A Project Report

On

Analysis of Lung cancer CT Scan images using deep neural networks

BY

Under the supervision of

Dr. Paresh Saxena

SUBMITTED IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS OF

CS F425: Deep Learning

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)

HYDERABAD CAMPUS

(Dec 2022)

Group Details

NAME	ID
Aaditya Rathi	2020A7PS2191H
Anirudh Narayan. D	2020A7PS0783H
Chirag Gadia	2020A7PS1721H
Vavilala Hrushikesh Reddy	2020A7PS0030H
Sankalp Kulkarni	2020A7PS1097H
Kedarnath Senegavaram	2020A7PS0245H

ACKNOWLEDGEMENTS

We are grateful to BITS Pilani for giving us this insightful opportunity of working on such a interesting course project. We would also like to thank the CSIS department for allowing us to work on a problem in the domain of Deep Learning.

Finally, we would like to express my profound gratitude to Dr. Paresh Saxena for allowing us to work under his supervision and clearing our doubts throughout the duration of the course. This project would have been impossible without his constant input.

Special thanks to Harsha Varun Marisetty, for the invaluable help and unwavering support in times of need.

Table of Contents

Contents

1. Introduction

Being one of the most common cancers, early diagnosis of lung cancer is necessary. The early-stage lung is generally small and somewhat difficult to diagnose. Various Convolutional Neural Networks (CNNs) architectures have been used to detect cancerous lung tumors. The size of such lung nodules ranges from 3 to 30 millimeters. Due to such high variance, classification and malignancy detection is challenging. Lung nodules have varying sizes and shapes. Nodule shape and size are extracted by global features, while nodule density and texture are identified by local features.

2. Literature Review

(Mundher Al-Shabia) proposed the novel architecture Local-Global Network. The Paper used self-attention layers based on non-local neural networks to extract global features. And it uses residual blocks derived from ResNet architecture. The self-attention methodology used in this paper is the implementation of a Self-Attention Generative Adversarial Network(SAGAN) by Goodfellow.

The most essential characteristic of Residual block is residual or skip connection. This connection solves the vanishing and exploding gradient problem by allowing the network to learn identity functions. It was first used to tackle the degradation problem in very deep neural networks. This paper uses a small residual block of kernel size 3x3 to extract local features. Small kernel size ensures that the number of parameters is low and faster computation.
Figure 1 shows a schematic representation of the architecture used in the paper.
BN here denotes batch normalization.

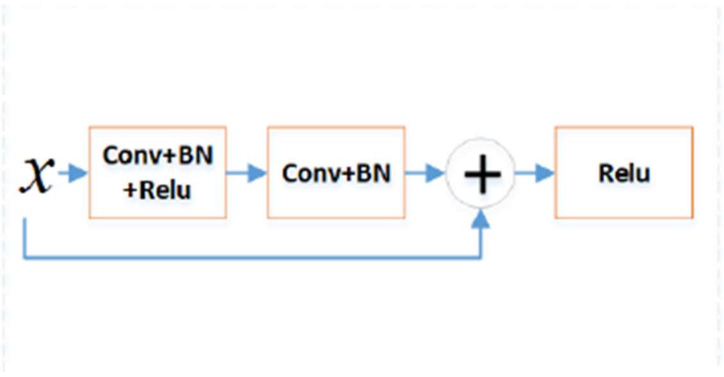


Figure 1: Residual Block

Generally, capturing global features will require bigger kernels which corresponds to high computations. The alternative is to use dilated convolutions, which allow us to capture global features with less number of parameters. Non-local blocks used to extract global features consist of different components. First, it computes the 1x1 convolution of the input features to get f(x), g(x), and h(x). Then every feature in g(x) is multiplied by f(x). Matrix multiplication allows us to capture global features without the use of parameters. In terms of attention modules, g(x) and f(x) are query and key. Since this is a self-attention query and the keys are the same. The method described here is represented mathematically by the equation below.

$$x + h(x)\, Softmax(f(x)^T g(x))^T\, \gamma$$

SoftMax ensures that output lies between 0 and 1 and its sum is 1. These feature maps act as attention masks, focusing attention on important features in h(x). This also implies that zero and low attention features are not passed to the next layer. Gamma in this equation is to regulate the influence of non-local features on the final output. Gamma here is initialized to zero.

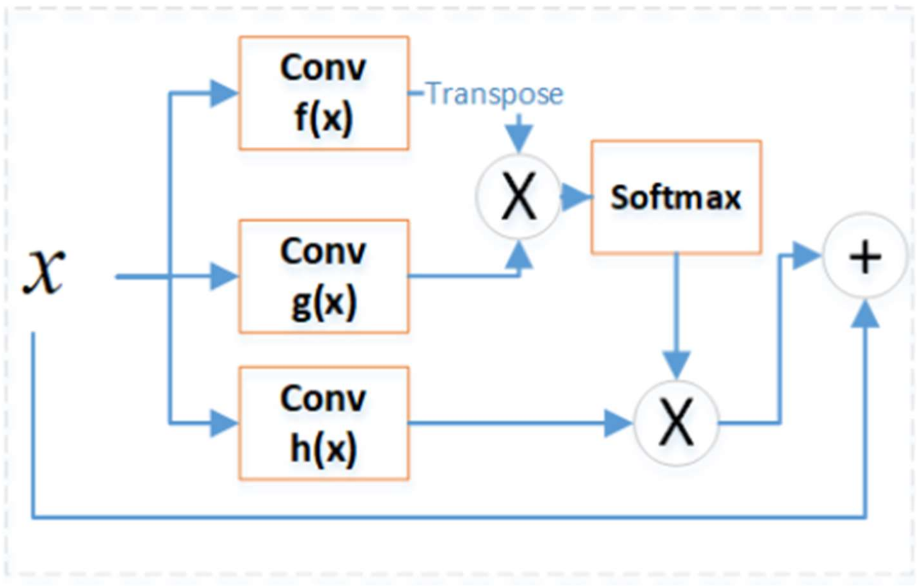


Figure 2: Self Attention Module

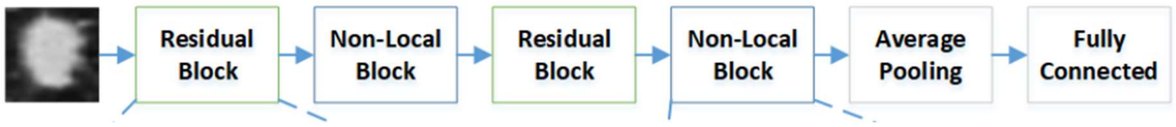


Figure 3: Network Architecture

The Paper compared the proposed method with various baseline methods, as listed in the table below.

Methodology	*AUC*	*Accuracy*	*Precision*	*Sensitivity*
Local-Global Our-implementation	94.36%	93%	89.47%	0.8292
Local-Global	95.62%	88.46%	0.8738	0.8866
Basic Resnet	93.42%	85.63%	0.8242	0.8892
AllAtn	86.18%	78.80%	0.7616	0.8103
AllAtnBig	85.89%	77.97%	0.8137	0.6995
Resnet50	86.82%	77.62%	0.8016	0.7069
Resnet18	86.41%	78.21%	0.7855	0.7488
Densenet121	92.50%	84.57%	0.8686	0.7980

Table 1: Comparison of our original implementation and various baseline methods.

3. Implementation Details

The Paper used the LIDC-IDRI dataset. This is a public dataset that contains around 1000 Computed Tomography Scans. The size of dataset was around 125 GB. So, we decided to use pre-processed dataset. The preprocessed dataset contained only lung nodules extracted from CT scans based on annotations done by Radiologists. A label of 1 means the tumor is malignant. We tried implementing the model in our GPUs using Pytorch with CUDA support. But we faced CUDA out of Memory error. We tried to use Google Colab. But our dataset was a very large number of small-sized images, which was causing issues during uploading and mounting.

https://github.com/raggg377/Deep_Learning_Project

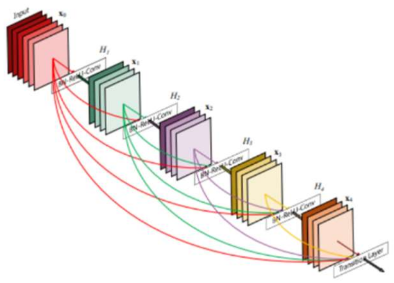
4. Our Contribution

4.1 Shortcomings of Original Paper

1. Very few ResNet layers in the Residual Blocks

   The authors of the paper we implemented have cited a paper, "Deep Residual Learning for Image Recognition" (Kaiming He), that introduced the Residual Block used in the current Local Global model. This paper has three original models containing 50, 101, and 152 ResNet layers in each Residual Block, but the authors instead chose to use only two ResNet layers in the Residual blocks in the Local Global model owing to the scarce dataset. So we decided to note the changes in accuracy and AUC when the number of ResNet layers is increased and see if the model is indeed overfitting or not.

2. Using ResNet instead of DenseNet



The DenseNet Architecture

DenseNet uses previous layers' output to predict the output of the next layer. We thought DenseNet would perform better because DenseNet uses previous layers' output to predict the output of the next layer. So, we thought it would improve the accuracy by preventing the loss caused by the Vanishing Gradient and the Exploding Gradient. But we observed that AUC and accuracy decreased, so we decided to continue with ResNet layers in residual blocks.

3. Suboptimal Gamma Value:
   The authors initialized the gamma value to zero, so the non-local features were not given any importance in the beginning.

4.2 Proposed Improvements

1. We increased the number of ResNet layers in Residual blocks from two to three to improve the accuracy and AUC. We also tried doubling the number of ResNet layers, but that resulted in overfitting, so we scaled it back to three.

Methodology	*AUC*	*Accuracy*	*Precision*	*Sensitivity*
No of layers=2	0.9456	93%	89.47%	0.8292
No. of layers=3	0.9631	94%	88.23%	0.

2. We replaced the ResNet layers with DenseNet layers and tried implementing it in the residual blocks but observed that AUC and accuracy have decreased, so we decided to stay with ResNet layers.

3. Changing the initial value of gamma to 0.25 helped us achieve better outputs in the earlier epochs, which would have taken a larger number of epochs otherwise, thereby reducing our training time and saving computational time and resources. The AUC and precision decreased very slightly, but the recall has a significant improvement from 82.92% to 90.24%.

Methodology	*AUC*	*Accuracy*	*Precision*	*Sensitivity*
Gamma=0	0.9456	93%	89.47%	0.8292
Gamma=0.25	0.9360	93%	88.09%	0.9024

4. References

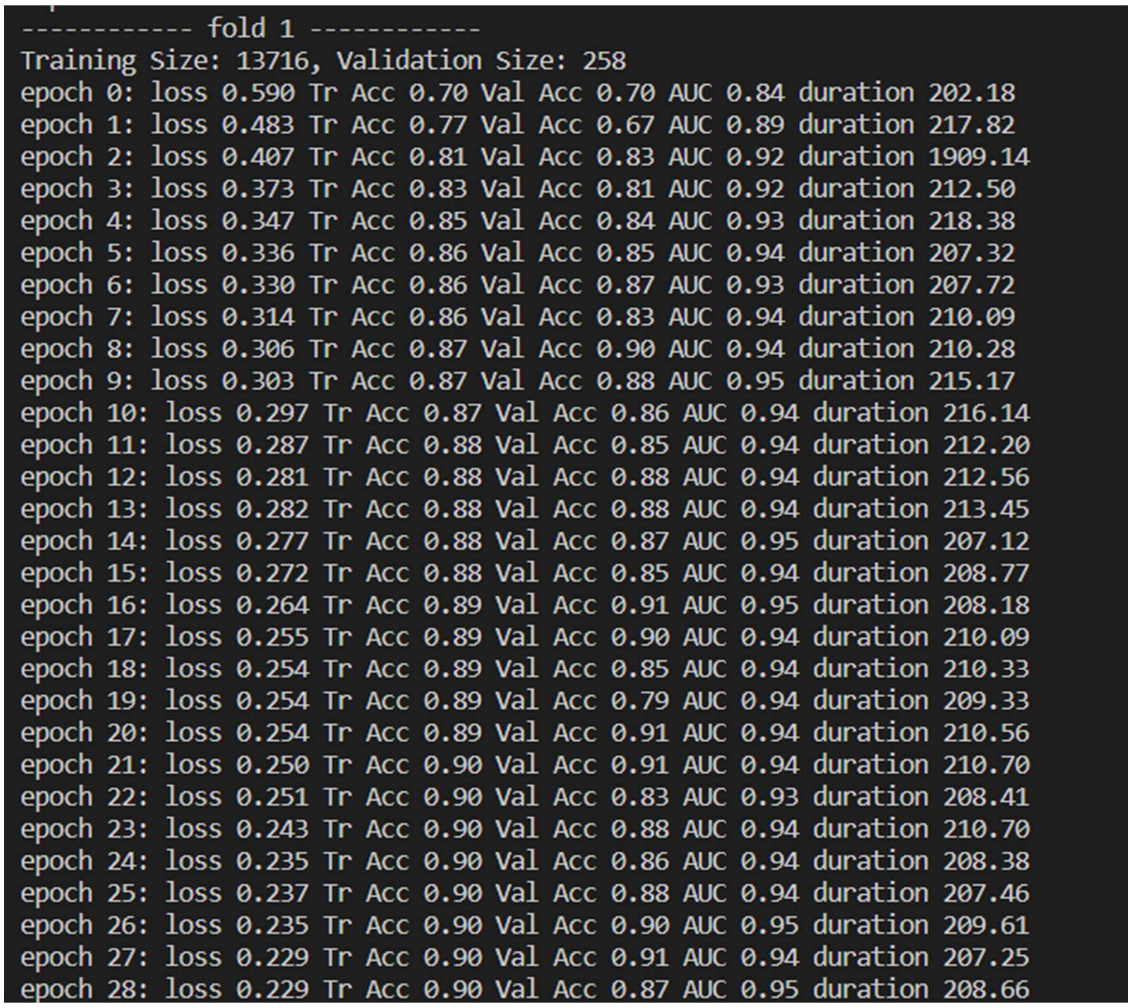
Bibliography

Kaiming He, X. Z. (n.d.). Deep Residual Learning for Image Recognition.

Mundher Al-Shabia, 1. B.-H. (n.d.). Lung Nodule Classification using Deep Local-Global Networks.

5. Appendix

Example of Implementation in our machines.



```
------------ fold 1 ------------
Training Size: 13716, Validation Size: 258
epoch 0: loss 0.590 Tr Acc 0.70 Val Acc 0.70 AUC 0.84 duration 202.18
epoch 1: loss 0.483 Tr Acc 0.77 Val Acc 0.67 AUC 0.89 duration 217.82
epoch 2: loss 0.407 Tr Acc 0.81 Val Acc 0.83 AUC 0.92 duration 1909.14
epoch 3: loss 0.373 Tr Acc 0.83 Val Acc 0.81 AUC 0.92 duration 212.50
epoch 4: loss 0.347 Tr Acc 0.85 Val Acc 0.84 AUC 0.93 duration 218.38
epoch 5: loss 0.336 Tr Acc 0.86 Val Acc 0.85 AUC 0.94 duration 207.32
epoch 6: loss 0.330 Tr Acc 0.86 Val Acc 0.87 AUC 0.93 duration 207.72
epoch 7: loss 0.314 Tr Acc 0.86 Val Acc 0.83 AUC 0.94 duration 210.09
epoch 8: loss 0.306 Tr Acc 0.87 Val Acc 0.90 AUC 0.94 duration 210.28
epoch 9: loss 0.303 Tr Acc 0.87 Val Acc 0.88 AUC 0.95 duration 215.17
epoch 10: loss 0.297 Tr Acc 0.87 Val Acc 0.86 AUC 0.94 duration 216.14
epoch 11: loss 0.287 Tr Acc 0.88 Val Acc 0.85 AUC 0.94 duration 212.20
epoch 12: loss 0.281 Tr Acc 0.88 Val Acc 0.88 AUC 0.94 duration 212.56
epoch 13: loss 0.282 Tr Acc 0.88 Val Acc 0.88 AUC 0.94 duration 213.45
epoch 14: loss 0.277 Tr Acc 0.88 Val Acc 0.87 AUC 0.95 duration 207.12
epoch 15: loss 0.272 Tr Acc 0.88 Val Acc 0.85 AUC 0.94 duration 208.77
epoch 16: loss 0.264 Tr Acc 0.89 Val Acc 0.91 AUC 0.95 duration 208.18
epoch 17: loss 0.255 Tr Acc 0.89 Val Acc 0.90 AUC 0.94 duration 210.09
epoch 18: loss 0.254 Tr Acc 0.89 Val Acc 0.85 AUC 0.94 duration 210.33
epoch 19: loss 0.254 Tr Acc 0.89 Val Acc 0.79 AUC 0.94 duration 209.33
epoch 20: loss 0.254 Tr Acc 0.89 Val Acc 0.91 AUC 0.94 duration 210.56
epoch 21: loss 0.250 Tr Acc 0.90 Val Acc 0.91 AUC 0.94 duration 210.70
epoch 22: loss 0.251 Tr Acc 0.90 Val Acc 0.83 AUC 0.93 duration 208.41
epoch 23: loss 0.243 Tr Acc 0.90 Val Acc 0.88 AUC 0.94 duration 210.70
epoch 24: loss 0.235 Tr Acc 0.90 Val Acc 0.86 AUC 0.94 duration 208.38
epoch 25: loss 0.237 Tr Acc 0.90 Val Acc 0.88 AUC 0.94 duration 207.46
epoch 26: loss 0.235 Tr Acc 0.90 Val Acc 0.90 AUC 0.95 duration 209.61
epoch 27: loss 0.229 Tr Acc 0.90 Val Acc 0.91 AUC 0.94 duration 207.25
epoch 28: loss 0.229 Tr Acc 0.90 Val Acc 0.87 AUC 0.95 duration 208.66
```