from considerably increased depth.
3	IEEE, 2019	Connor Taghi M. Khoshgoftaar	A survey on Image Data Augmentation for Deep Learning	Comparative study of various augmentation techniques
4	IEEE, 2020	Wanwan Zheng Mingzhe Jin	The Effects of Class Imbalance and Training Data Size on Classifier Learning	naïve Bayes, logistic regression model are less susceptible to class imbalance while they have relatively poor predictive performance
5	IEEE, 2017	Marcus D. Bloice, Christof Stocker, Andreas Holzinger	Augmentor: An Image Augmentation Library for Machine Learning	Augmentor makes it easier to perform artificial data generation, by providing a stochastic, pipeline-based API that allows for fine-grained control over the creation of augmented data and provides many functions for augmentation technique.

Journal Type and year		Authors	Title	Summary
6	IEEE, 2018	Alexandr A. Kalinin, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Alexander Buslaev	Albumentations: fast and flexible image augmentations	A fast and flexible library for image augmentations with many various image transform operations available, that is also an easy-to-use wrapper around other augmentation libraries
7	IEEE, 2018	Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, David Lopez-Paz	MixUp augmentation for image classification	mixup trains a neural network on convex combinations of pairs of examples and their labels. Turn on screen reader support
8	IEEE, 2019	Zhiting Hu, Bowen Tan, Ruslan Salakhutdinov, Tom Mitchell, Eric P. Xing	Learning Data Manipulation for Augmentation and Weighting	the resulting algorithms significantly improve the image and text classification performance in low data regime and class-imbalance problems.
9	IEEE, 2021	Shanchuan Lin, Linjie Yang, Imran Saleemi, Soumyadip Sengupta	Robust High-Resolution Video Matting with Temporal Guidance	Input is downsampled for the encoder-decoder network, consists of an encoder that extracts individual frame's features, a recurrent decoder that aggregates temporal information, Deep Guided Filter module for high-resolution upsampling. Then DGF is used to upsample result.
10	Springer, 2021	Xu Sun, Huihui Fang Yehui Yang, Dongwei Zhu Lei Wang, Junwei Liu Yanwu Xu	Robust Retinal Vessel Segmentation from a Data Augmentation Perspective	Data augmentation modules, namely, channel-wise random Gamma correction (correction on each channel) and channel-wise random vessel augmentation. (Morphological transformations on fine grained vessels.)

[Click me for some more info](#)

Also in next page too

# **Abstract**

- ▶ One particularly painful feature of real-world data is that it can be imbalanced. An imbalanced dataset is a dataset where there are many more datapoints for one category than others.
- ▶ This research presents the Solving Class Imbalance problems using Random Sampling & Data Augmentation Techniques. Readers will understand how Under-Sampling, Over-Sampling, Synthetic Minority Over-sampling Technique & Data Augmentation using Image and custom datasets.
- ▶ The model performance is being improved with the removing the class imbalance problem using various Augmentation approaches by building a custom Augmented Dataset generator & a Custom Augmentation library
- ▶ The parameters on which the techniques will be compared are on accuracy to solve the Class imbalance problem to maximize accuracy and reduce error & find the best possible method to solve it.



# Proposed System

► **Dataset** : An Image Classification Dataset (Intel Scenery), Chest Xray Pneumonia, CIFAR 10

► **Model** : Resnet18

► **Solution** :

Techniques implemented to solve class imbalance -

► **Under Sampling** –

► Random Under Sampling

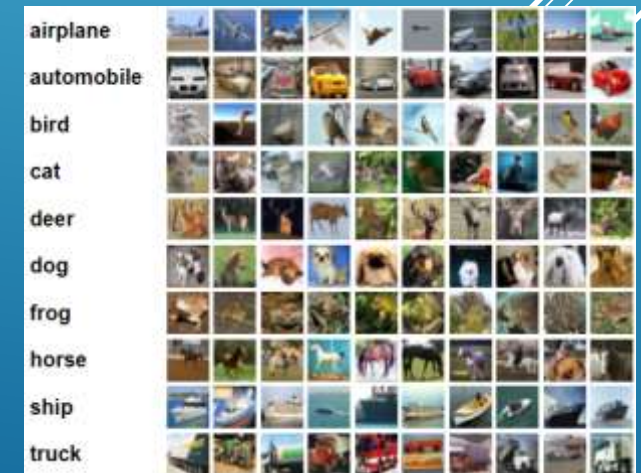
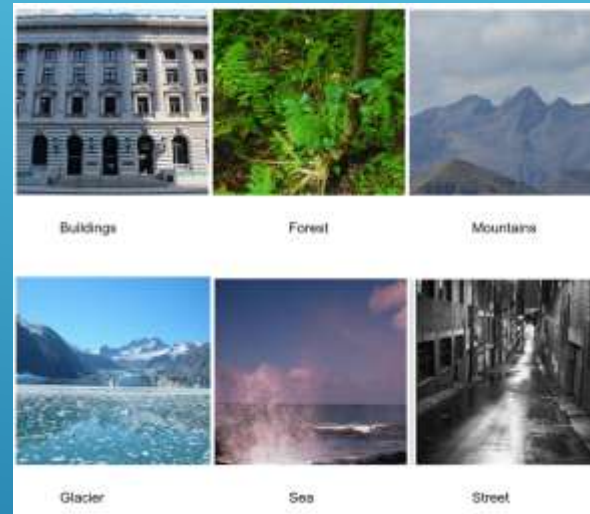
► **Over Sampling** –

► Random Over Sampling,

► Data Augmentation

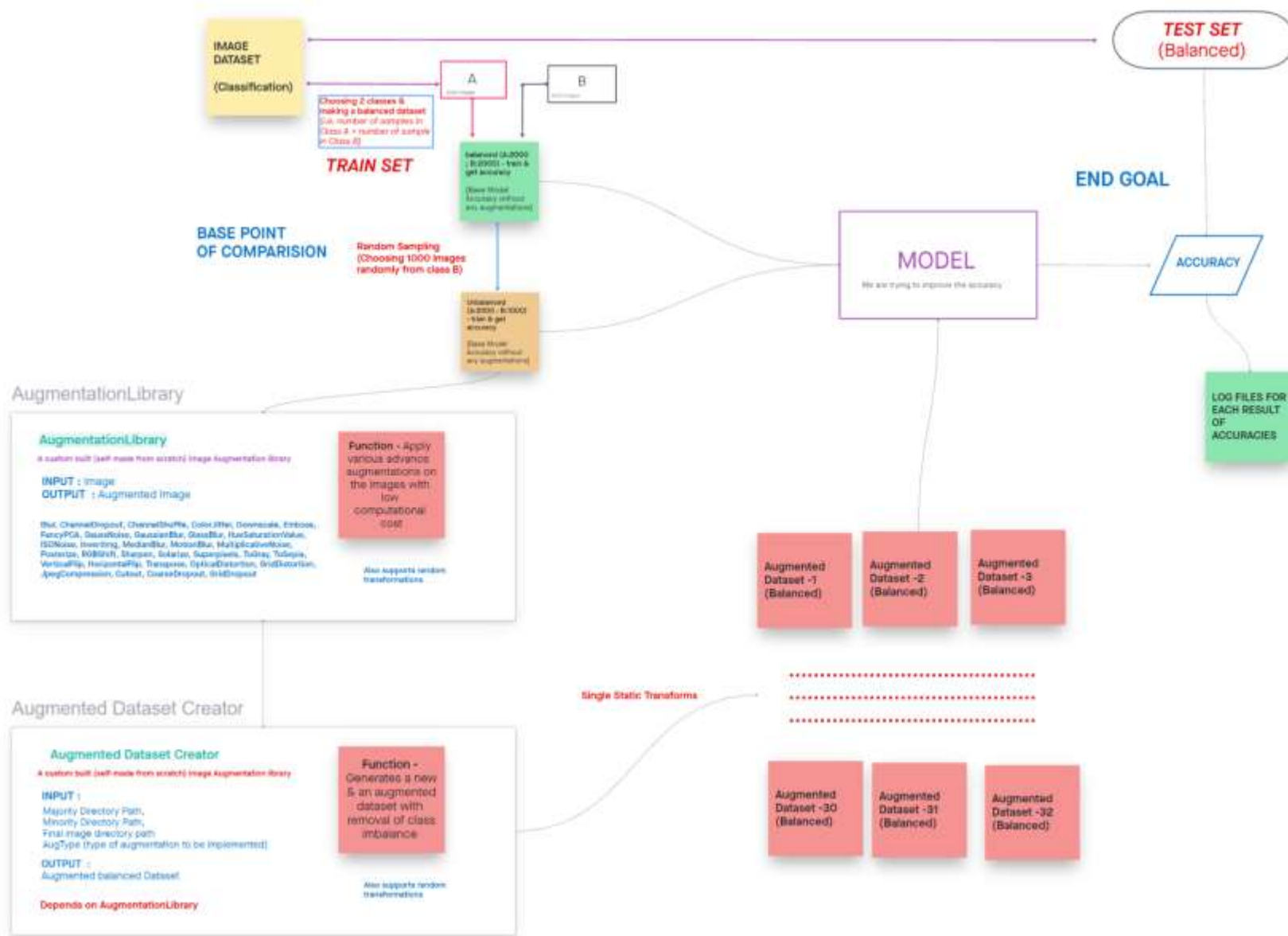
► **Metric used to Compare** : Accuracy

► **Goal** : Find the best Data augmentation technique for the dataset with construction of augmentation library & augmented dataset generator





# Workflow of the Proposed System



# A Video Explanation of the workflow



# AUGMENTATION LIBRARIES

	Package Name	imgaug package	Albumentations package	Augmentor package
	Description	<p>It contains:</p> <p>Over 40 image augmentors and augmentation techniques;</p>	<p>Over 60 pixel level and spatial level transformations</p>	<p>Augmentor package contains less possible augmentations than other packages</p>
		<p>Functionality to augment images with masks, bounding boxes, keypoints and heatmaps.</p>	<p>Transforming images with masks bounding boxes and keypoints;</p>	<p>Extra features like size-preserving rotations, size-preserving shearing and cropping, which is more suitable for machine learning.</p>
		<p>This functionality makes it very easy to augment the dataset containing images for segmentation and object detection problems</p>	<p><b>Organizing augmentations into pipelines -</b> Albumentations package allows to construct advanced pipelines similar to imgaug pipelines.</p>	<p>Augmentor package also allows to compose augmentation pipelines and use them with PyTorch.</p>
		<p>Complex augmentation pipelines</p>	<p><b>PyTorch integration -</b> effortlessly migrate from torchvision to Albumentations, because this package provides specialized utilities to be used with PyTorch. Migrating to Albumentations will help to speed up the data generation part and train deep learning models faster. See detailed tutorial on migration from torchvision to Albumentations</p>	
		<p>Many helper functions for augmentation visualization, conversion, and more.</p>	<p>Albumentations is a fast and flexible image augmentation library.</p>	

# Augmentation Techniques

- GaussianBlur - Blur the input image using a Gaussian filter with a random kernel size.
- GlassBlur - Apply glass noise to the input image.
- HueSaturationValue - Randomly change hue, saturation and value of the input image.
- ISONoise - Apply camera sensor noise.
- InvertImg - Invert the input image by subtracting pixel values from 255.
- MedianBlur - Blur the input image using a median filter with a random aperture linear size.
- MotionBlur - Apply motion blur to the input image using a random-sized kernel.
- MultiplicativeNoise - Multiply image to random number or array of numbers.
- Posterize - Reduce the number of bits for each color channel.
- RGBShift - Randomly shift values for each channel of the input RGB image.
- Sharpen - Sharpen the input image and overlays the result with the original image.
- Solarize - Invert all pixel values above a threshold.

- Blur - Blur the input image using a random-sized kernel.
- CLAHE - Apply Contrast Limited Adaptive Histogram Equalization to the input image.
- ChannelDropout - Drops out a channel based on a range
- ChannelShuffle - Randomly rearrange channels of the input RGB image.
- ColorJitter - Randomly changes the brightness, contrast, and saturation of an image.
- Downscale - Decreases image quality by downscaling and upscaling back.
- Emboss - Emboss the input image and overlays the result with the original image.
- FancyPCA - Augment RGB image using FancyPCA from Krizhevsky's paper "ImageNet Classification with Deep Convolutional Neural Networks"
- GaussNoise - Apply gaussian noise to the input image.

- GridDistortion - Grid-distortion is an image warping technique which is driven by the mapping between equivalent families of curves, arranged in a grid structure. Until recently only curve sets arranged in a regular rectangular grid were considered.
- JpegCompression - Decrease Jpeg compression of an image.
- Cutout - CoarseDropout of the square regions in the image
- CoarseDropout - CoarseDropout of the rectangular regions in the image
- GridDropout - GridDropout, drops out rectangular regions of an image and the corresponding mask in a grid fashion.

- Superpixels - Transform images partially /completely to their superpixel representation. This implementation uses skimage's version of the SLIC algorithm.
- ToGray - Convert the input RGB image to grayscale. If the mean pixel value for the resulting image is greater than 127, invert the resulting grayscale image.
- ToSepia - Applies sepia filter to the input RGB image
- VerticalFlip - Flip the input vertically around the x-axis.
- HorizontalFlip - Flip the input horizontally around the y-axis.
- Transpose - Transpose the input by swapping rows and columns.
- OpticalDistortion - Distortion can be thought of as the difference in magnification across a field of view. This is usually calculated as a percentage of image size. By taking the measured distance in the image and comparing it to the predicted distance, we can calculate the optical distortion



# Augmentation Dataset Creator

Augmented Dataset Creator

## Augmented Dataset Creator

A custom built (self-made from scratch) image Augmentation library

### INPUT :

Majority Directory Path,  
Minority Directory Path,  
Final image directory path  
AugType (type of augmentation to be implemented)

### OUTPUT :

Augmented balanced Dataset

Depends on AugmentationLibrary

**Function -**  
Generates a new  
& an augmented  
dataset with  
removal of class  
imbalance

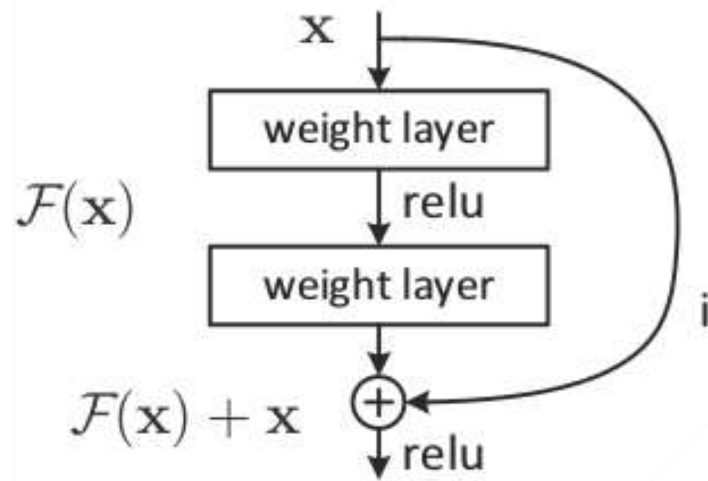
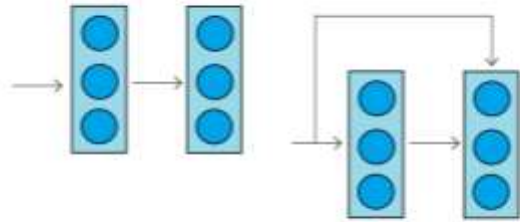
Also supports random  
transformations

# MODEL USED & ARCHITECTURE

## RESNET18

- ❑ X is input, Y is output ,  $Y=F(X)$ . The logic behind RESNET is to make Input=Output
- ❑ If we make  $F(X)=0$ , then it is easy for us to make input = output
$$Y=F(X)+X$$
$$Y=X+0$$
$$Y=X$$
- ❑ In normal networks we learn from Y but in Residual network we learn from F(X) and our target is to make  $F(X)=0$  then only we can make input=output
- ❑ RESNET 1<sup>st</sup> introduced the concept of skip connection.
- ❑ Here, we add the original input to the output of the convolutional block
- ❑ It has 18 layers

# SKIP CONNECTION



Why is the relu applied after adding the skip connection?

- ❑ If we had performed relu before addition, then the residues will all be positives or zero. Only positive increments to the identity are learnt, which significantly reduces the learning capacity.
- ❑ For example in the sin function,  $\sin(3\pi/2) = -1$ , which would need negative residue.
- ❑ Similarly, using sigmoid will also be disadvantageous, because it produces residues only within 0 to 1.

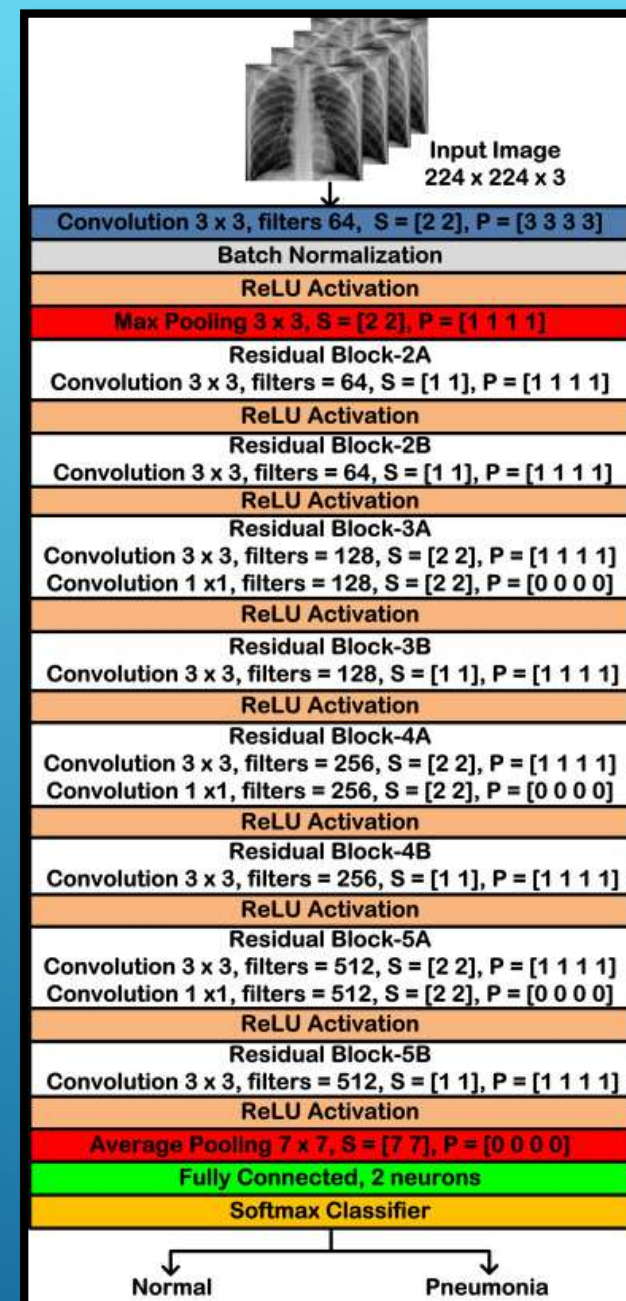
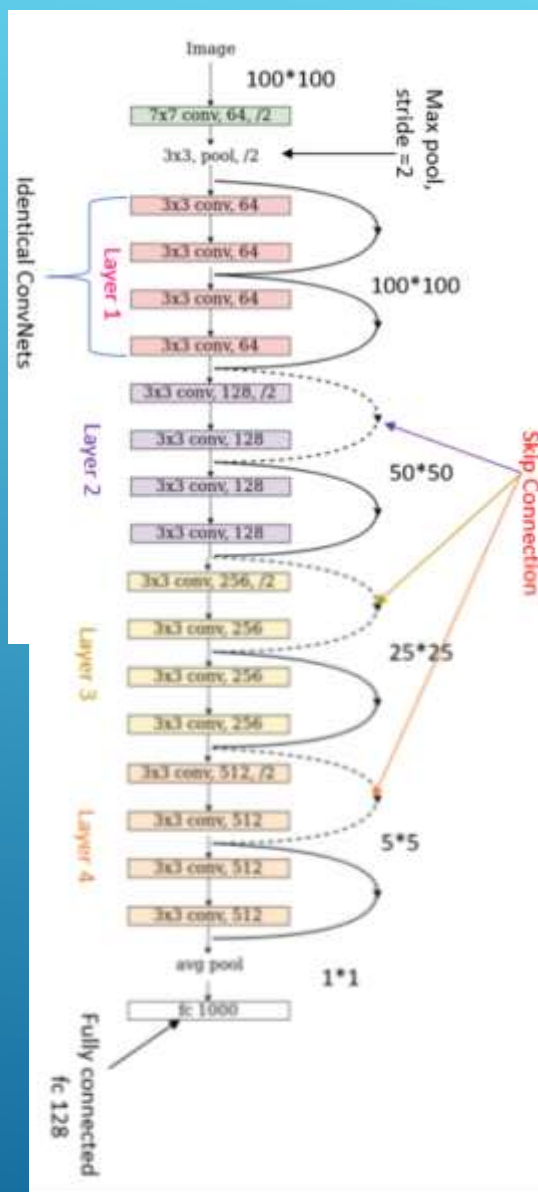
=====

Why are there two weight layers in one residual block?

- ❑ if we had used a single weight layer, adding skip connection before relu, gives  $F(x) = Wx+x$ , which is a simple linear function.
- ❑ This is equivalent to just a single weight layer and there is no point in adding skip connection.
- ❑ So we need at least one non-linearity before adding skip connection, which is achieved by using two layers.



conv1	$112 \times 112 \times 64$	$7 \times 7, 64, \text{stride } 2$
conv2_x	$56 \times 56 \times 64$	$3 \times 3 \text{ max pool, stride } 2$ $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	$7 \times 7 \text{ average pool}$
fully connected	1000	$512 \times 1000 \text{ fully connected}$



1	0	1	0	1	0
0	1	1	0	1	1
1	0	1	0	1	0
1	0	1	1	1	0
0	1	1	0	1	1
1	0	1	0	1	0

Input

1	0	1
0	1	1
1	0	1

Image patch  
(Local receptive field)

\*

1	2	3
4	5	6
7	8	9

Kernel  
(filter)

31			

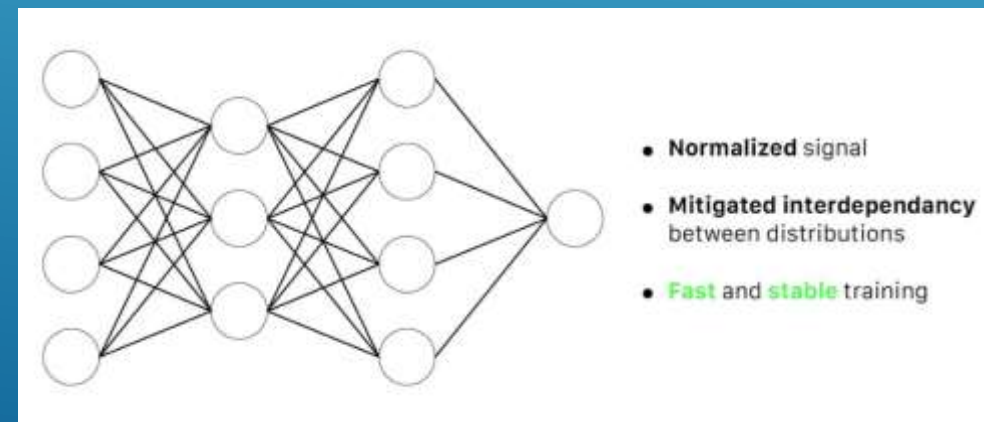
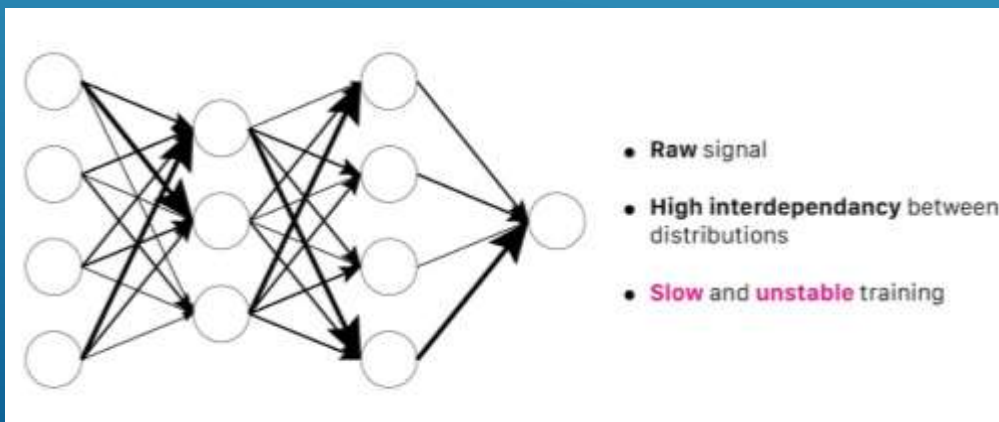
Output

# CONVOLUTION OPERATION

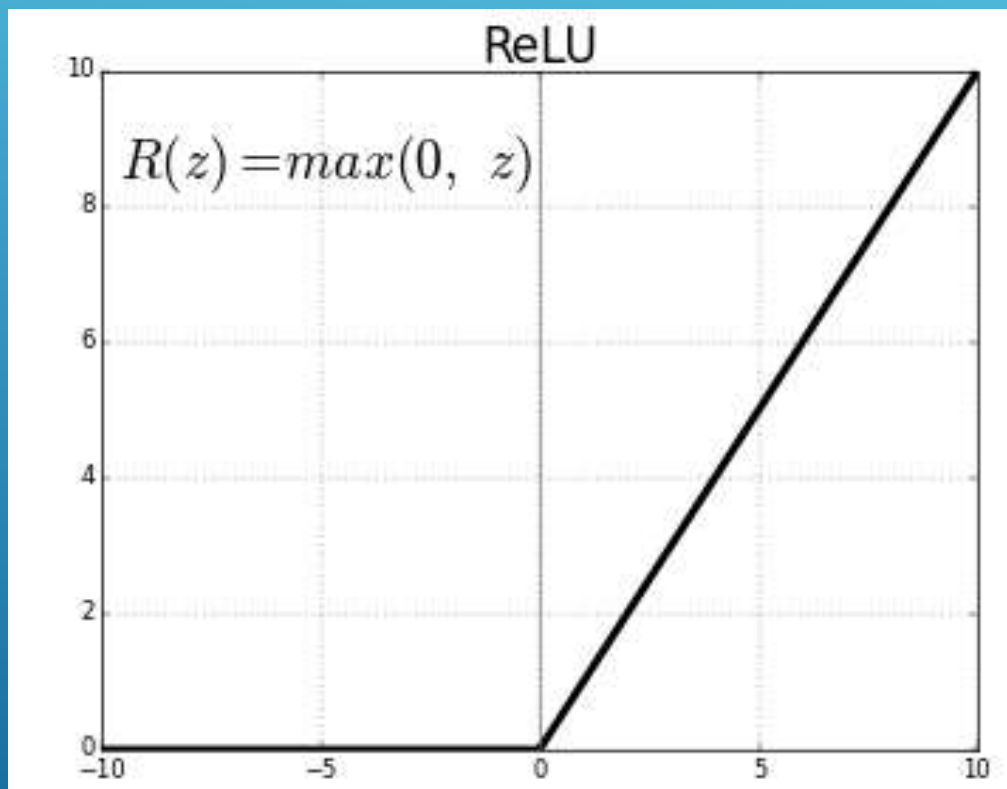
We do elementwise multiplication using a kernel on the input image & it is used for feature extraction.

# BATCH NORMALIZATION

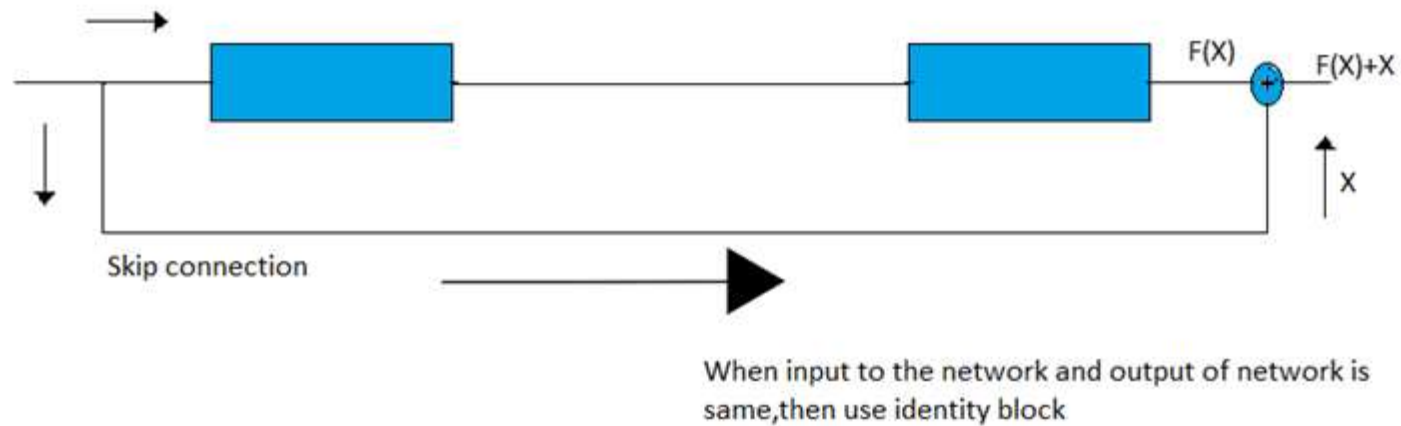
- ❑ For training very deep neural networks
- ❑ the network to do learning more independently.
- ❑ normalize the output of the previous layers before passing them on as the input of the next hidden layer.
- ❑ Stabilizes the learning process
- ❑ Reduces the number of training epochs required to train deep networks.



# RELU ACTIVATION

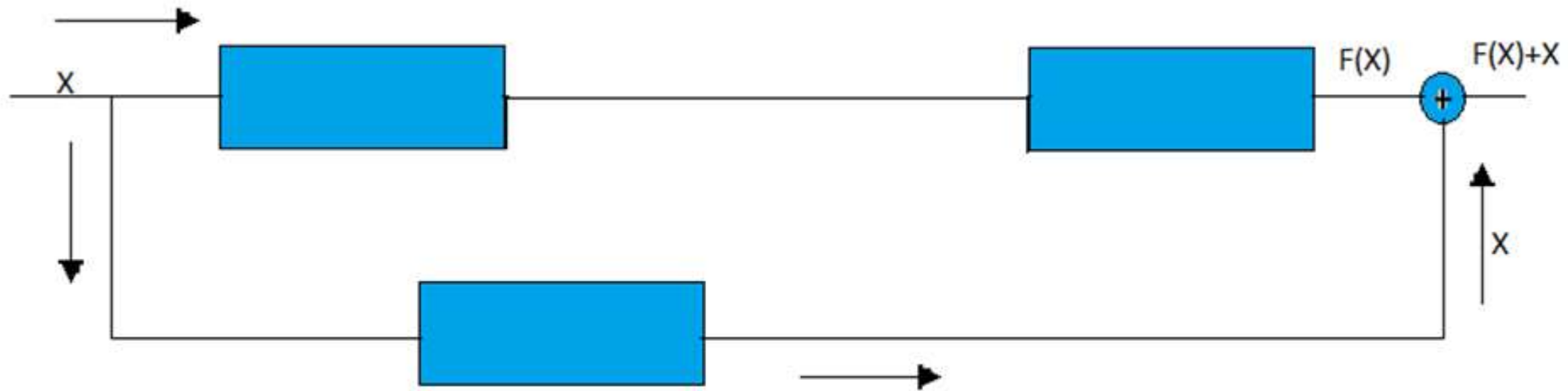


# IDENTITY BLOCK ( INPUT SIZE = OUTPUT SIZE )



# CONVOLUTIONAL BLOCK

( INPUT SIZE  $\neq$  OUTPUT SIZE )



Convolutional layer added in shortcut path



# POOLING OPERATION

❑ Pooling reduces the size of the feature map in order to help us with using the fewer parameters in the network i.e dimensionality reduction.

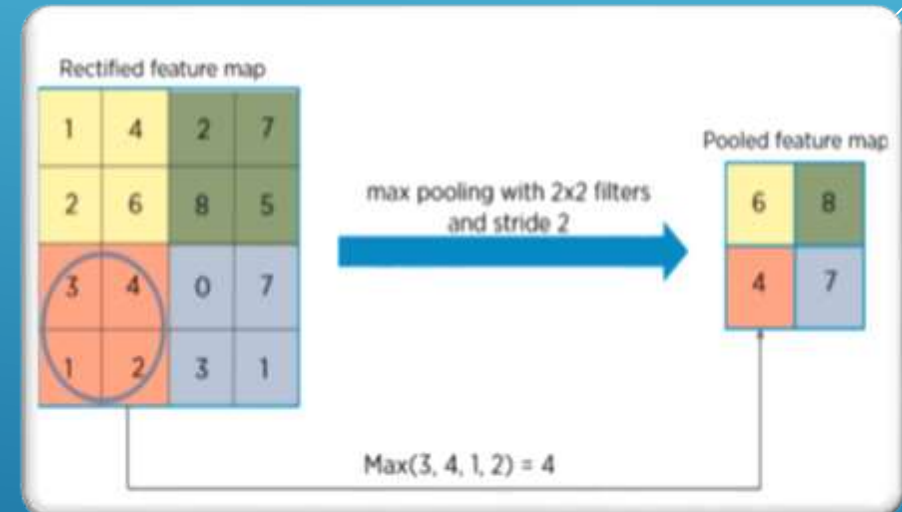
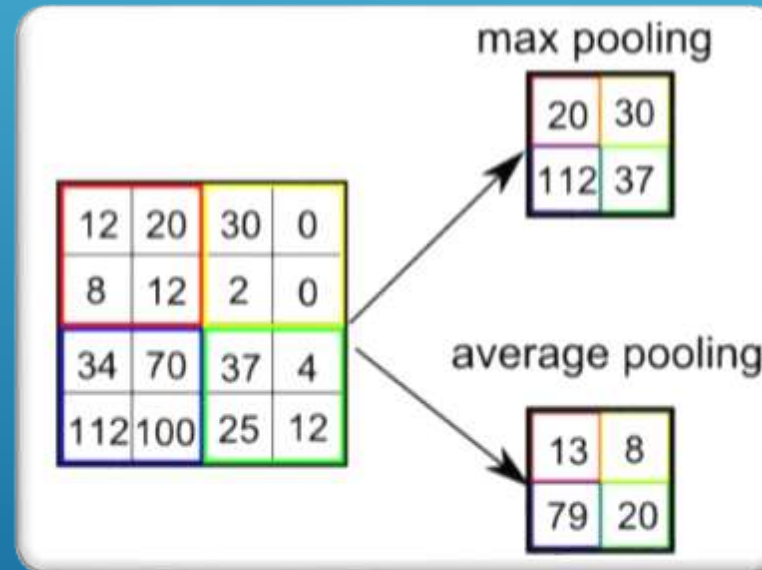
❑ Used to find the essence.

❑ 3 types –

✓ Min - Pixel with Minimum value is selected

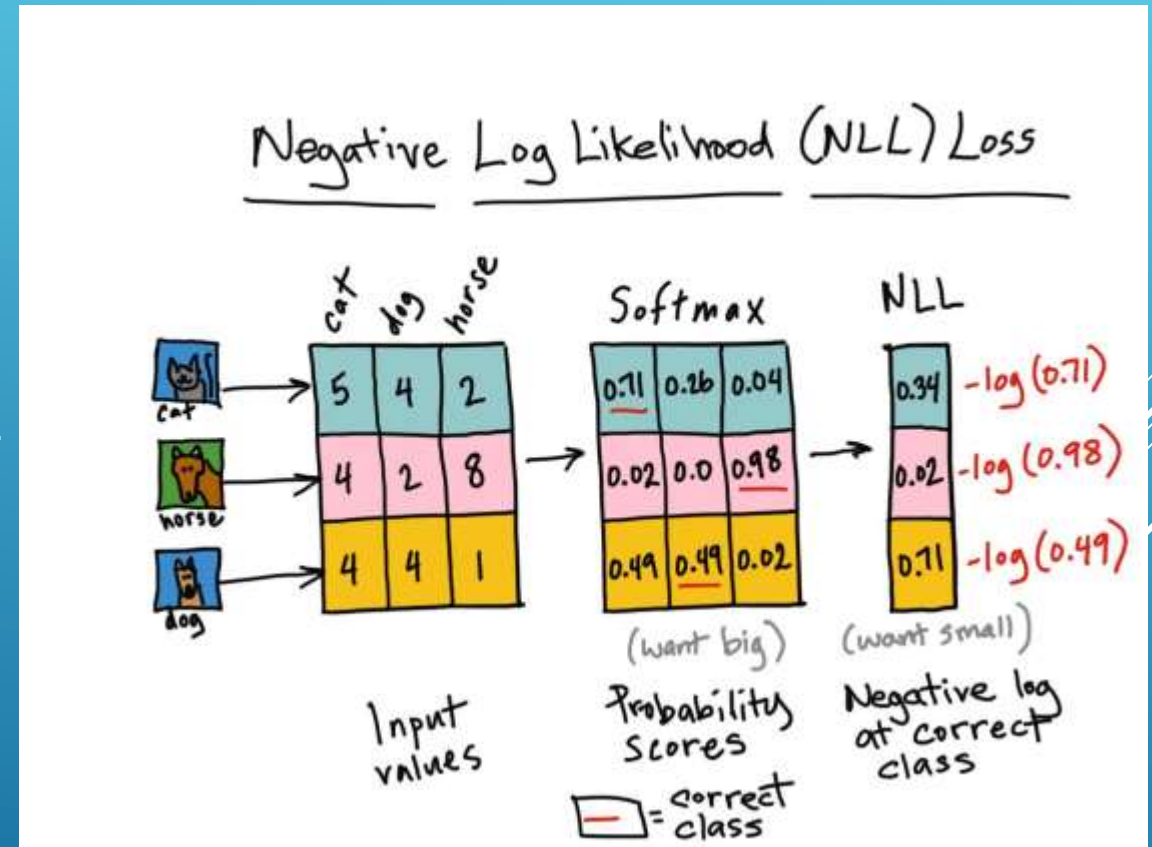
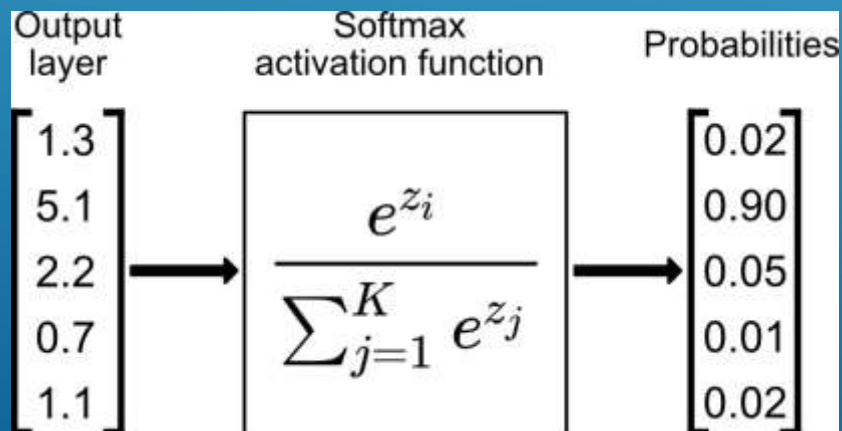
✓ Max – Pixel with Maximum value is selected

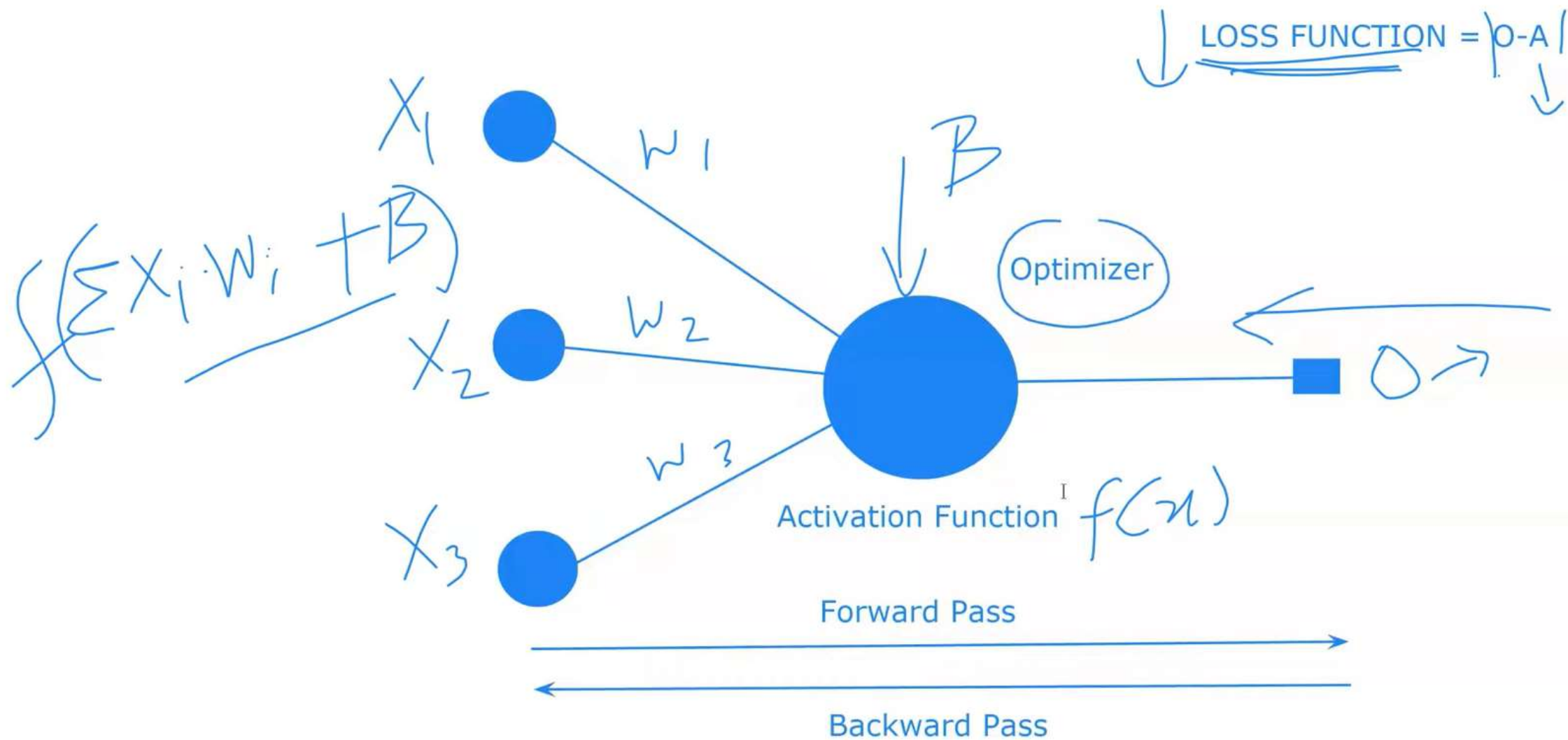
✓ Average – Average of the pixel value is taken



# NEGATIVE LOG LIKELIHOOD LOSS FUNCTION, SOFTMAX & ADAM OPTIMIZER

- ❑ Loss functions define what a good prediction is and isn't a way to measure how well the model is performing.
- ❑ Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate (0.003) to reduce the losses.
- ❑ Adam is the best among the adaptive optimizers in most of the cases. Good with sparse data. a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like SGD with momentum.





# **Research Brief**

- ❑ Compared many Augmentation libraries, and chosen the efficient features of it and built an custom augmentation library.
- ❑ Created a detailed summary of each type of augmentations that the library supports
- ❑ Developed a Dataset Balance & Unbalance Creator
- ❑ Built a Augmentation Dataset Creator based on the Augmentation library

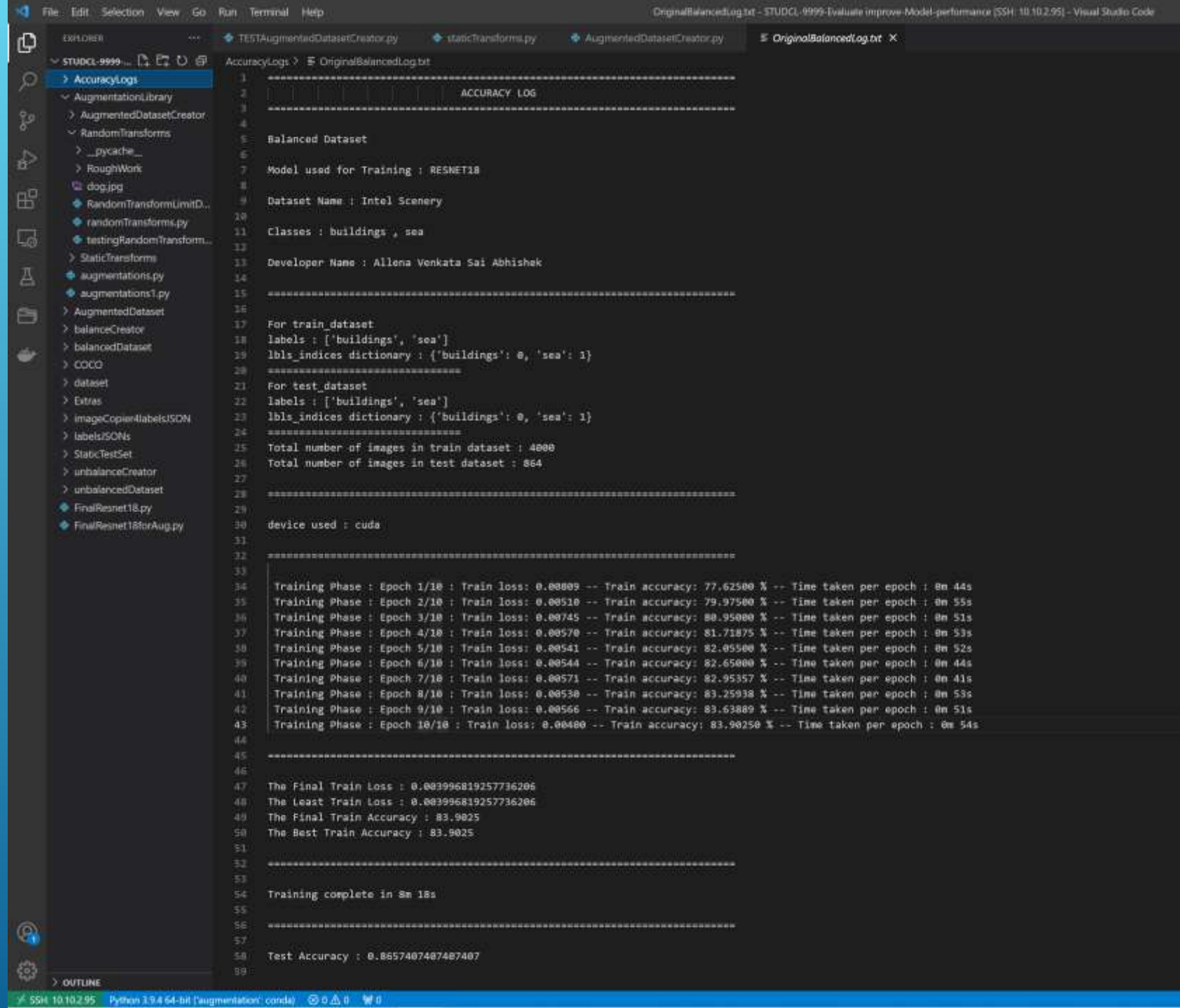
# **Research Brief**

- ❑ Implemented the RESNET18 model
- ❑ Ran for 100 epochs for each test case [Augmented Dataset ] & choosing the best trained model using the test accuracies
- ❑ Ran the test cases for 1:2, 1:4, 1:8, 1:16, 1:20 imbalance
- ❑ Ran the tests for 3 datasets
- ❑ Analyzed & compared the results to compare the augmentations that improve the model accuracy



# RESULTS?

- ❑ Finding accuracies on a fixed test set for all the test cases in which we are training using different test cases
- ❑ Evaluating the model accuracies for each test case saved in accuracy log files
- ❑ Finding out the best Augmentation techniques that gives the best accuracy for the Intel Scenery Dataset



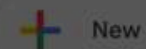
```
File Edit Selection View Go Run Terminal help
TESTAugmentedDatasetCreator.py staticTransforms.py AugmentedDatasetCreator.py OriginalBalancedLog.txt
EXPLORER
STUDCL-9999
  AccuracyLogs
  AugmentationLibrary
  AugmentedDatasetCreator
  RandomTransforms
  _pycache_
  RoughWork
  dog.jpg
  RandomTransformLimitD...
  randomTransforms.py
  testingRandomTransform...
  StaticTransforms
  augmentations.py
  augmentations1.py
  AugmentedDataset
  balanceCreator
  balancedDataset
  COCO
  dataset
  Extras
  imageCopier4labelsISON
  labels/SONs
  StaticTestSet
  unbalanceCreator
  unbalancedDataset
  FinalResnet18.py
  FinalResnet18forAug.py
AccuracyLogs
OriginalBalancedLog.txt
1 -----
2 ACCURACY LOG
3 -----
4
5 Balanced Dataset
6
7 Model used for Training : RESNET18
8
9 Dataset Name : Intel Scenery
10
11 Classes : buildings , sea
12
13 Developer Name : Allena Venkata Sai Abhishek
14
15 -----
16
17 For train dataset
18 labels : ['buildings', 'sea']
19 lbls_indices dictionary : {'buildings': 0, 'sea': 1}
20 -----
21 For test_dataset
22 labels : ['buildings', 'sea']
23 lbls_indices dictionary : {'buildings': 0, 'sea': 1}
24 -----
25 Total number of images in train dataset : 4000
26 Total number of images in test dataset : 864
27
28 -----
29
30 device used : cuda
31
32 -----
33
34 Training Phase : Epoch 1/10 : Train loss: 0.08809 -- Train accuracy: 77.62500 % -- Time taken per epoch : 0m 44s
35 Training Phase : Epoch 2/10 : Train loss: 0.08510 -- Train accuracy: 79.97500 % -- Time taken per epoch : 0m 55s
36 Training Phase : Epoch 3/10 : Train loss: 0.08745 -- Train accuracy: 80.95000 % -- Time taken per epoch : 0m 51s
37 Training Phase : Epoch 4/10 : Train loss: 0.08570 -- Train accuracy: 81.71875 % -- Time taken per epoch : 0m 53s
38 Training Phase : Epoch 5/10 : Train loss: 0.08541 -- Train accuracy: 82.05500 % -- Time taken per epoch : 0m 52s
39 Training Phase : Epoch 6/10 : Train loss: 0.08544 -- Train accuracy: 82.65000 % -- Time taken per epoch : 0m 44s
40 Training Phase : Epoch 7/10 : Train loss: 0.08571 -- Train accuracy: 82.95357 % -- Time taken per epoch : 0m 41s
41 Training Phase : Epoch 8/10 : Train loss: 0.08530 -- Train accuracy: 83.25938 % -- Time taken per epoch : 0m 53s
42 Training Phase : Epoch 9/10 : Train loss: 0.08566 -- Train accuracy: 83.63889 % -- Time taken per epoch : 0m 51s
43 Training Phase : Epoch 10/10 : Train loss: 0.08400 -- Train accuracy: 83.90250 % -- Time taken per epoch : 0m 54s
44
45 -----
46
47 The Final Train Loss : 0.083996819257736206
48 The Least Train Loss : 0.083996819257736206
49 The Final Train Accuracy : 83.9025
50 The Best Train Accuracy : 83.9025
51
52 -----
53
54 Training complete in 8m 18s
55
56 -----
57
58 Test Accuracy : 0.8657407407407407
59
60 -----
61 -----
62 -----
63 -----
64 -----
65 -----
66 -----
67 -----
68 -----
69 -----
70 -----
71 -----
72 -----
73 -----
74 -----
75 -----
76 -----
77 -----
78 -----
79 -----
80 -----
81 -----
82 -----
83 -----
84 -----
85 -----
86 -----
87 -----
88 -----
89 -----
90 -----
91 -----
92 -----
93 -----
94 -----
95 -----
96 -----
97 -----
98 -----
99 -----
100 -----
```





Drive

Search in Drive



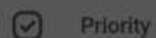
New

My Drive &gt; MTech Final Year Project &gt; Accuracy Logs



Files

Name ↑



Priority



My Drive



Shared with me



Recent



Starred



Trash



Storage

2.56 GB used

# ACCURACY LOGS



OriginalBalancedLog.txt



OriginalUnBalancedLo...



seaBlurLog.txt



seaChannelDropoutLo...



seaChannelShuffleLog...



seaCoarseDropoutLog....



seaColorJitterLog.txt



seaCutoutLog.txt



seaDownscaleLog.txt



seaEmbossLog.txt

# RESULTS OF ACCURACY FOR SINGLE STATIC AUGMENTATIONS

Type of Augmentation		Accuracies
		0.84606
ChannelDropout		0.16203
ChannelShuffle		0.85532
ColorJitter		0.14583
Downscale		0.87152
Emboss		0.85416
FancyPCA		0.86111
GaussNoise		0.5
GaussianBlur		0.86574
GlassBlur		0.85185
HueSaturationValue		

[illegible]

FileHomeInsertPage LayoutFormulasDataReviewViewTell me what you want to do...

Cut

Copy

Paste

Format Painter

Clipboard

Arial

12

B

I

U

Font

Alignment

General

Number

Normal

Bad

Good

Neutral

Calculation

Check Cell

Conditional Formatting

Format as Table

Styles

Cells

Σ AutoSum

Fill

Clear

Editing

Venkata Sri Reshmi Allena

Share

V10

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1																					
2																					
3																					
4																					
5																					
6																					
7																					
8																					
9																					
10																					
11																					
12																					
13																					
14																					
15																					
16																					
17																					
18																					
19																					
20																					
21																					
22																					
23																					
24																					
25																					
26																					
27																					
28																					
29																					
30																					
31																					
32																					
33																					

INTEL SCENE CLASSIFICATION DATA

FINAL METRICS

21

41


81


161


201


74%




 Paste


 Cut


 Copy


 Format Painter


Arial 12


 **B**


 *I*


 U


 A

















General

 %





Normal


Bad


Good


Neutral


Calculation


Check Cell


 Insert


 Delete


 Format

 AutoSum

 Fill

 Clear

 Sort & Filter

 Find & Select

Clipboard Font Alignment Number Styles Cells Editing

V10

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
2			Above Max Threshold																		
3			Below Min Threshold																		
4																					
5																					
6			2:1 (2000:1000)				4:1 (2000:500)				8:1 (2000:250)				16:1 (2000:125)				20:1 (2000:100)		
7			BASE MAXIMUM ACCURACY	0.86574			BASE MAXIMUM ACCURACY	0.86574			BASE MAXIMUM ACCURACY	0.86574			BASE MAXIMUM ACCURACY	0.86574			BASE MAXIMUM ACCURACY	0.86574	
8			BASE MINIMUM ACCURACY	0.84722			BASE MINIMUM ACCURACY	0.79745			BASE MINIMUM ACCURACY	0.65046			BASE MINIMUM ACCURACY	0.59143			BASE MINIMUM ACCURACY	0.58217	
9																					
10			Type of Augmentation	Accuracies			Type of Augmentation	Accuracies			Type of Augmentation	Accuracies			Type of Augmentation	Accuracies			Type of Augmentation	Accuracies	
11	1		CoarseDropout	0.87615		1	JpegCompression	0.86689		1	ColorJitter	0.86226		1	InvertImg	0.87731		1	FancyPCA	0.87731	
12	2		Emboss	0.87152		2	RGBShift	0.86574		2	JpegCompression	0.86226		2	VerticalFlip	0.86921		2	MultiplicativeNoise	0.86805	
13	3		Blur	0.86921		3	HorizontalFlip	0.86458		3	RGBShift	0.86111		3	HorizontalFlip	0.86574		3	HueSaturationValue	0.86458	
14	4		InvertImg	0.86805		4	MultiplicativeNoise	0.86342		4	GaussNoise	0.85995		4	RGBShift	0.86458		4	RGBShift	0.86342	
15	5		GlassBlur	0.86574		5	GaussNoise	0.86111		5	Superpixels	0.85879		5	GaussNoise	0.86458		5	VerticalFlip	0.86226	
16	6		GaussNoise	0.86111		6	GridDistortion	0.85069		6	InvertImg	0.85763		6	RGBShift	0.85648		6	HorizontalFlip	0.85879	
17	7		RGBShift	0.86111		7	InvertImg	0.84953		7	Downscale	0.85185		7	GridDistortion	0.84375		7	ColorJitter	0.85648	
18	8		ISONoise	0.85995		8	HueSaturationValue	0.84837		8	FancyPCA	0.85069		8	HueSaturationValue	0.84143		8	Posterize	0.85532	
19	9		HorizontalFlip	0.85995		9	OpticalDistortion	0.84606		9	OpticalDistortion	0.84837		9	MultiplicativeNoise	0.84143		9	JpegCompression	0.85069	
20	10		CLAHE	0.85612		10	Blur	0.8449		10	VerticalFlip	0.84722		10	MotionBlur	0.84027		10	Downscale	0.84143	
21	11		ColorJitter	0.85532		11	CoarseDropout	0.84375		11	MultiplicativeNoise	0.8449		11	Posterize	0.83796		11	Transpose	0.84027	
22	12		Solarize	0.85532		12	CLAHE	0.83912		12	ISONoise	0.84259		12	Transpose	0.83796		12	ISONoise	0.83912	
23	13		FancyPCA	0.85416		13	VerticalFlip	0.83912		13	Transpose	0.84259		13	Downscale	0.8368		13	ChannelShuffle	0.82986	
24	14		OpticalDistortion	0.85416		14	FancyPCA	0.83333		14	Emboss	0.84027		14	Emboss	0.83217		14	Emboss	0.82175	
25	15		Posterize	0.853		15	MotionBlur	0.83217		15	Posterize	0.83912		15	JpegCompression	0.83217		15	GaussNoise	0.81828	
26	16		HueSaturationValue	0.85185		16	ToSepia	0.83217		16	HorizontalFlip	0.8368		16	CoarseDropout	0.82754		16	Superpixels	0.81828	
27	17		ChannelShuffle	0.84837		17	Emboss	0.82986		17	CoarseDropout	0.8368		17	ChannelShuffle	0.81597		17	InvertImg	0.81134	
28	18		MotionBlur	0.84722		18	ColorJitter	0.82754		18	MotionBlur	0.83101		18	ISONoise	0.81597		18	GridDistortion	0.81134	
29	19		JpegCompression	0.84722		19	Downscale	0.82407		19	HueSaturationValue	0.82986		19	OpticalDistortion	0.80787		19	Cutout	0.79745	
30	20		ChannelDropout	0.84606		20	Transpose	0.82407		20	MedianBlur	0.82523		20	Superpixels	0.80671		20	MotionBlur	0.79398	
31	21		MultiplicativeNoise	0.84606		21	MedianBlur	0.82291		21	ChannelShuffle	0.82407		21	ColorJitter	0.80092		21	OpticalDistortion	0.78125	
32	22		GridDistortion	0.84259		22	Posterize	0.82291		22	GridDistortion	0.82407		22	MedianBlur	0.79513		22	CoarseDropout	0.7662	
33	23		Sharpen	0.83912		23	ISONoise	0.81944		23	ToGray	0.81944		23	Cutout	0.79513		23	ToGray	0.76388	
34	24		VerticalFlip	0.83912		24	GlassBlur	0.81365		24	Blur	0.79513		24	ToGray	0.76504		24	GlassBlur	0.75231	
35	25		Superpixels	0.8368		25	Superpixels	0.80324		25	Cutout	0.78819		25	GaussianBlur	0.74074		25	MedianBlur	0.73726	
36	26		GaussianBlur	0.83101		26	ToGray	0.80208		26	GaussianBlur	0.78703		26	Blur	0.728		26	Blur	0.7118	
37	27		MedianBlur	0.8287		27	Cutout	0.79976		27	GlassBlur	0.78472		27	GlassBlur	0.68634		27	ToSepia	0.71064	
38	28		Cutout	0.8287		28	GaussianBlur	0.7905		28	CLAHE	0.7662		28	ToSepia	0.66898		28	GaussianBlur	0.70138	
39	29		Transpose	0.82407		29	ChannelDropout	0.77083		29	ToSepia	0.74537		29	ChannelDropout	0.61921		29	ChannelDropout	0.66319	
40	30		Downscale	0.82175		30	GridDropout	0.77083		30	ChannelDropout	0.73958		30	Solarize	0.60185		30	CLAHE	0.63078	

INTEL SCENE CLASSIFICATION DATA FINAL METRICS 21 41 81 161 201

# Conclusion

## Effectiveness of augmentations techniques for the datasets

### For Intel Scene Dataset -

INEFFECTIVE	EFFECTIVE
GaussianBlur ChannelDropout GridDropout Sharpen Solarize	FancyPCA GaussNoise InvertImg RGBShift HueSaturationValue JpegCompression MultiplicativeNoise

### For Chest Xray Pneumonia Dataset -

INEFFECTIVE	EFFECTIVE
GaussianBlur Glass Blur ChannelDropout GridDropout Sharpen Solarize	VerticalFlip Transpose GaussNoise MotionBlur HueSaturationValue

### For CIFAR 10 Dataset -

INEFFECTIVE	EFFECTIVE
GaussianBlur ChannelDropout GridDropout Sharpen Solarize	FancyPCA GaussNoise InvertImg RGBShift HueSaturationValue JpegCompression MultiplicativeNoise



# FUTURE SCOPE

Augmentation libraries and techniques have improved a lot over the past decade with an exponential increase in data. There is a lot of research going on & few scope for future are to implement additional features such as getting the Random Transformations with the type and parameter values of it, Implementing the Random Augmentations in Augmented Dataset Creator, Getting the Accuracies for Random Augmented Datasets, implementing more cases an combination by using multiple transforms and getting the prediction scores and using the combination of the majority, moderate and lower prediction scores for the augmentation techniques. Then getting the accuracies of each for evaluation. GANs can also be implemented to create the augmented images. Data scientists & ML Engineers can use the system to improve the model to learn more with less data by implementing the effective augmentation for the datasets based on insights derived from this research with further implementation of the research in data augmentation & machine learning. This will help the customers to enhance the business etc.

# References

1. **A systematic study of the class imbalance problem in convolutional neural networks** - Mateusz Buda, Atsuto Maki, Maciej A. Mazurowski ( <https://arxiv.org/pdf/1710.05381v2.pdf> )
2. **Deep Residual Learning for Image Recognition** Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun ( <https://arxiv.org/pdf/1512.03385v1.pdf> )
3. **A survey on Image Data Augmentation for Deep Learning** Connor Taghi M. Khoshgoftaar ( <https://journalofbigdata.springeropen.com/track/pdf/10.1186/s40537-019-0197-0.pdf> )
4. **Radial-Based Undersampling for Imbalanced Data Classification** Michał Koziarski ( <https://arxiv.org/pdf/1906.00452v2.pdf> )
5. **Augmentor: An Image Augmentation Library for Machine Learning** Marcus D. Bloice, Christof Stocker, Andreas Holzinger ( <https://arxiv.org/pdf/1708.04680.pdf> )
6. **Albumentations: fast and flexible image Augmentations** Alexandr A. Kalinin, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Alexander Buslaev ( <https://www.mdpi.com/2078-2489/11/2/125> )
7. **MixUp augmentation for image classification** Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, David Lopez-Paz ( <https://arxiv.org/pdf/1710.09412.pdf> )
8. **Learning Data Manipulation for Augmentation and Weighting** Zhiting Hu, Bowen Tan, Ruslan Salakhutdinov, Tom Mitchell, Eric P. Xing ( <https://arxiv.org/pdf/1910.12795v1.pdf> )
9. **Robust High-Resolution Video Matting with Temporal Guidance** Shanchuan Lin, Linjie Yang, Imran Saleemi, Soumyadip Sengupta ( <https://arxiv.org/pdf/2108.11515.pdf> )
10. **Robust Retinal Vessel Segmentation from a Data Augmentation Perspective** Xu Sun, Huihui Fang, Yehui Yang, Dongwei Zhu, Lei Wang, Junwei Liu, Yanwu Xu ( <https://arxiv.org/pdf/2007.15883.pdf> )
11. **The Effects of Class Imbalance and Training Data Size on Classifier Learning** Wanwan Zheng, Mingzhe Jin ( <https://link.springer.com/article/10.1007/s42979-020-0074-0> )
12. **ImageNet Classification with Deep Convolutional Neural Networks** Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton ( <https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf> )
13. **Generative Adversarial Networks** Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio ( <https://arxiv.org/pdf/1406.2661v1.pdf> )
14. **SMOTE: Synthetic Minority Over-sampling Technique** N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer ( <https://arxiv.org/pdf/1106.1813v1.pdf> )
15. **ResNet-like Architecture with Low Hardware Requirements** Elena Limonova, Daniil Alfonso, Dmitry Nikolaev, Vladimir V. Arlazarov ( <https://arxiv.org/pdf/2009.07190v2.pdf> )

# SRS (Software Requirement Specification)

- ▶ Text Editor : VS Code
- ▶ OS: Windows/Linux/macOS
- ▶ Storage Space : 6GB
- ▶ RAM : 8GB
- ▶ GPU Required : Advisable, for running the model faster (NVIDIA GPU)

# Libraries used

- ▶ Numpy (For arrays)
- ▶ PIL (Pillow/Python Image Library)
- ▶ OpenCV
- ▶ Tensorflow ( Implement deep learning algorithms, since it allows us to take advantage of GPUs for more efficient training )
- ▶ Keras (offer simple and consistent APIs & for developing and evaluating deep learning models) [*Being a high-level API on top of TensorFlow, we can say that Keras makes TensorFlow easy.* ]
- ▶ Matplotlib
- ▶ Sys
- ▶ Torch ( Version - '1.10.0' )
- ▶ Pytorch (For using inbuilt transforms)
- ▶ Pprint
- ▶ Os
- ▶ Random
- ▶ Pandas (If need to store data in CSV etc)
- ▶ JSON ( storing the data generated in a JSON etc )
- ▶ Albumentation (Using transforms)
- ▶ Glob
- ▶ Shutil

**THANK YOU**

