Detection of Acute Lymphocytic Leukemia with Convolutional Neural Network

Ragi R Unnithan
MS. Computer Science
Hood College
Frederick, MD, USA
ru3@hood.edu

ABSTRACT

Acute Lymphocytic Leukemia (ALL) is a deadly cancer that not only affects adults but also accounts for 25% of childhood cancers. Timely and accurate diagnosis of the cancer is an important factor for an effective treatment to improve survival rate. The image of ALL cells under the microscope is very similar in morphology to that of the normal cells. In this work, we are trying to answer whether a Convolutional Neural Network (CNN) can reliably classify cancer and normal cell images to assist in the diagnosis of ALL. This project aims to build a CNN model that can classify ALL from normal cells as diagnosing ALL is difficult even for trained medical operator. The network is developed in Python framework using libraries Tensorflow and Keras.

CCS CONCEPTS

•Deep Learning •Convolutional Neural Network •Image Classification

KEYWORDS

Deep Learning, Convolutional Neural Network, Feature Extraction, Deep Neural Network, Image Augmentation, Batch Normalization

1 Introduction

Leukemia is a hematological disorder and type of blood cancer that weakens the human immune system by generating a surplus number of malignant WBCs. Leukemia is classified into four major classes: Acute Myeloblastic Leukemia (AML), Acute Lymphoblastic Leukemia (ALL), Chronic Myeloblastic Leukemia (CML), and Chronic Lymphoblastic Leukemia (CLL).

ALL, which typically affects the blood and bone marrow, is the most common type of cancer in childhood but can also occur in adults. It is caused due to the increased lymphocytes in the human body. It is acute because the disease progresses rapidly and creates immature blood cells, rather than mature ones. The word “lymphocytic” in acute lymphocytic leukemia refers to the white blood cells called lymphocytes, and ALL affects these white blood cells. Usually, manual evaluation of cells is used by a hematology analyzer to diagnose ALL.

ALL cell classification depends on the morphological characteristics of the cells and requires a skilled medical operator. This procedure is time consuming, tedious, costly and error prone. Moreover, there are three different subtypes of ALL, which makes the detection of ALL extremely challenging. L1 are small and homogeneous, round with no cleft-ing, and with no obvious nucleoli or vacuoles. L2 are larger and heterogeneous, irregular shape and often clefted, and have defined nuclei and vacuoles. L3 have the shape of L1 but have prominent nucleoli and vacuoles. Given the challenges associated with manual identification of ALL, a machine learning (ML) algorithm will help to identify the blood cells with leukemia from the healthy cells, given a large training data set is available for the process. Figure 1. shows the morphology of a normal cell and an ALL cell, and is evident that normal cells are spherical and non-clefted whereas ALL cells are irregularly shaped and clefted.

Deep learning, a type of machine learning, and is very powerful and versatile tool that could be used to identify large ALL cells. These algorithms are used efficiently in significant research areas like medical image processing, supercomputing, investment modelling, and fraud detections. Convolutional Neural Network (CNN) is a popular subcategory of deep learning algorithm, specially designed for visual pattern recognition. In this project, we aim to build and train a CNN that can classify normal and ALL cells in microscopic blood images.

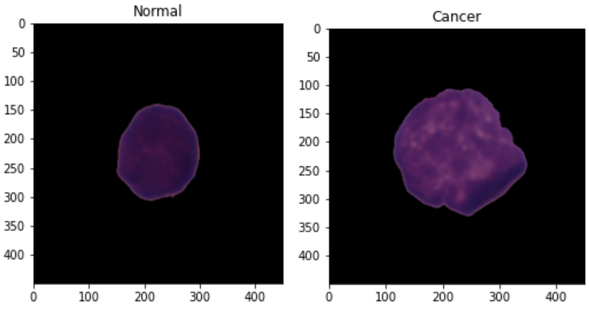


Figure 1: Normal (left) vs. cancer (right) cell

2 Materials and Methods

This section represents, the details about the dataset we used, data augmentation and finally applying CNN to classify ALL.

2.1 Dataset

Dataset: In this project, we used the dataset that was obtained from Kaggle, named C_NMC_Leukemia. The images from this dataset were segmented from microscopic images and there is a total of

15,114 images from 118 patients with two labelled classes: Normal cell and Leukemia blast. Images are 450x450 RGB stored as .bmp files, a raster graphic bitmaps which stores images as 2D matrices.

From the 118 patients, 69 have Acute Lymphocytic Leukemia (ALL) and 49 are normal patients. This dataset was separated for training, testing, and validation. The training dataset contains images from 73 patients, in which 47 is labelled as ALL and 26 as normal. In the validation dataset, we have 13 ALL patients and 15 normal patients from a total of 28 patients. The test dataset comprises of 17 patients, 9 ALL and 8 normal. Table 1 depicts the number of samples in training, validation, and test dataset.

Table 1. Datatset

Dataset – subtype	Total	Normal	ALL
Training	10661	3389	7272
Validation	1867	1219	648
Test	2586	-	-

2.2 Exploratory Data Analysis

Exploratory data analysis was performed on the dataset, to understand more about the data. Bar plots and pie charts were generated for better understanding of the data distribution. Figure 2 shows the number of normal and ALL cell in the dataset. This is a highly unbalanced dataset.

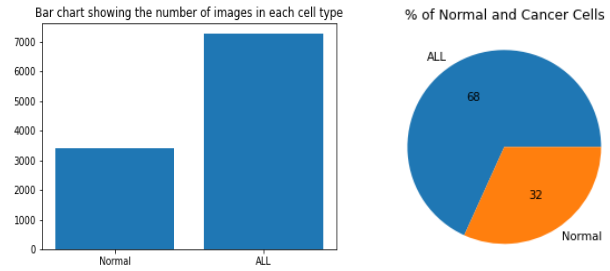


Figure 2: Number of images in each cell type, and % of normal and cancer cells

2.3 Dataset Split

While the training and the validation datasets are labelled, whereas the test dataset is not properly labelled. So, we decided to split the 10661 images in the training directory into 25% for validation and 75% for training per each cell type. We will be using the validation dataset for testing the prediction accuracy for evaluating our model performance.

Table 2.

Dataset - subtype	Total	Normal	ALL
Training	7996	2542	5454
Validation	2665	847	1818
Test	1867	1219	648

2.4 Dataset Preprocessing

Images are in pixel format ranging from 0 to 255. The images are rescaled between ranges 0 to 1, to treat all images in the same manner. It also helps the neural network to have a higher chance of converging.

2.5 Convolutional Neural Network from Scratch

CNN is composed of convolutional layers, activation function, pooling layers, and fully connected layers. In the CNN classification models, the convolutional layer, the activation function, and the pooling layer constitute the feature extraction layer to extract the features, while the full connection layer forms a classification layer for classification. The pooling layer is a down-sampling operation to reduce the dimensionality of the extracted features while retaining important information of the features. The convolutional layer is the core structure of CNN.

Convolutional layers are preferred over the multi-layer perceptron (MLP) as MLP uses one perceptron per input. So, for image processing, it will be using one perceptron for each pixel and if it is a color image then it will be multiplied by 3 (RGB). Another issue is that MLP react differently to an input image, and its shifted version. Thus, MLP is not the best network for image classification.

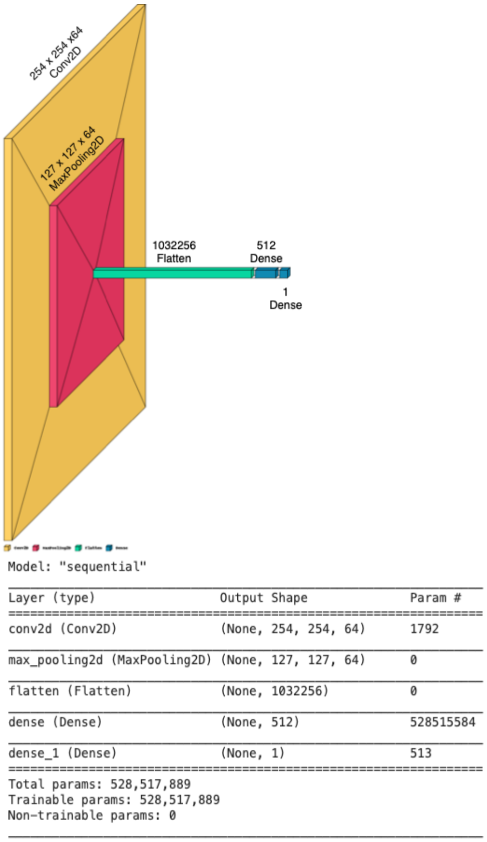
In CNN, filters are used for detecting each feature. A two-dimensional filter will be moved through the entire image from top left to bottom right and at each point in the image using convolution operation, a value is calculated. Once the filter has passed through the image, each filter generates a feature map, which is then taken through an activation function. This activation functions decide whether a certain feature is present in each location in each image. In pooling layer, we can apply different pooling techniques, but the most common is max-pooling and this is what is applied in this project, and it takes the maximum value in a certain filter region. The fully connected layer flattens the result before the classification.

In this project, first a simple vanilla model with a single Conv2d layer and a single dense layer is implemented. In the convolutional layer, 64 filters were implemented for detecting the features. The filter size is 3x3. The convolutional layer uses relu activation function. Following the convolutional layer is a max-pooling layer with a 2x2 pooling window. Following the max-pool layer is a flatten layer and then two fully connected dense layer. The last dense layer uses a sigmoid activation function with an output size of 1.

This model has an accuracy of 60% and has a recall of 16%. Accuracy, loss and recall are used as metrics in selecting the best model. Recall is also included because outcome of false negative is much worse for the patient than false positive. The architecture and summary of the vanilla model is shown below in Figure 3.

Successive larger layers and more complex architectures are constructed using by adding additional Conv2D and max-pooling layers. The most complex model has five convolutions in 3 blocks of 3 layer and this model try to overfit the training data. Dropout layers and batch normalization were also implemented to see whether it improved the model's performance.

Binary cross-entropy is used for the loss function as this is a binary classification problem. We tried both RMSProp and Adam for optimization.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 254, 254, 64)      1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 127, 127, 64)      0
_________________________________________________________________
flatten (Flatten)            (None, 1032256)           0
_________________________________________________________________
dense (Dense)                (None, 512)               528515584
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 528,517,889
Trainable params: 528,517,889
Non-trainable params: 0
_________________________________________________________________
```

Figure 3: CNN architecture and summary of 1C1D

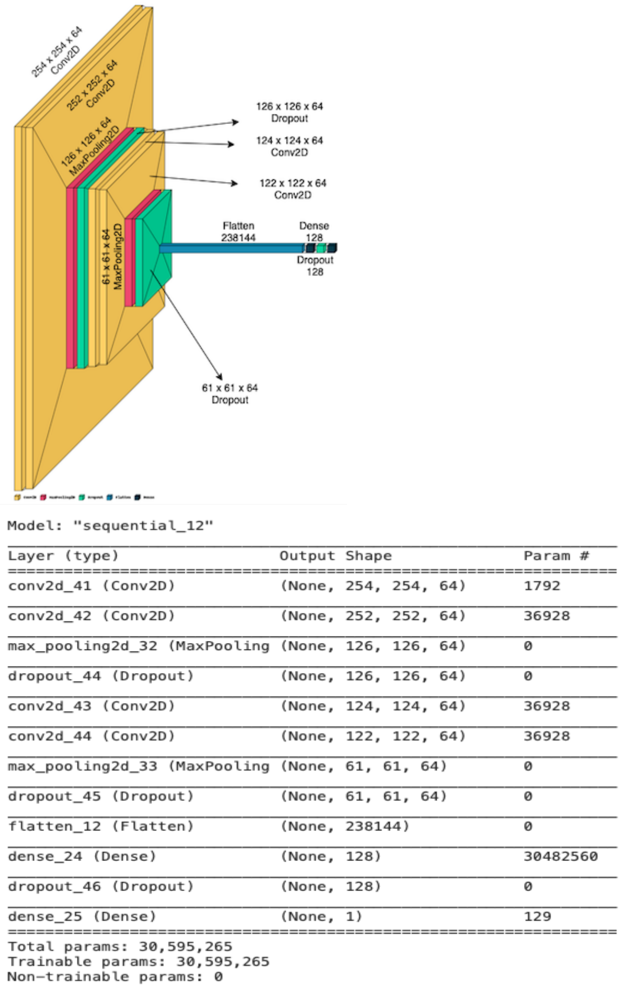
Out of all the models that we developed the best model is the 2x2C1D model with a dropout of 0.5. This model had a total of 4 convolutions and two dense layers. Figure 4 shows the architecture and summary of the 2x2C1D- Dropout (0.5) model.

In the chosen model, the input layer gets a 256 x 256 x 3 input, in which 256 x 256 is the image size with 3 channels. There is a total of 4 convolutional layers, the first and the second convolutional layer has 64 filters of size 3x3. Both uses relu as the activation function. This layer is followed by a max-pooling layer which uses a 2x2 pooling window. Following the max-pooling layer is a Dropout layer with a value of 0.5 to combat overfitting.

Following the Dropout layer is another 2 convolutional layers with the 64 filters of size 3x3. These layers also use relu as the activation function. Then another max-pooling layer is introduced with a pooling window of 2x2 followed by a Dropout layer with the same value of 0.5. Then a flatten layer, followed by a dense layer is added. A dropout of size 0.5 is introduced after that. The final dense layer uses a softmax activation function with an output size of 1.

3 Results

Table 3 shows the summary of the results we obtained from each model. Several models were created with multiple layers with and without Batch Normalization and Dropout. Different learning rates were also included. The primary metric used to evaluate the model is "Accuracy". The secondary metric used to pick the best model is "Recall". For the model "1X1C-1D - ZeroPadding and DropOut", the recall is 1. But we can't use this model as the accuracy is low. With accuracy sacrificed the mode will be of no use.

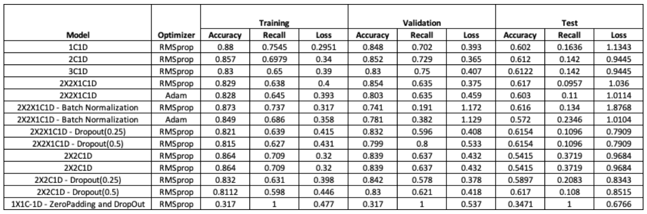


```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_41 (Conv2D)           (None, 254, 254, 64)      1792
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 252, 252, 64)      36928
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 126, 126, 64)      0
_________________________________________________________________
dropout_44 (Dropout)         (None, 126, 126, 64)      0
_________________________________________________________________
conv2d_43 (Conv2D)           (None, 124, 124, 64)      36928
_________________________________________________________________
conv2d_44 (Conv2D)           (None, 122, 122, 64)      36928
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 61, 61, 64)        0
_________________________________________________________________
dropout_45 (Dropout)         (None, 61, 61, 64)        0
_________________________________________________________________
flatten_12 (Flatten)         (None, 238144)            0
_________________________________________________________________
dense_24 (Dense)             (None, 128)               30482560
_________________________________________________________________
dropout_46 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_25 (Dense)             (None, 1)                 129
=================================================================
Total params: 30,595,265
Trainable params: 30,595,265
Non-trainable params: 0
```

Figure 4: CNN architecture and summary of 2x2C1D- Dropout (0.5)

The model 2x2C1D- Dropout (0.5) was able to achieve a training accuracy of 94% and validation accuracy of 85% with testing accuracy of 61% with 100 epochs. Figures 5 and 6 shows the "loss" and "Accuracy" of 2x2C1D- Dropout (0.5) model. We can conclude that CNN can be used for classification of ALL cells from normal cells and can be used as a tool by a physician for verifying his diagnosis.

Table 3.



		Training			Validation			Test		
Model	Optimizer	Accuracy	Recall	Loss	Accuracy	Recall	Loss	Accuracy	Recall	Loss
1C1D	RMSprop	0.88	0.7545	0.2951	0.848	0.702	0.393	0.602	0.1636	1.1343
2C1D	RMSprop	0.857	0.6979	0.34	0.852	0.729	0.365	0.612	0.142	0.9445
3C1D	RMSprop	0.83	0.65	0.39	0.83	0.75	0.407	0.6122	0.142	0.9445
2X2X1C1D	RMSprop	0.829	0.638	0.4	0.854	0.635	0.375	0.617	0.0957	1.036
2X2X1C1D	Adam	0.828	0.645	0.393	0.803	0.635	0.459	0.603	0.11	1.0114
2X2X1C1D - Batch Normalization	RMSprop	0.873	0.737	0.317	0.741	0.191	1.172	0.616	0.134	1.8768
2X2X1C1D - Batch Normalization	Adam	0.849	0.686	0.358	0.781	0.382	1.129	0.572	0.2346	1.0104
2X2X1C1D - Dropout(0.25)	RMSprop	0.821	0.639	0.415	0.832	0.596	0.408	0.6154	0.1096	0.7909
2X2X1C1D - Dropout(0.5)	RMSprop	0.815	0.627	0.431	0.799	0.8	0.533	0.6154	0.1096	0.7909
2X2C1D	RMSprop	0.864	0.709	0.32	0.839	0.637	0.432	0.5415	0.3719	0.9684
2X2C1D	RMSprop	0.864	0.709	0.32	0.839	0.637	0.432	0.5415	0.3719	0.9684
2X2C1D - Dropout(0.25)	RMSprop	0.832	0.631	0.398	0.842	0.578	0.378	0.5897	0.2083	0.8343
2X2C1D - Dropout(0.5)	RMSprop	0.8112	0.598	0.446	0.83	0.621	0.418	0.617	0.108	0.8515
1X1C-1D - ZeroPadding and DropOut	RMSprop	0.317	1	0.477	0.317	1	0.537	0.3471	1	0.6766

4 Conclusions

The performance of the model can be improved by implementing different methods. In this project, we were able to improve the performance of the model by increasing the number of epochs to 100. We also tried two different optimizers – RMSprop and Adam optimizer. But we see increase in performance when using RMSprop. Batch Normalization and Dropout were also implemented to improve overfitting. We experimented with different dropout values ranging from 0.1 to 0.5 and we settled on 0.5 as this increased the accuracy. We could implement early stopping and cross-validation for avoiding overfitting. Implementing early stopping will allow the model to stop training once a threshold of overfitting has been reached. Image data augmentation can also be introduced to increase the data sample. Here in this project, the class imbalance is kept for verifying whether the model can handle imbalance.

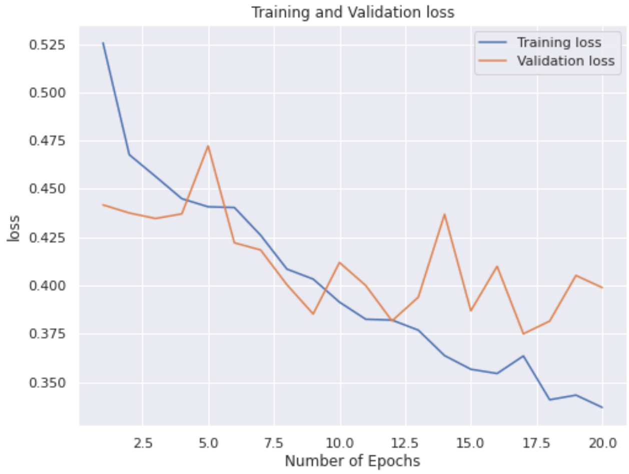


Figure 5: Training and validation loss

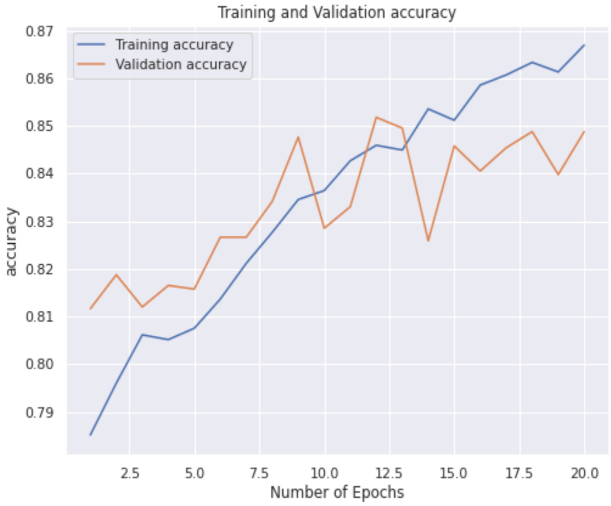


Figure 6: Training and validation accuracy

ACKNOWLEDGMENTS

I would like to express sincere thanks to Professor Dr. Xinlian Liu for this course and guidance throughout the semester as it helped me in learning various unknown topics related to deep learning, image classification.

REFERENCES

1. Ahmed, Yigit, Isik, and Alpkocak. “Identification of Leukemia Subtypes from Microscopic Images Using Convolutional Neural Network.” *Diagnostics* 9, no. 3 (August 25, 2019): 104. https://doi.org/10.3390/diagnostics9030104.
2. Mondal, Chayan, Md. Kamrul Hasan, Md. Tasnim Jawad, Aishwariya Dutta, Md. Rabiul Islam, Md. Abdul Awal, Mohiuddin Ahmad, Salem A. Alyami, and Mohammad Ali Moni. “Acute Lymphoblastic Leukemia Detection from Microscopic Images Using Weighted Ensemble of Convolutional Neural Networks.” Preprint. ENGINEERING, May 19, 2021. https://doi.org/10.20944/preprints202105.0429.v1.
3. Shaheen, Maneela, Rafiullah Khan, R. R. Biswal, Mohib Ullah, Atif Khan, M. Irfan Uddin, Mahdi Zareei, and Abdul Waheed. “Acute Myeloid Leukemia (AML) Detection Using AlexNet Model.” Edited by Dan Selisteanu. *Complexity* 2021 (May 28, 2021): 6658192. https://doi.org/10.1155/2021/6658192.
4. https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac
5. https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
6. https://towardsdatascience.com/detecting-leukemia-with-a-cnn-af699b19ab99
7. Jiang, Zhencun, Zhengxin Dong, Lingyang Wang, and Wenping Jiang. “Method for Diagnosis of Acute Lymphoblastic Leukemia Based on ViT-CNN Ensemble Model.” Edited by Suresh Manic. *Computational Intelligence and Neuroscience* 2021 (August 21, 2021): 1–12. https://doi.org/10.1155/2021/7529893.

8. Ttp, Thanh, Giao N. Pham, Jin-Hyeok Park, Kwang-Seok Moon, Suk-Hwan Lee, and Ki-Ryong Kwon. “Acute Leukemia Classification Using Convolution Neural Network in Clinical Decision Support System.” In *Computer Science & Information Technology (CS & IT)*, 49–53. Academy & Industry Research Collaboration Center (AIRCC), 2017. https://doi.org/10.5121/csit.2017.71305.
9. https://cs230.stanford.edu/blog/datapreprocessing/
10. https://github.com/Arsey/keras-transfer-learning-for-oxford102/issues/1
11. https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/
12. https://datascience.stackexchange.com/questions/14899/how-to-draw-deep-learning-network-architecture-diagrams
13. Mostavi, Milad, Yu-Chiao Chiu, Yufei Huang, and Yidong Chen. “Convolutional Neural Network Models for Cancer Type Prediction Based on Gene Expression.” *BMC Medical Genomics* 13, no. S5 (April 2020): 44. https://doi.org/10.1186/s12920-020-0677-2.
14. https://shiva-verma.medium.com/understanding-different-loss-functions-for-neural-networks-dd1ed0274718
15. https://medium.com/@dipti.rohan.pawar/improving-performance-of-convolutional-neural-network-2ecfe0207de7
16. Shafique, Sarmad, and Samabia Tehsin. “Acute Lymphoblastic Leukemia Detection and Classification of Its Subtypes Using Pretrained Deep Convolutional Neural Networks.” *Technology in Cancer Research & Treatment* 17 (January 1, 2018): 153303381880278. https://doi.org/10.1177/1533033818802789.
17. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
18. https://www.youtube.com/watch?v=km7pxKy4UHU
19. https://towardsdatascience.com/building-a-convolutional-neural-network-male-vs-female-50347e2fa88b
20. https://www.upgrad.com/blog/basic-cnn-architecture/