Diagnosis of Oral Cancer using Convolution Neural Network

Shivam Barot[1] Prakrut Suthar[2] Maharshi Prajapati[3] Kinjal Patel[4]
[1,2,3]Student [4]Assistant Professor
[1,2,3,4]Department of Computer Engineering
[1,2,3,4]L. J. Institute of Engineering & Technology, Ahmedabad, Gujarat, India

Abstract— Cancer has the highest growth rate among all diseases globally. Oral cancer is one of the most dangerous cancer which affects and originates from the oral cavity and neck. Overuse of tobacco and smoking cigarettes are the primary risk factor for developing oral cancer. In the present research, oral cancer patients can be identified through the use of deep learning algorithm for the diagnosis of the oral cancer. A deep learning based algorithm named CNN (Convolution Neural Network) is proposed in this paper which uses machine learning for the detection and identification of oral cancer.

Keywords: Oral Cancer, Convolution Neural Network, Machine Learning, Deep Learning

I. INTRODUCTION

Previously Cancer was an incurable disease, with innovation in technology, it has become curable if we can detect it in early stages. Oral cancer is the exponential and unstoppable increase in the number of cells or mutation and affects the surrounding tissues. Its detection is extremely important in the early stages else it can be fatal. It is caused by the following factors:

1) Smoking
2) Tobacco Consumption
3) Excessive consumption of alcohol
4) Human Papillomavirus (HPV)

Whereas the most common symptoms of Oral cancer are the following:

1) Swellings/thickenings, lumps or bumps, rough spots/crusts/or eroded areas on the lips, gums, or other areas inside the mouth
2) The development of velvety white, red, or speckled (white and red) patches in the mouth
3) Unexplained bleeding in the mouth
4) Unexplained numbness, loss of feeling, or pain/tenderness in any area of the face, mouth, or neck
5) Persistent sores on the face, neck, or mouth that bleed easily and do not heal within 2 weeks
6) A soreness or feeling that something is caught in the back of the throat
7) Difficulty chewing or swallowing, speaking, or moving the jaw or tongue
8) Hoarseness, chronic sore throat, or change in voice
9) Ear pain
10) A change in the way your teeth or dentures fit together
11) Dramatic weight loss

Treatment of Oral cancer is done by surgeries to remove the cancerous growth followed by chemotherapy and/or radiation therapy to destroy any remaining cancer cells.

We will get the image of the patch as an input and will apply it to our pre-trained model and predict whether it is of cancer's or not and will give to the user.

II. METHODOLOGY

In this section, we explain our approach in detail. We used Convolution Neural Network (CNN) for prediction. Though Dataset was not available on the web for images of cancer patients and non-cancer patches, we made our dataset of approximately 250 images per group by clicking images from various ENT hospitals and also verified the dataset with ENT doctors. We developed our trained model by training the dataset using CNN algorithm. The steps of training are the following:

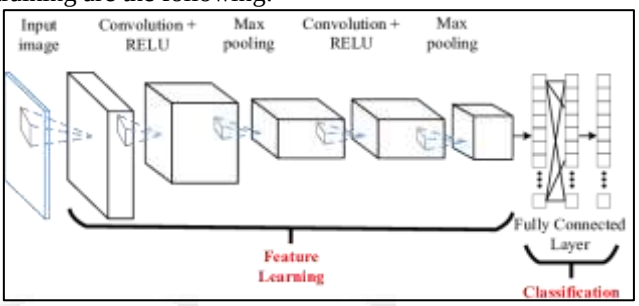

Fig. 1: Training of CNN Algorithm

A. Convolution:

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. We are giving dataset images as input to Convolution layer. In this layer, image matrix will multiply with filter matrix and will extract the matrix which is called "Feature Map". In each layer, there is a bank of m1 filters. The number of how many filters are applied in one stage is equivalent to the depth of the volume of output feature maps. Each filter detects a particular feature at every location on the input. The output ${Y_i}^{(l)}$ of layer l consists of ${m_1}^{(l)}$ feature maps of size ${m_2}^{(l)} \times {m_3}^{(l)}$. The i^{th} feature map, denoted${Y_i}^{(l)}$, is computed as

$$ {Y_i}^{(l)} = {B_i}^{(l)} + \sum_{j=1}^{{m_1}^{(l-1)}} {K_{i,j}}^{(l)} * {Y_j}^{(l-1)} $$

Where, ${B_i}^{(l)}$ is a bias matrix and ${K_{i,j}}^{(l)}$ is the filter of size $2{h_1}^{(l)} + 1 \times 2{h_2}^{(l)} + 1$ connecting the j^{th} feature map in layer $(l-1)$ with i^{th} feature map in layer.

The result of staging these convolutional layers in conjunction with the following layers is that the information of the image is classified like in vision. That means that the pixels are assembled into edglets, edglets into motifs, motifs into parts, parts into objects, and objects into scenes.

B. RELU:

ReLU stands for Rectified Linear Unit. This is sub-step of Convolution. We are using this to introduce non-linearity to

our model. The output of this layer is the following equation:

$$Y_i^{(l)} = \max(0, Y_i^{(l-1)})$$

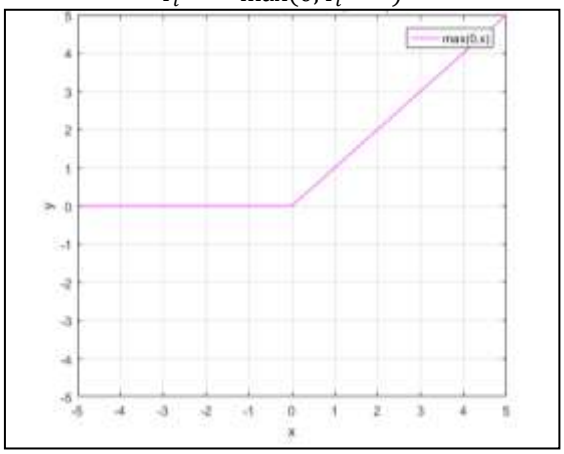

Fig. 2: Rectified Linear Unit (ReLU)

The rectified linear units come with three significant advantages in convolutional neural networks compared to the traditional logistic or hyperbolic tangent activation functions:

1) Rectified linear units propagate the gradient efficiently and therefore reduce the likelihood of a vanishing gradient problem that is common in deep neural architectures.
2) Rectified linear units threshold negative values to zero, and therefore solve the cancellation problem as well as result in a much more sparse activation volume at its output. The sparsity is useful for multiple reasons but mainly provides robustness to small changes in input such as noise.
3) Rectified linear units consist of only simple operations in terms of computation (mainly comparisons) and therefore much more efficient to implement in convolutional neural networks.
4) As a result of its advantages and performance, most of the recent architectures of convolutional neural networks utilize only rectified linear unit layers (or its derivatives such as noisy or leaky ReLUs) as their non-linearity layers instead of traditional non-linearity and rectification layers.

C. Pooling:

This layer is used to reduce the size of the input images of the dataset so we can speed-up the processing. We are using max pooling for this project which will take the largest element from the rectified feature map.

Here, we are using 8-layered CNN so above 2 steps will repeat for 8 times.

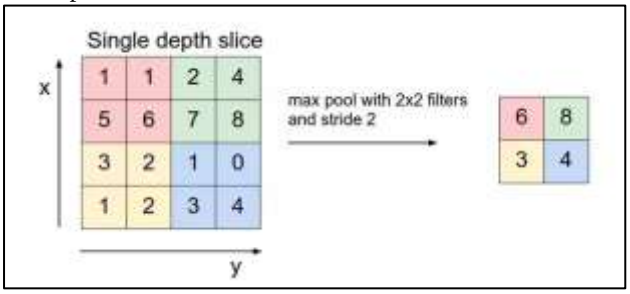

Fig. 3: Max pooling

The pooling layer l has two hyperparameters, the spatial extent of the filter $F^{(l)}$and the stride $S^{(l)}$. It takes an input volume of size ${m_1}^{(l-1)} \times {m_2}^{(l-1)} \times {m_3}^{(l-1)}$ and provides an output volume of size ${m_1}^{(l)} \times {m_2}^{(l)} \times {m_3}^{(l)}$ where;

$${m_1}^{(l)} = {m_1}^{(l-1)}$$
$${m_2}^{(l)} = \left({m_2}^{(l-1)} - F^{(l)}\right)S^{(l)} + 1$$
$${m_3}^{(l)} = \left({m_3}^{(l-1)} - F^{(l)}\right)S^{(l)} + 1$$

The key concept of the pooling layer is to provide translational invariance since particularly in image recognition tasks, the feature detection is more important compared to the feature's exact location. Therefore the pooling operation aims to preserve the detected features in a smaller representation and does so, by discarding less significant data at the cost of spatial resolution.

The pooling layer operates by defining a window of size $F^{(l)} \times F^{(l)}$ and reducing the data within this window to a single value. The window is moved by $S^{(l)}$positions after each operation similarly to the convolutional layer and the reduction is repeated at each position of the window until the entire activation volume is spatially reduced.

It is noteworthy that the window for pooling layers does not have to be a square and can be parametrized with ${F_1}^{(l)}$and ${F_2}^{(l)}$ resulting in a rectangle of size ${F_1}^{(l)} \times {F_2}^{(l)}$. However, this is extremely uncommon and is therefore left out of the notation is most of the publications, including this wiki.

The most common methods for reduction are max pooling and average pooling. Max pooling operates by finding the highest value within the window region and discarding the rest of the values. Average pooling on the other hand uses the mean of the values within the region instead.

D. Fully Connected Layer:

We will flatten our matrix into a vector and will feed it into a fully connected layer. Classification of problem and output is going to be generated in this layer.

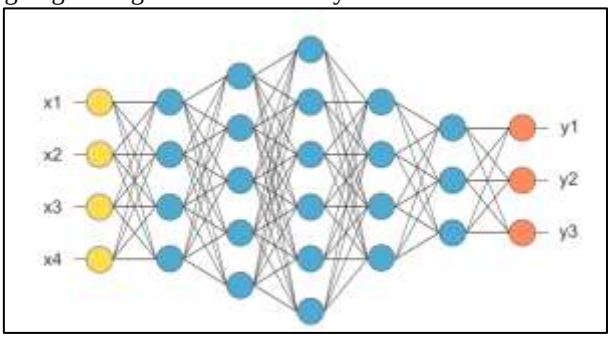

Fig. 4: After pooling layer, flattened as FC layer

The fully connected layers in a convolutional network are practically a multilayer perceptron (generally a two or three layer MLP) that aims to map the ${m_1}^{(l-1)} \times {m_2}^{(l-1)} \times {m_3}^{(l-1)}$ activation volume from the combination of previous different layers into a class probability distribution. Thus, the output layer of the multilayer perceptron will have ${m_1}^{(l-1)}$ outputs, i.e. output neurons where ii denotes the number of layers in the multilayer perceptron.

The key difference from a standard multilayer perceptron is the input layer where instead of a vector, an activation volume is taken as the input. As a result, the fully connected layer is defined as:
If $l-1$ is a fully connected layer;

$$y_i^{(l)} = f\left(z_i^{(l)}\right) with\ z_i^{(l)} = \sum_{j=1}^{m_1^{(l-1)}} w_{i,j}^{(l)}\, y_i^{(l-1)}$$

Otherwise;

$$y_i^{(l)} = f\left(z_i^{(l)}\right) with\ z_i^{(l)} = \sum_{j=1}^{m_1^{(l-1)}} \sum_{j=1}^{m_1^{(l-1)}} \sum_{j=1}^{m_1^{(l-1)}} w$$

The goal of the complete fully connected structure is to tune the weight parameters $w_{i,j}{}^{(l)}$or $w_{i,j,r,s}{}^{(l)}$ to create a stochastic likelihood representation of each class based on the activation maps generated by the concatenation of convolutional, non-linearity, rectification and pooling layers. Individual fully connected layers operate identically to the layers of the multilayer perceptron with the only exception being the input layer.

It is noteworthy that the function f once again represents the non-linearity, however, in a fully connected structure the non-linearity is built within the neurons and is not a separate layer.

III. Summary

The working of CNN is in the following steps:

1) Provide input image into convolution layer
2) Perform convolution on the image and apply ReLU activation to the matrix.
3) Perform pooling to reduce dimensionality size
4) Add convolutional layers
5) Flatten the output and feed into a fully connected layer (FC Layer)
6) Output the class using an activation function and classifies image.
7) Users should also consider consulting a doctor for more clarity.

IV. Conclusion

The task of predicting cancer using various deep learning algorithms is still in developing phase and far from as a viable option, As with our current trained model, we are achieving 92% of accuracy in predicting the result of whether the user has oral cancer or not. So, we are not recommending to use solely our project, but a better option is to use with the consultation of a doctor, combined. We are working towards increasing accuracy and committed to achieve that.

V. Future Enhancements

Increasing the accuracy of our project is the main focus. We are also planning to make this project available in the Application form on Android Play Store and Apple app store and also as a Web-App so we can reach out and help more people who don't have access to excellent healthcare.

Acknowledgment

Authors acknowledge the support provided by the Department of Computer Engineering, L.J Institute of Engineering & Technology, Ahmedabad, Gujarat, India and Gujarat Technological University, Ahmedabad, Gujarat.

References

[1] Yamashita, R., Nishio, M., Do, R.K.G. et al. Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9
[2] N. Aloysius, and M. Geetha, "A review on deep convolutional neural networks," Proceedings of the 2017 IEEE International Conference on Communication and Signal Processing, ICCSP 2017. pp. 588-592.
[3] A. Dhillon, and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," Progress in Artificial Intelligence, 2019/12/20, 2019.
[4] K. Lalithamani, and A. Punitha, "Detection of Oral Cancer Using Deep Neural Based Adaptive Fuzzy System in Data Mining Techniques", IJRTE 2019
[5] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions."
[6] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," arXiv preprint arXiv:1802.03268, 2018

