

RCCNet: An Efficient Convolutional Neural Network for Histological Routine Colon Cancer Nuclei Classification

Amila Sikalo
1938032

Sapienza Università di Roma

September 8, 2021

Abstract

This project is based on the paper dealing with the classification of histological routine colon cancer nuclei named „RCCNet: RCCNet: An Efficient Convolutional Neural Network for Histological Routine Colon Cancer Nuclei Classification” that is available at <https://paperswithcode.com/paper/rccnet-an-efficient-convolutional-neural>. The main objective of this paper is to develop the Convolutional Neural Network (CNN) model as simple as possible. Because of its potential uses in the realm of medical image analysis, efficient and exact categorization of histology cell nuclei is essential. It would make it much easier for doctors to comprehend and research many aspects of cancer treatment.

Due to cellular variability, classifying histology cell nuclei is a complex process. The proposed RCCNet model consists of 3,869,764 learnable parameters which are significantly less compared to the popular CNN models such as AlexNet with 58,325,066 learnable parameters. In this paper, these two approaches were implemented and compared.

1. Introduction

One of the most popular and widely used Deep Learning networks is the Convolutional Neural Network (CNN). CNN has made Deep Learning increasingly popular in recent years. CNN's primary benefit over its predecessors is that it automatically finds significant characteristics without the need for human intervention, making it the most widely utilized. Several CNN designs have been proposed in the last ten years. The architecture of a model is an important component in enhancing the performance of many applications. From 1989 until the present, the CNN architecture has undergone number of changes. Structure reformulation, regularization, parameter optimizations, and other changes are examples of such modifications. On the other hand, it should be emphasized that the major improvement in CNN performance was primarily

attributed to the rearrangement of processing units and the development of new blocks. The utilization of network depth was used to conduct the most novel advances in CNN designs [1].

Furthermore, data augmentation strategies were introduced to enhance the amount of available data while avoiding the overfitting problem. These methods are data-space solutions for any problem with little data. Data augmentation is a term that refers to a range of techniques for improving the characteristics and size of training datasets. The following data augmentation techniques are mostly used in the literature: Horizontal and Vertical Shift Augmentation, Horizontal and Vertical Flip Augmentation, Random Rotation Augmentation, Random Brightness Augmentation and Random Zoom Augmentation.

The rest of the paper is organized as follows. In the second section, the dataset containing the images of histological routine colon cancer nuclei patches is explained. In the next two sections, the algorithm and overall CNN architectures are

introduced. The results are presented in the fifth section, and the paper concludes with sixth section.

2. Dataset

I utilized a publicly available ‘[CRCHistoPhenotypes](#)’ dataset, which contains histological routine colon cancer nuclei patches, to determine the performance of the proposed RCCNet for the job. In the original paper, authors state that there are 22444 nuclei patches in this dataset, divided into four classes: epithelial, inflammatory, fibroblast, and miscellaneous. Every class is represented in the table below.


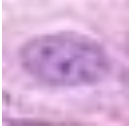


	
Epithelial	Inflammatory
	
Fibroblast	Miscellaneous

Figure 1: Represented Data

It is also reported that there are 7722 patches in the ‘Epithelial’ category, 5712 patches in the ‘Fibroblast’ category, 6971 patches in the ‘Inflammatory’ category, and 2039 patches in the ‘Miscellaneous’ category. Each patch is $32 \times 32 \times 3$ inches in size. However, Github repository contains only 14 images for each of the categories, meaning that the result of the model training will not be valid.

3. CNN TRAINING ALGORITHM

During the training phase the categorical cross entropy loss is computed and the parameters

(weights) of the network are updated by computing the gradient of parameters with respect to the loss function [2]. The cross-entropy loss is used to compute the performance of a classifier whose output is a probability of an input belonging to a certain class. Let n be a three-dimensional input image to the network with class label $c \in C$ where

$$C = \{c_1, c_2, \dots, c_n\}$$

is the set of class labels. In the current classification task, $C = \{0, 1, 2, 3\}$. The output of the network is a vector y which is

$$y = f(x)$$

where f denotes the forward pass computation function and $y = [y_1, y_2, \dots, y_{cn}]$ represents the class scores for the n classes. The cross-entropy loss for n if the target class (as given in the training set) is c_i , is given as:

$$L = -\log \left(\frac{e^{y_{c_i}}}{\sum_{k=c_1}^{c_n} e^{y_k}} \right)$$

The total loss over a mini-batch of training examples is considered in the training process. This is by default in TensorFlow 2.x. At test time, for a given input image, the class label having the highest score is the predicted class label. This predicted class label c_j is computed as:

$$c_j = \arg \max_i p(y_i)$$

where $p(y_i)$ is the probability that x belongs to class c_j , which is computed as:

$$p(y_i) = \frac{e^{y_i}}{\sum_{k=c_1}^{c_n} e^{y_k}}$$

4. CNN ARCHITECTURE

In the proposed architecture, the histological images of dimension $32 \times 32 \times 3$ as input to the network are considered. This CNN model has three blocks with seven trainable layers:

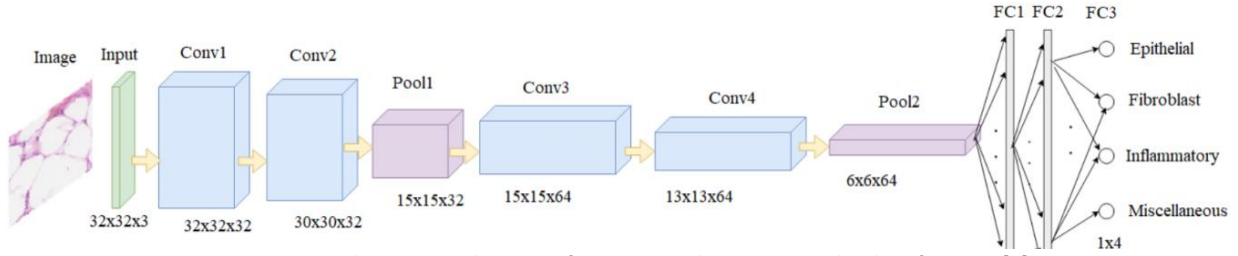


Figure 2: Proposed RCCNet architecture for routine colon cancer nuclei classification. [2]

- First block: two convolutional layers, Conv1 and Conv2 are used just after the input layer. The Conv2 layer is followed by a pooling layer (Pool1) to reduce the spatial dimension by half.
- Second block: two convolutional layers (i.e., Conv3 and Conv4 layers) are followed by another pooling layer (Pool2).
- Third block: three fully connected layers, namely FC1, FC2, and FC3 are used in the proposed architecture.

The first convolutional layer Conv1 produces a $32 \times 32 \times 32$ -dimensional feature map by convolving 32 filters of dimension $3 \times 3 \times 3$. The zero padding by 1 pixel in each direction is done in Conv1 layer to retain the same spatial dimensional feature map.

The Conv2 layer has the 32 filters of dimension $3 \times 3 \times 32$ with no padding which produces a $30 \times 30 \times 32$ -dimensional feature map. The stride is set to 1 in both Conv1 and Conv2 layers. In Pool1 layer, the sub-sampling with the receptive field of 2×2 is applied with a stride of 2 and without padding which results in feature map of size $15 \times 15 \times 32$.

The Conv3 layer produces 64 feature maps of spatial dimension 15×15 (i.e., spatial dimension is retained by applying zero padding with a factor of 1), which is obtained by applying 64 filters of dimension $3 \times 3 \times 32$ with a stride of 1. Similar to Conv2 layer, Conv4 layer also does not apply padding and uses stride of 1. The Conv4 layer

produces 64 feature maps of dimension 13×13 , obtained by convolving the 64 filters of size $3 \times 3 \times 64$. The second sub-sampling layer Pool2 also uses the kernel size of 2×2 with a stride of 2, which results in a $6 \times 6 \times 64$ -dimensional feature map. The right and bottom border feature values of input are not considered in Pool2 layer to get rid of dimension mismatch between input and kernel size. The feature map generated by Pool2 layer is flattened into a single feature vector of length 2304 before third block. So, the input to FC1 layer is 2304-dimensional feature vector and output is 512-dimensional feature vector. Both input and output to FC2 layer is 512-dimensional feature vectors. The last fully connected layer FC3 takes the input of dimension 512 and produces the 4 values as the output corresponding to the scores for 4 classes. This architecture consists of 3,869,764 trainable parameters from 7 trainable layers (i.e., Conv1, Conv2, Conv3, Conv4, FC1, FC2, and FC3 layers). On top of the last fully connected layer FC3 of proposed RCCNet model, a 'softmax classifier' for multi-class classification is used to generate the probabilities for each class. The probabilities generated by the 'softmax classifier' is further used to compute the loss during training phase and to find the predicted class during testing phase [2].

AlexNet is a convolutional neural network that has had a significant influence on machine learning, particularly in the application of deep learning to machine vision. Convolutions, max pooling, dropout, data augmentation, ReLU activations, and SGD with momentum were all

part of it. After each convolutional and fully connected layer, it added ReLU activations.

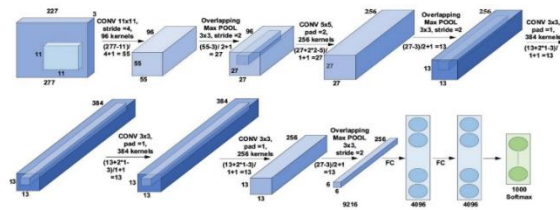


Figure 3 - AlexNet architecture

5. RESULTS

Both RCCNet and AlexNet were implemented within Google Colab environment providing free online Jupyter notebook setup.

For the RCCNet, the following network architecture is obtained:

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 28, 28, 96)	7296
batch_normalization_15 (Batch Normalization)	(None, 28, 28, 96)	384
conv2d_7 (Conv2D)	(None, 24, 24, 256)	614656
batch_normalization_16 (Batch Normalization)	(None, 24, 24, 256)	1024
block1_maxpooling2 (MaxPooling2D)	(None, 12, 12, 256)	0
block1_conv3 (Conv2D)	(None, 12, 12, 384)	885120
batch_normalization_17 (Batch Normalization)	(None, 12, 12, 384)	1536
block1_conv4 (Conv2D)	(None, 12, 12, 384)	1327488
batch_normalization_18 (Batch Normalization)	(None, 12, 12, 384)	1536
block1_conv5 (Conv2D)	(None, 12, 12, 256)	884992
batch_normalization_19 (Batch Normalization)	(None, 12, 12, 256)	1024
dropout_2 (Dropout)	(None, 12, 12, 256)	0
flatten (Flatten)	(None, 36864)	0
predictions (Dense)	(None, 4)	147460
Total params: 3,872,516		
Trainable params: 3,869,764		
Non-trainable params: 2,752		

Figure 4 - Model

After training and 10 epochs, the results were as given in the accuracy over epochs plot. The final accuracy of the training set is 0.8636.

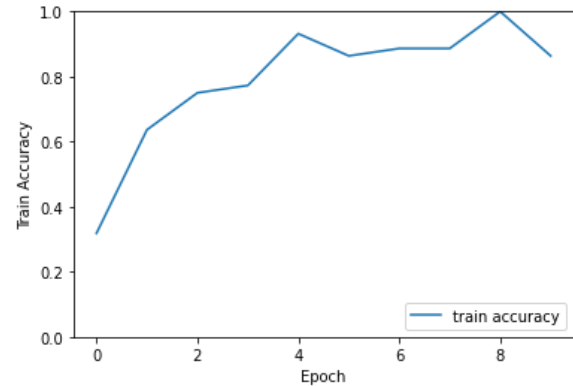


Figure 5 - RCCNet Train Accuracy

The classification report (from scikit-learn library) yields a 25% accuracy with test data.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	3
1	0.00	0.00	0.00	3
2	0.25	1.00	0.40	3
3	0.00	0.00	0.00	3
accuracy			0.25	12
macro avg	0.06	0.25	0.10	12
weighted avg	0.06	0.25	0.10	12

Figure 6 - RCCNet Classification Report

As the dataset provided with the Github repository contains only 56 images divided into four of aforementioned classes, this result is expected. In order to artificially increase the dataset, I used the technique called data augmentation, where different transformations of the images were added to the dataset.

Authors of the original paper artificially increase the dataset by introducing the cifar10 dataset and concatenating it to the provided images of the histological routine colon cancer images. I have observed, although with limited training, due to limited hardware resources, that these cifar10 images do not increase the overall training performance with such a small dataset.

The following data augmentation techniques were performed

Horizontal and Vertical Shift Augmentation

A shift to a picture implies shifting all of the image's pixels in one direction, such as horizontally or vertically, while maintaining the image's dimensions. This implies that certain pixels will be clipped from the picture, and new pixel values will need to be given in a portion of the image.

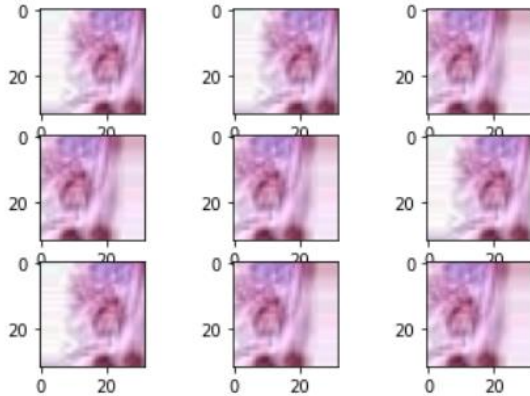


Figure 7 – Horizontal Shift Augmentation

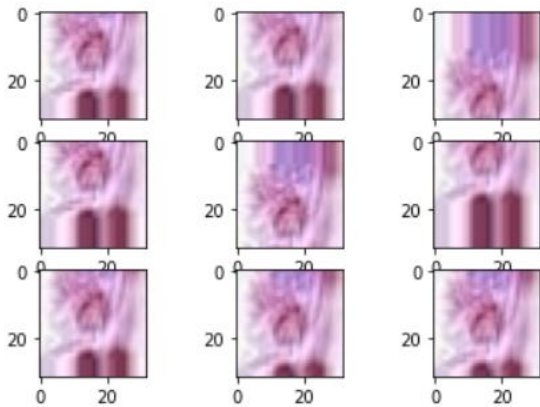


Figure 8 – Vertical Shift Augmentation

Horizontal and Vertical Flip Augmentation

In the case of a vertical or horizontal flip, an image flip involves reversing the rows or columns of pixels.

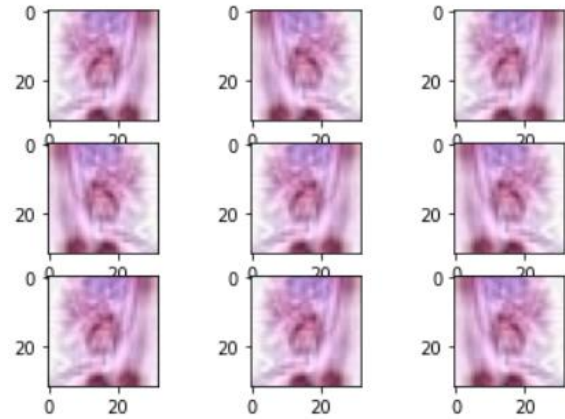


Figure 9 – Horizontal and Vertical Flip Augmentation

Random Rotation Augmentation

A rotation augmentation turns the picture clockwise by a specified number of degrees between 0 and 360 at random. The rotation will most likely rotate pixels out of the image frame, leaving blank spaces in the frame that must be filled in.

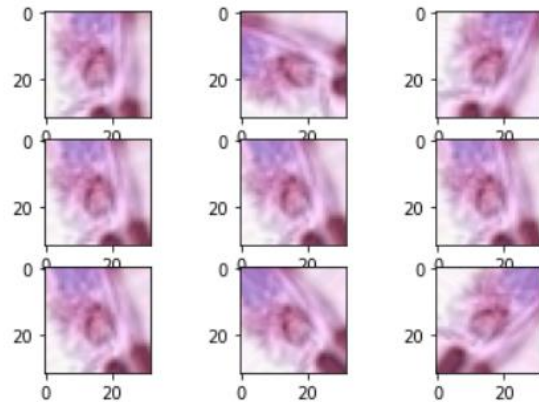


Figure 10 – Random Rotation Augmentation

Random Brightness Augmentation

Another approach is to randomly change the images' brightness. The goal is for a model to be able to generalize across photos with varied illumination conditions.

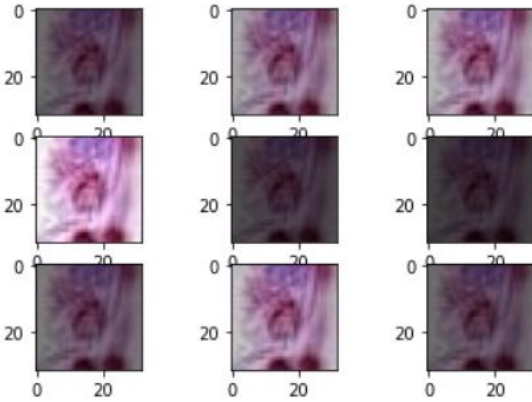


Figure 11 – Random Brightness Augmentation

Random Zoom Augmentation

A zoom augmentation enlarges the image at random and either adds additional pixel values to the image or interpolates pixel values.

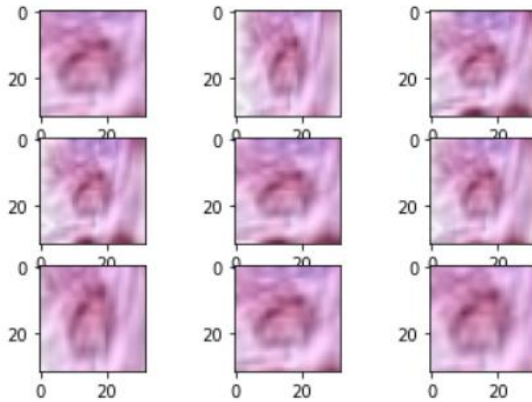


Figure 12 – Random Zoom Augmentation

AlexNet

	precision	recall	f1-score	support
0	0.00	0.00	0.00	3
1	0.25	1.00	0.40	3
2	0.00	0.00	0.00	3
3	0.00	0.00	0.00	3
accuracy			0.25	12
macro avg	0.06	0.25	0.10	12
weighted avg	0.06	0.25	0.10	12

Figure 13 - AlexNet Classification Report

AlexNet is a network architecture specifically designed for large image datasets that are well defined in terms of semantic contents. This dataset is neither - it is rather small and the semantic differences are discriminable to humans, i.e. medical professionals. Therefore, results for AlexNet training performance are not satisfactory.

However, the same problem can be addressed for the proposed RCCNET. The poor performance of the dataset is yielding very bad training performance. Data augmentation seems not to affect it in this case due to the similarity of the images in different classes (color contents, brightness, texture etc.)

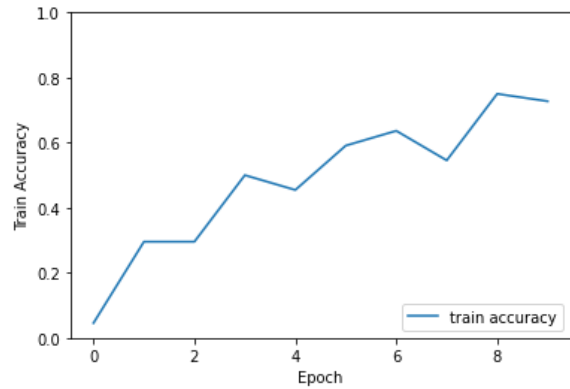


Figure 14 – AlexNet Train Accuracy

It can be observed that the proposed RCCNet architecture, although much simpler, provides better results with both the original dataset with 56 images and with augmented dataset than AlexNet. Furthermore, both networks yield better training accuracy with augmented dataset, as expected with larger datasets, i.e. more input information. Adam NN optimization algorithm is used for both networks with the default TensorFlow Adam parameters and, as expected, the training of the RCCNet is quicker due to its simpler structure.

6. CONCLUSION

This paper presented two CNN architectures for classifying histological routine colon cancer images with four different classes.

The provided dataset was insufficient for proper analysis and CNN modelling due to low number of image – only 56 were present in the Github repository.

Both networks suffer because of this in a sense that they cannot generalize well with low amount of input information. Data augmentation techniques somewhat reduce this problem, but it is still insufficient for any meaningful results.

More data should be provided for better classifier modelling and analysis.

7. Bibliography

- [1] e. a. Laith Alzubaidi, "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," *Journal of Big Data*, 2021.
- [2] S. H. S. B. e. al., "RCCNet: An Efficient Convolutional Neural Network for Histological Routine Colon Cancer Nuclei Classification," in *International Conference on Control, Automation, Robotics and Vision*, Singapore, 2018.