Detection of Acute Lymphocytic Leukemia (ALL) with Convolutional Neural Network

Name: Ragi Unnithan
Professor: Xinlian Liu
Date: 12/08/2021

M.S. Computer Science
Hood College
Frederick
MD

Introduction

- Acute Lymphocytic Leukemia (ALL) is a type of cancer of the blood and bone narrow
- ALL is the most common cause of pediatric cancer and the most frequent cause of death from cancer before 20 years of age
- Cell classification usually depends on morphological characteristics of the cell and requires a skilled medical operator
- These procedures are time consuming, tedious, costly and error prone
- In this project, we are trying to see whether a Convolutional Neural Network can classify Normal and ALL cells from microscopic blood images

Understanding the Dataset

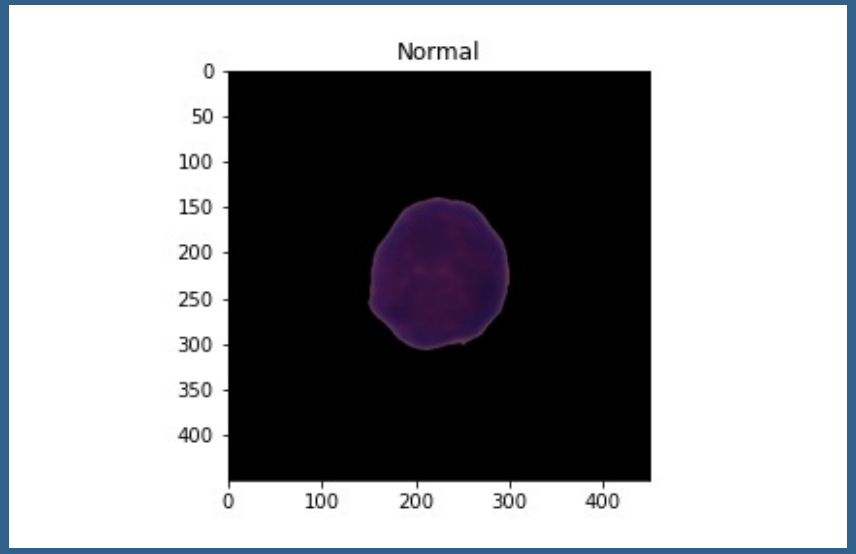
- Obtained from Kaggle – C_NMC_Leukemia
- Images are segmented from microscopic images
- Total of 15,135 images from 118 patients with 2 labels
  - 49 patients without ALL
  - 69 patients with ALL

Cell Type	Train	Validation	Test
ALL	47	13	9
Normal	26	15	8
Total	73	28	17

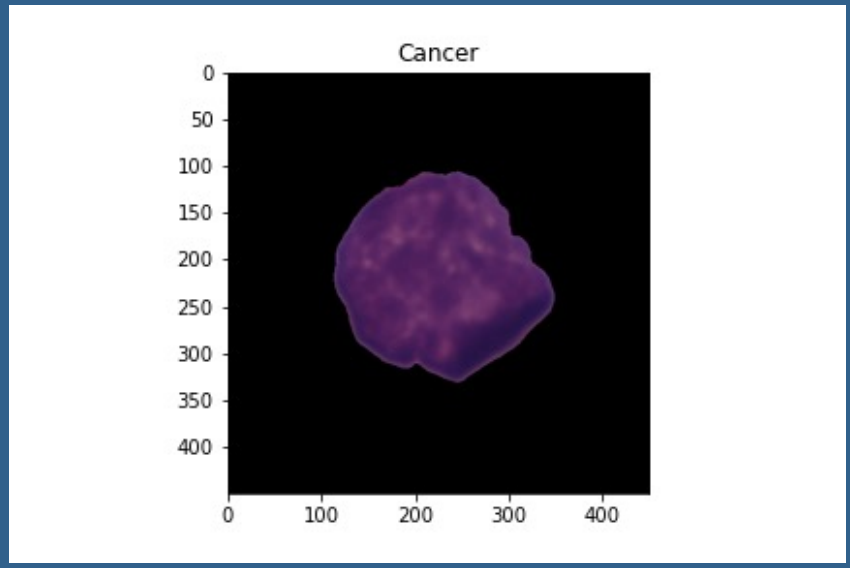
Dataset – subtype	Total	Normal	ALL
Training	10661	3389	7272
Validation	1867	1219	648
Test	2586	-	-

Exploratory Data Analysis

- **Normal Cells**
  - Spherical
  - Non-clefted
  - Few vacuoles

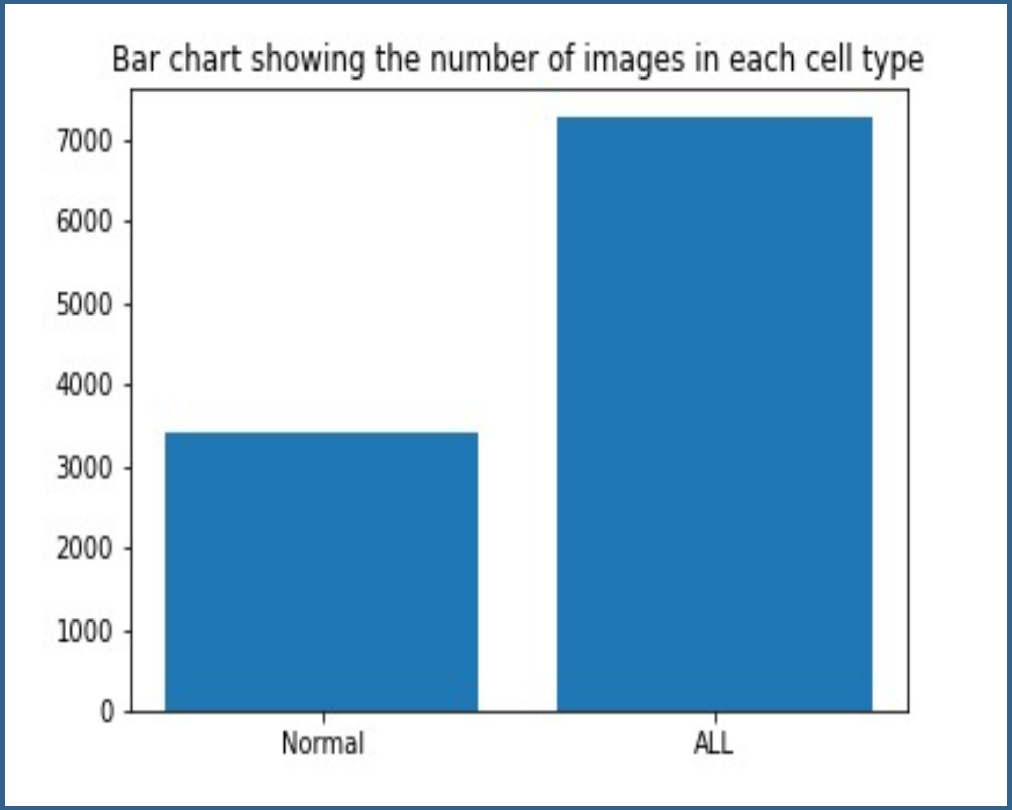


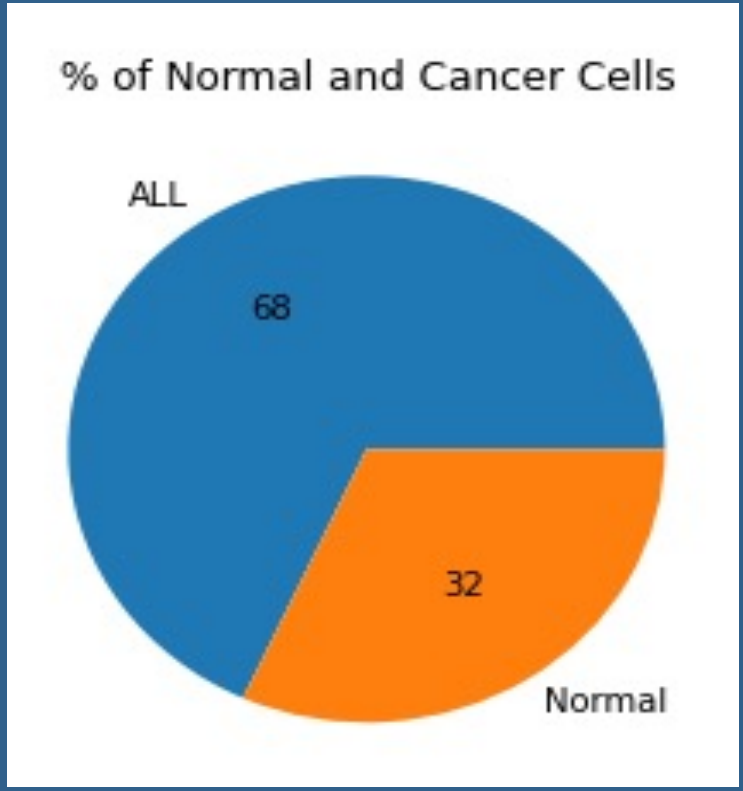
- **Caner Cells**
  - Non-Spherical
  - Clefted
  - Vacuoles



Data Distribution

Imbalanced dataset; 68% is ALL and 32% is Normal





Dataset Split

- Training and Validation dataset are labelled
- Test dataset not labelled
- So, the 10661 data in the training is split into training (75%) and validation (25%)
- The validation dataset is used as testing

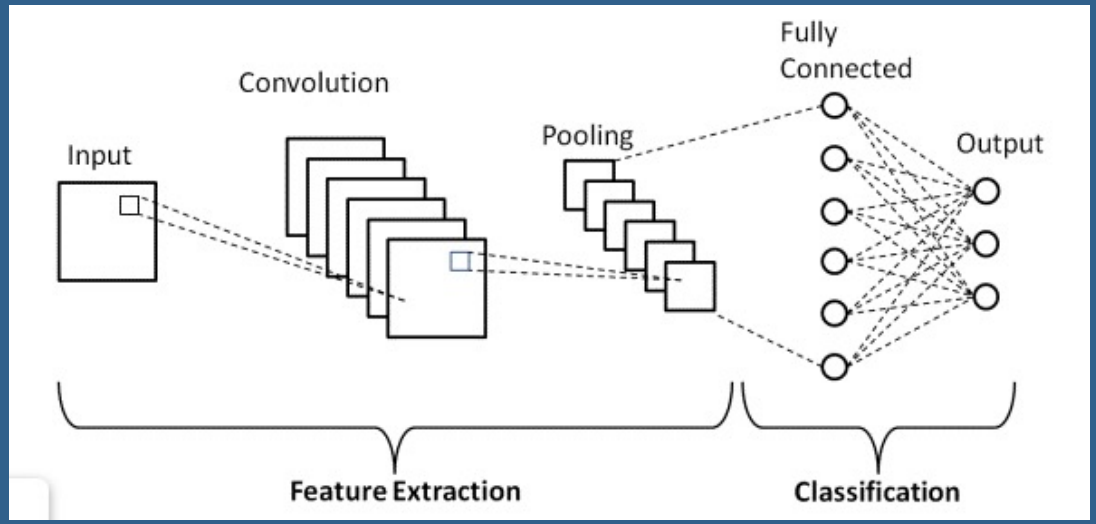
Dataset - subtype	Total	Normal	ALL
Training	7996	2542	5454
Validation	2665	847	1818
Test	1867	1219	648

Dataset Preprocessing

- Images are 450 x 450 RGB images stored as .bmp files
- Images are resized to 256 x 256
- Rescaled between ranges 0 to 1.

Convolutional Neural Network (CNN)

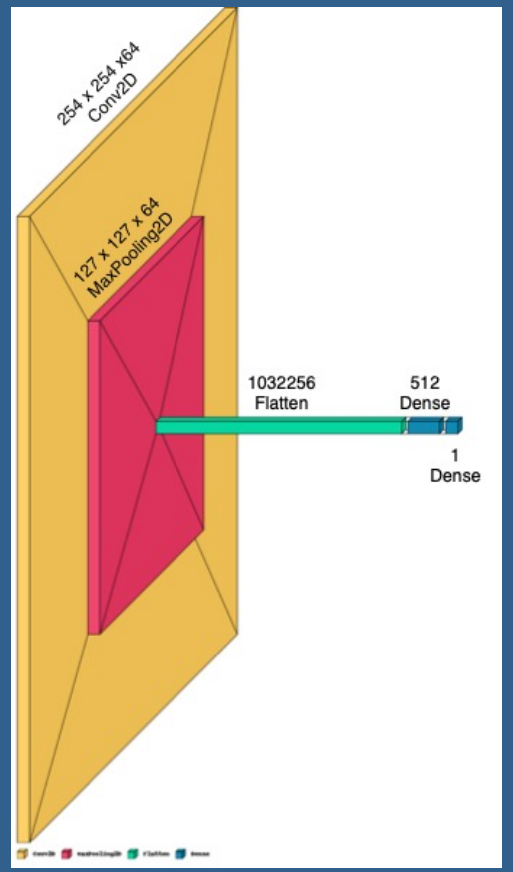
- Convolutional Neural Network is composed of
  - Convolutional layers
  - Activation Function
  - Pooling Layers
  - Fully Connected Layers



Source: https://www.upgrad.com/blog/basic-cnn-architecture/

A simple Vanilla Model (1C1D)

- Input is an image of size 256 x 256 x 3
- A single convolutional layer
  - 64 filters
  - filter size is 3 X 3
  - Relu activation [
- Max Pooling
  - 2 x 2 pooling window
- Flatten layer
- Two fully connected dense layer




```
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
_________________________________________________________________
```

[(W−K+2P)/S]+1
W is the input volume
K is the Kernel size
P is the padding
S is the stride

Parameters = (3*3*3 + 1) * 64

Result

Model	Optimizer	Training			Validation			Test		
		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

Model Selection

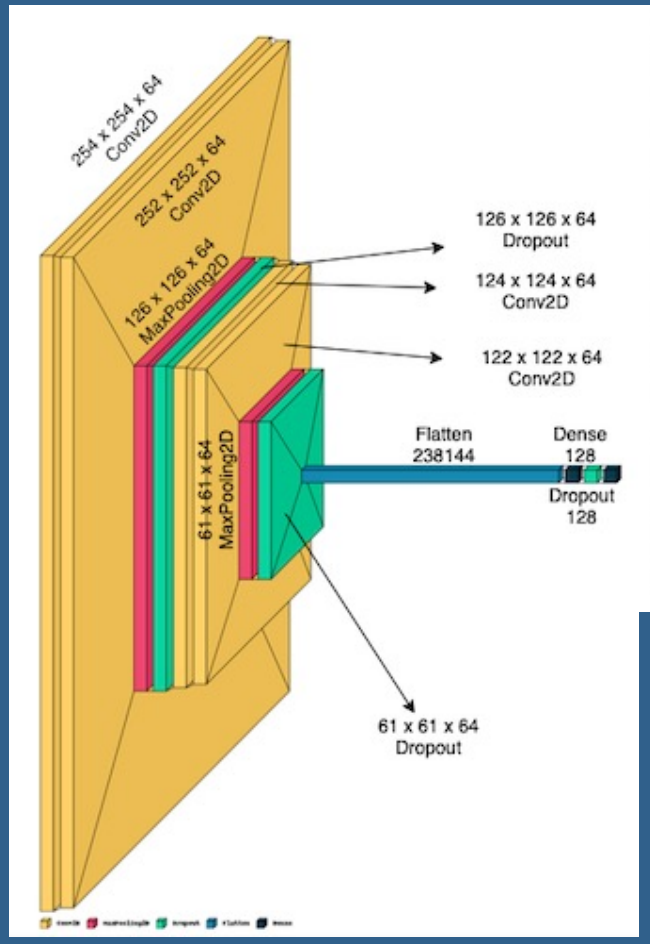
- Many models with successive larger layers and complex architectures were build
- Multiple convolutional layers
- Max pooling layers
- Dropout layers (0.25 and 0.5)
- Batch Normalization
- Binary-cross entropy is used for the loss function as this is a binary classification problem
- Optimizer – Adam and RMSProp

Summary from all the Models

Model	Optimizer	Training			Validation			Test		
		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 - 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

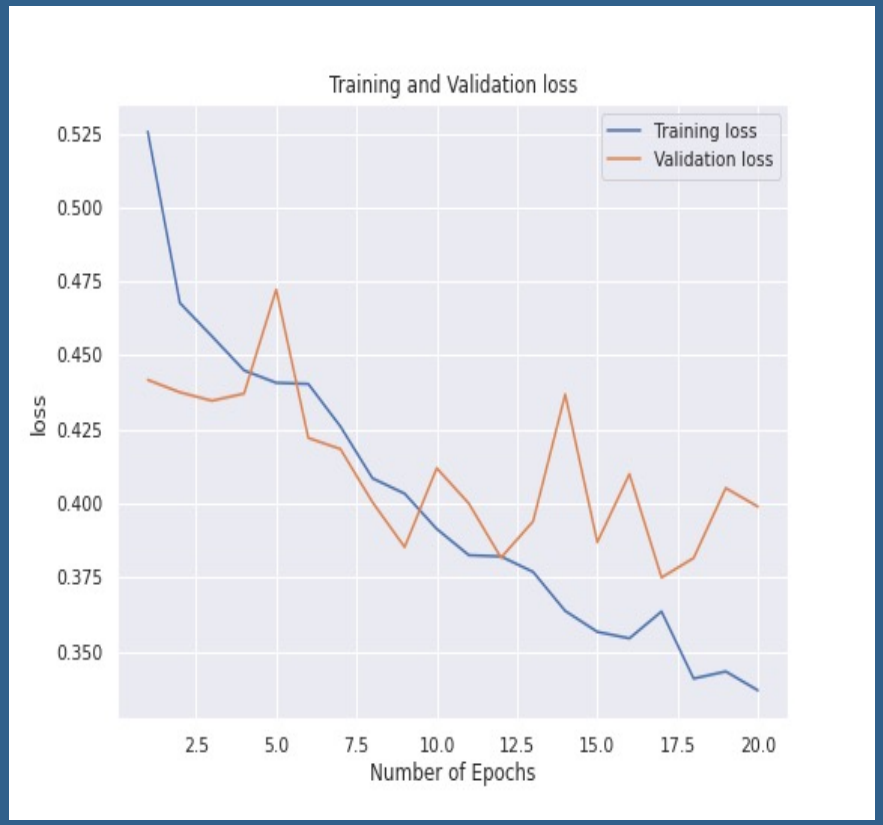
Chosen Model – 2x2C1D

- Input is an image of size 256x256x3
- Four convolutional layer
  - 64 filters
  - filter size is 3 X 3
  - Relu activation
- Two Max Pooling
  - 2 x 2 pooling window
- Three Dropout layers with a value of 0.5
- Flatten layer
- Two Dense layers

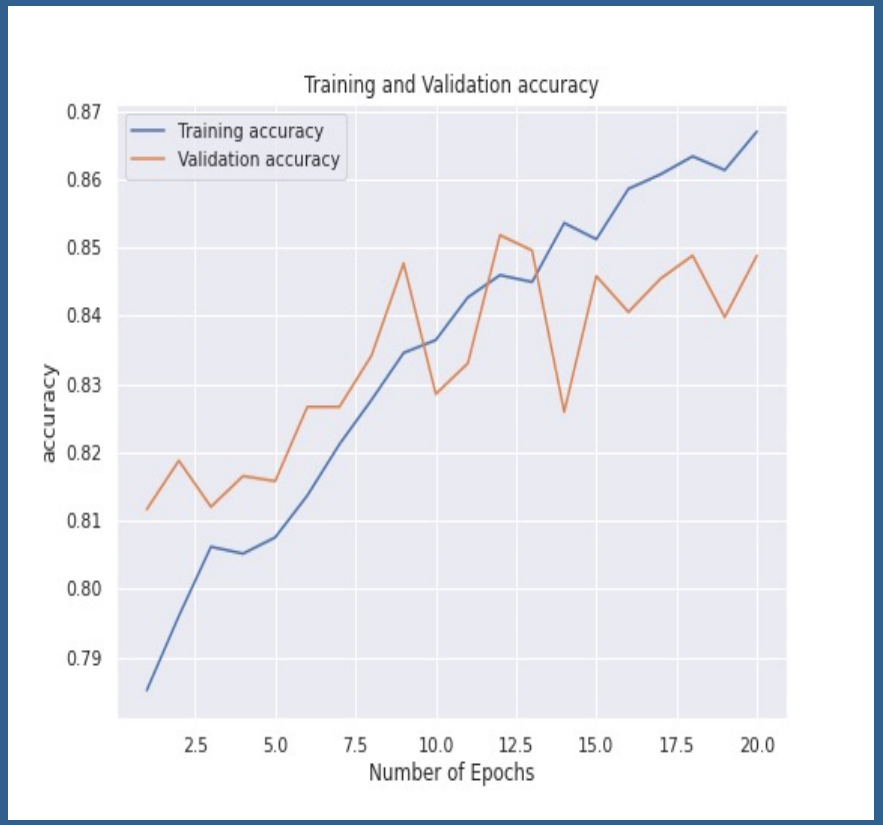



```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_41 (Conv2D)           (None, 254, 254, 64)      1792
_________________________________________________________________
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
```

Result – 2x2C1D



- With 100 epochs
  - Training accuracy – 94%
  - Validation accuracy – 85%
  - Testing accuracy – 61%



Model	Optimizer	Training			Validation			Test		
		Accuracy	Recall	Loss	Accuracy	Recall	Loss	Accuracy	Recall	Loss
2X2C1D - Dropout(0.5)	RMSprop	0.8112	0.598	0.446	0.83	0.621	0.418	0.617	0.108	0.8515

Conclusion and Future Goals

- CNN can be used for image classification and as a tool for identifying ALL
- This method can be used by a physician to verify his diagnosis
- Batch Normalization and Dropout can be used for reducing overfitting
- In future we can implement early stopping and cross-validation for avoiding overfitting
- Image data augmentation can be implemented to the normal dataset for the eliminating class imbalance
- Try implementing Alexnet or other models to improve the accuracy

Questions ?

Thank you