By Group 7
Weiting Lin
Yuxi Jiang
Hungta Chen
Jaehyeon Park

On
June 6th

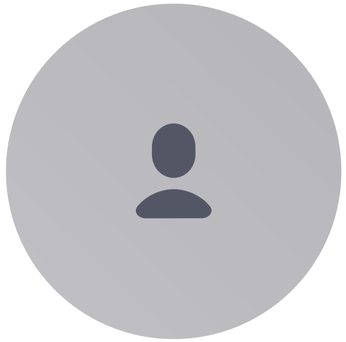
STA 208 Final Project

Big Data Analysis for Covid Detection

Group Members

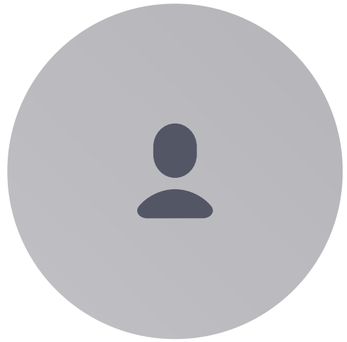
Weiting Lin

MS
wtin@ucdavis.edu



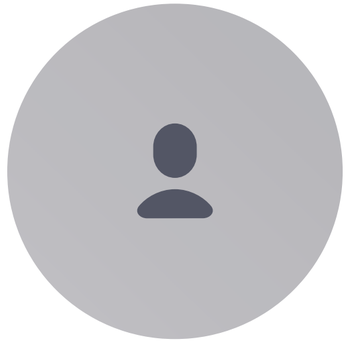
Yuxi Jiang

MS
yuijiang@ucdavis.edu



Jaehyeon Park

MS
jpapark@ucdavis.edu



Hungta Chen

MS
htcchen@ucdavis.edu

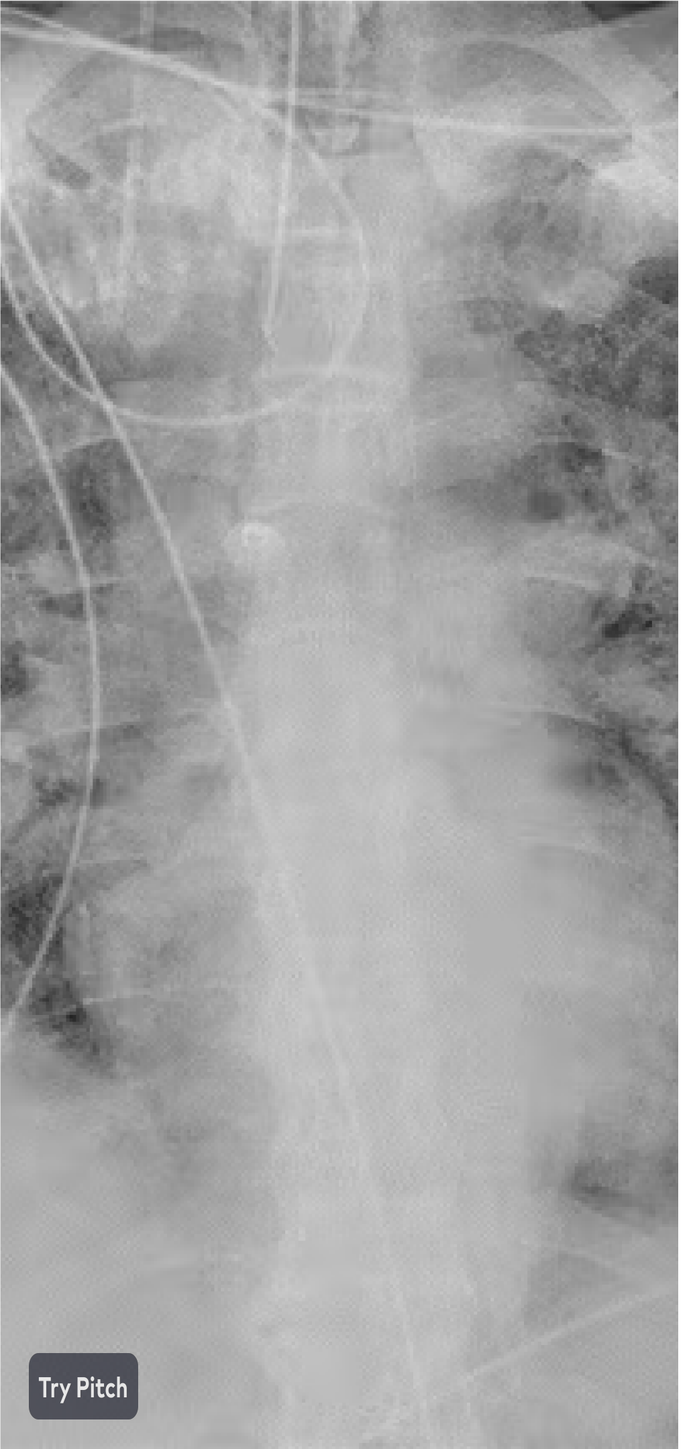
Overview

1 —

Introduction

Topic

Due to the Covid-19 pandemic, the lung is directly affected and prone to infection. However, it's important to note that the virus can also infect the lungs, **leading to potential misdiagnosis based on X-ray images**. In order to assist doctors in accurately diagnosing cases, it becomes crucial to perform image classification to differentiate between X-ray images of virus-infected lungs and those infected with Covid-19. To achieve this, we have obtained a dataset from **Kaggle**, comprising of three distinct files containing uninfected lung X-ray images, virus-infected lung X-ray images, and Covid-infected lung images. To fulfill our objective, we will develop an image classification system utilizing CNN (Convolutional Neural Networks), leveraging the available datasets.



Data sources

Our dataset consists of 1823 images of an annotated posteroanterior (PA) view of Chest X-ray images. Labeled Optical Coherence Tomography(OCT) and CXR Images used for viral pneumonia and non-pneumonia or normal cases

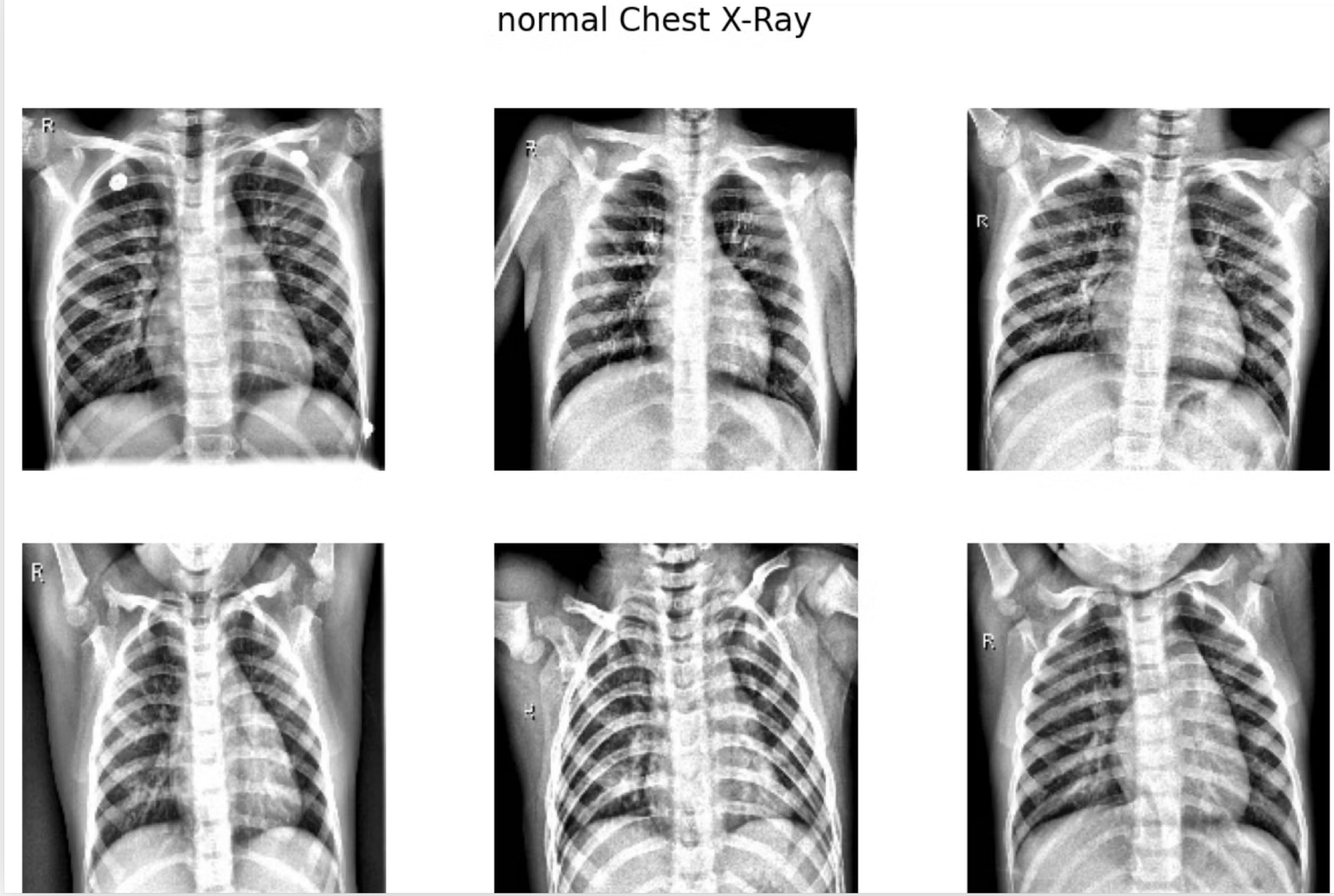
kaggle

In particular, this dataset consists of 536 images of COVID-19,619 images of viral pneumonia and 668 images of normal cases. The age range of COVID-19 cases in the dataset is 18-75 years.

2 ——

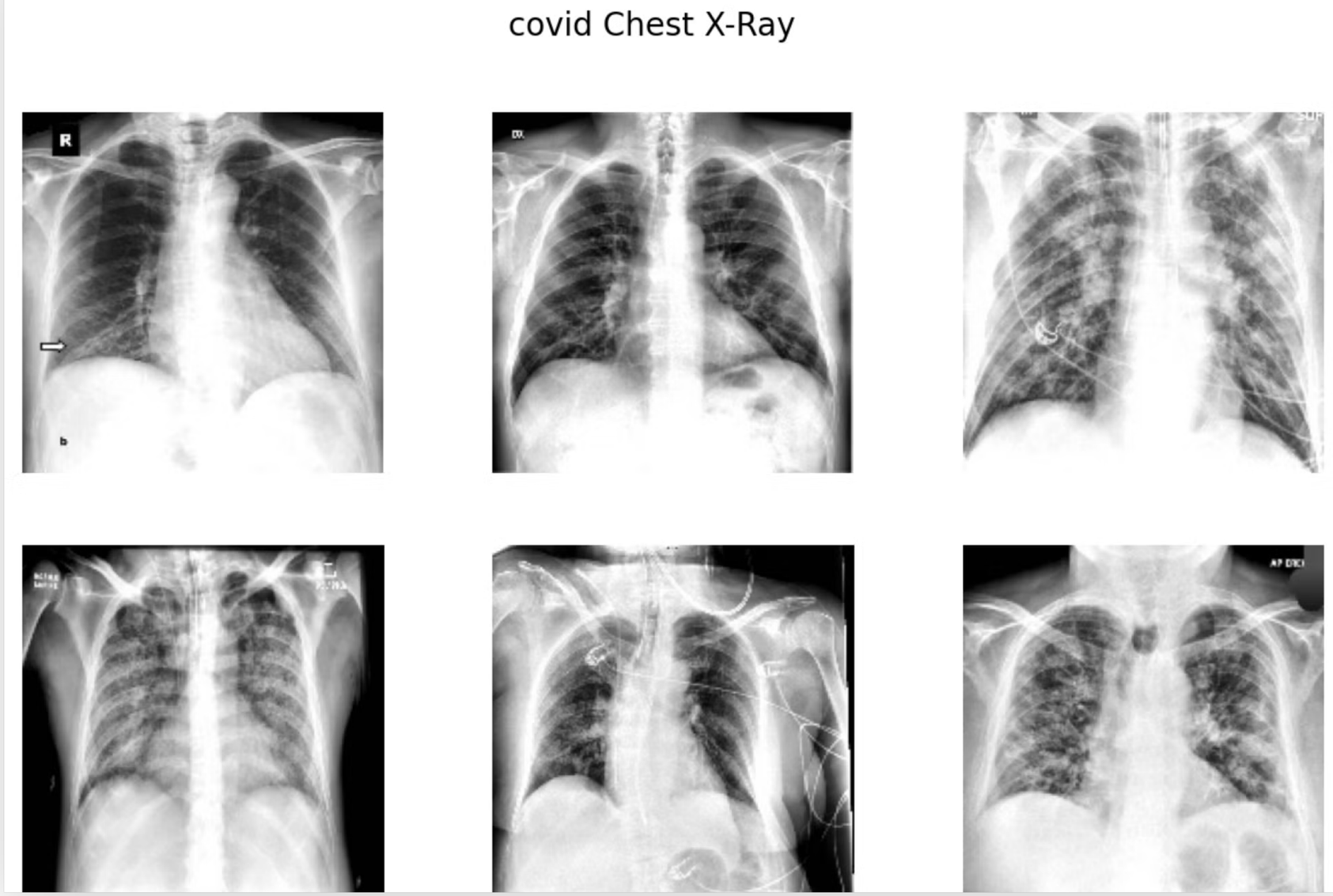
Data preprocessing

```
visual("normal",normal_adj)
```



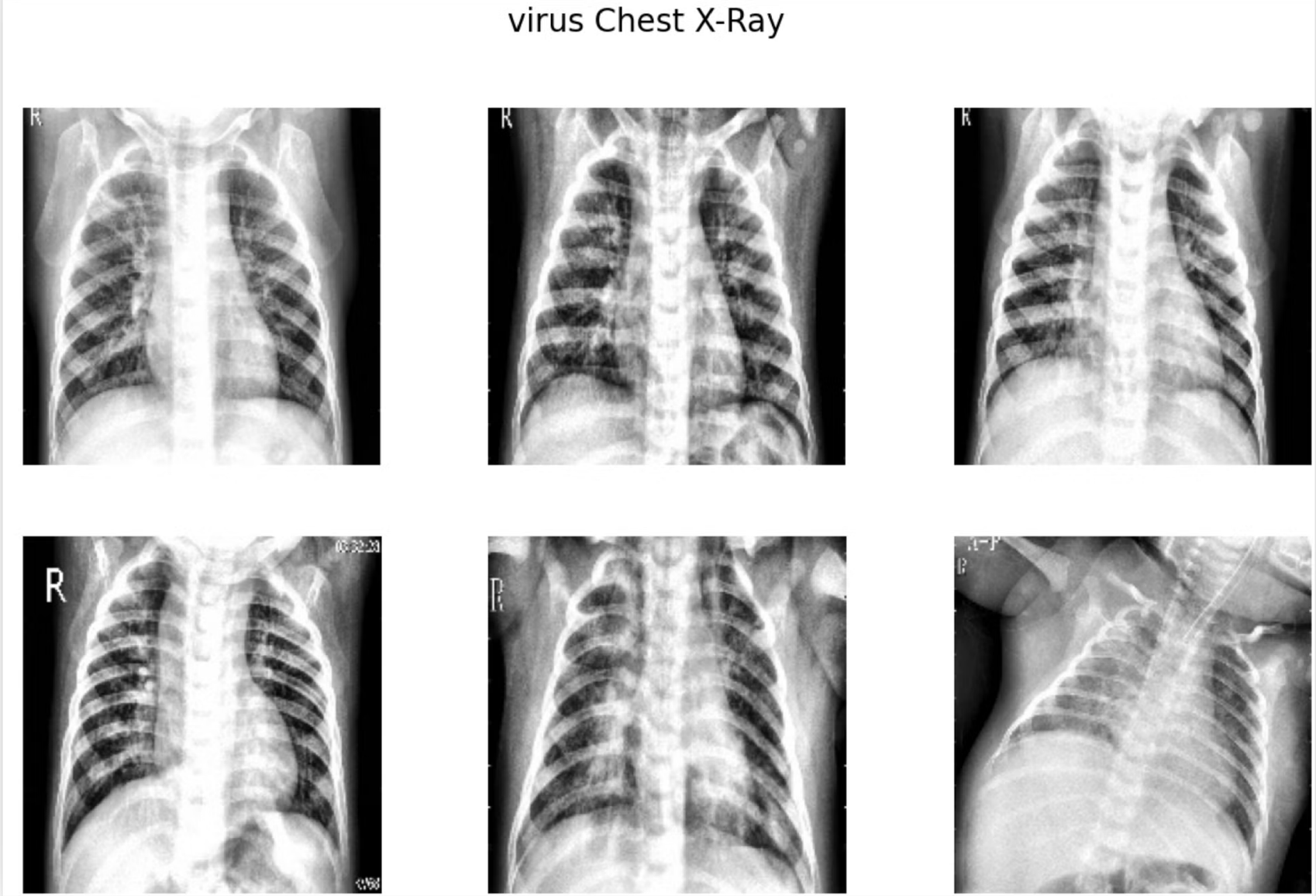
Normal Chest X-ray

```
visual("covid",covid_adj)
```



Covid Chest X-ray

```
visual("virus",virus_adj)
```



Virus Chest X-ray

3 ——

Methodology

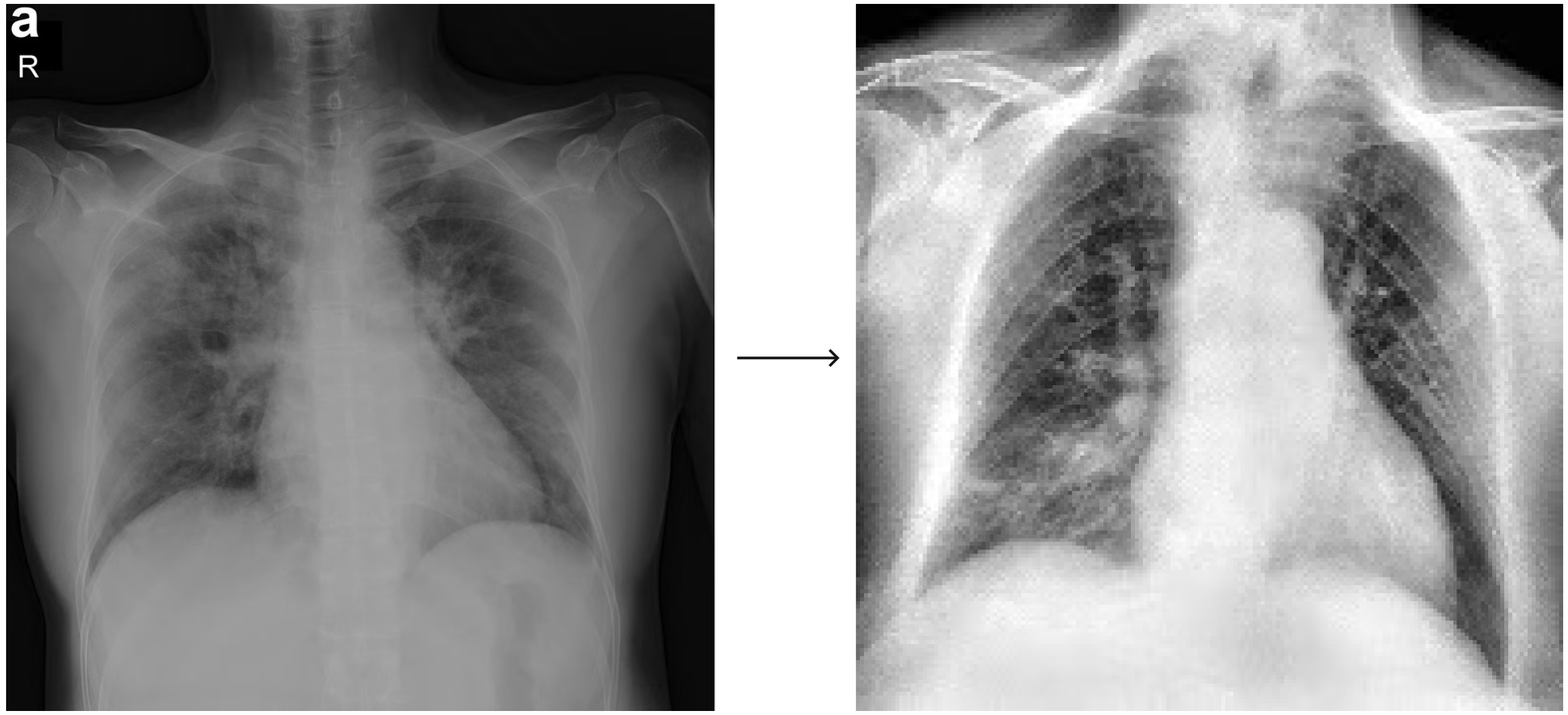
Methodology

01

CLAHE

(Contrast Limited Adaptive Histogram Equalization)

Enhance **the contrast and brightness of images**. It analyzes the histogram of an image and adaptively adjusts the contrast in different regions, resulting in improved details and visual quality. In our project, we will employ CLAHE to enhance the contrast and brightness of the images in our dataset.



Methodology

02

CNN

(Convolutional Neural Network)

A **powerful deep learning architecture** commonly used for computer vision tasks. It consists of convolutional layers that extract features from images and pooling layers that downsample the extracted features. In our project, we will develop an image classification model

03

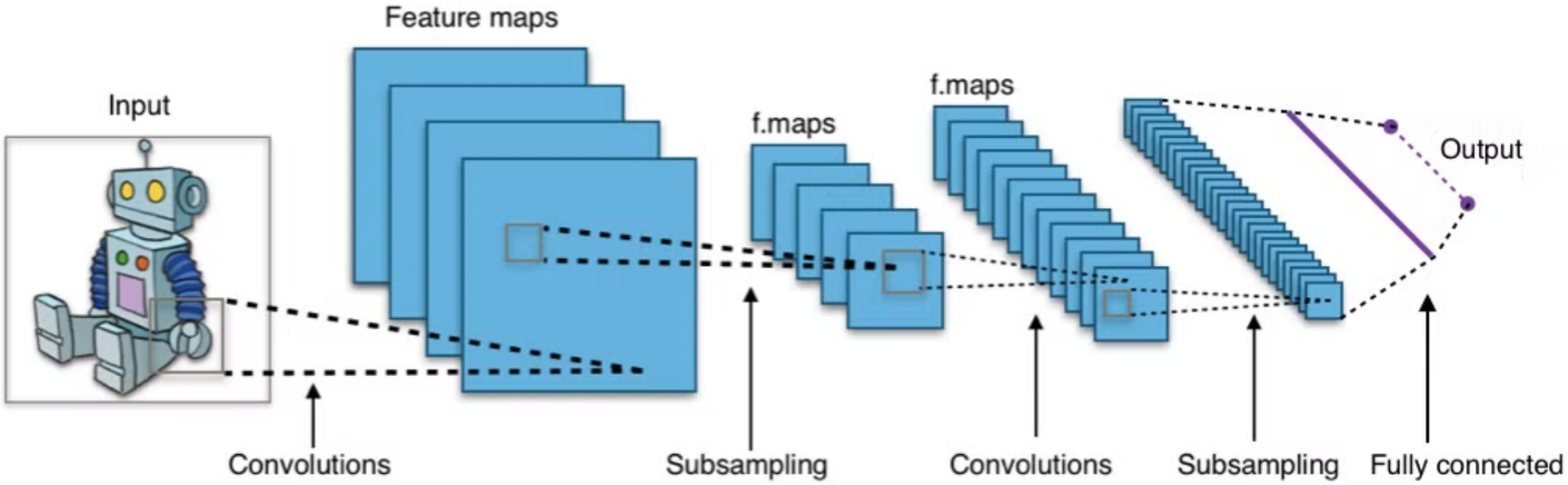
Alternative approach

(using numpy)

By implementing the CNN algorithm using numpy, we can compare its performance with the results obtained using the **tensorflow.keras package**. This allows us to evaluate the effectiveness and efficiency of different implementation approaches. The comparison will provide insights into the trade-offs and advantages of using specialized deep learning frameworks versus more general-purpose numerical computing libraries.

CNN

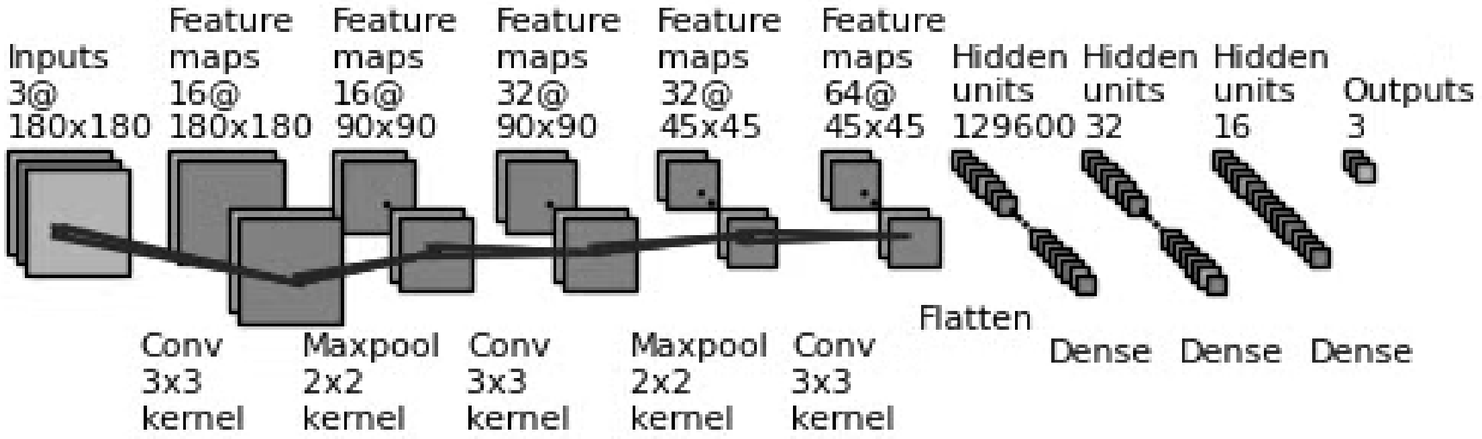
(Convolutional Neural Network)



CNN

(Convolutional Neural Network)

1. Resize image from 360*360 to 180*180
2. Change the iteration number from 10000 to 50
3. Change batch size from 200 to 50



4 —

Results

code

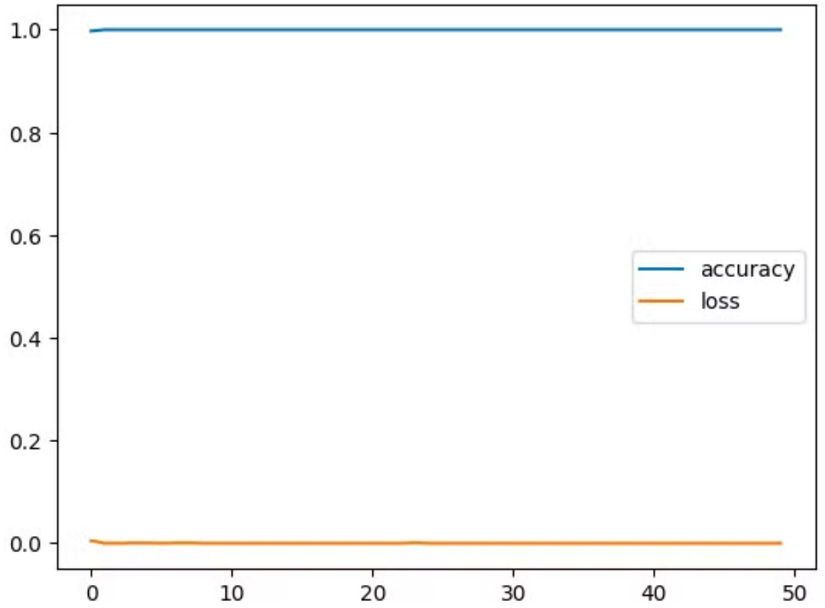
Results

```
def CNN(x_train,x_test,y_train,y_test,x_cv,y_cv,opt):

    x_train=np.asarray(x_train)
    x_test=np.asarray(x_test)
    y_train=np.asarray(y_train)
    y_test=np.asarray(y_test)
    x_cv=np.asarray(x_cv)
    y_cv=np.asarray(y_cv)

    #convlution & maxpool
    model = models.Sequential()
    model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(180, 180, 3),strides=(1, 1),padding="same"))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(32, (3, 3), activation='relu',strides=(1, 1),padding="same"))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.Conv2D(64, (3, 3), activation='relu',strides=(1, 1),padding="same"))

    #flatten layer & output
    model.add(layers.Flatten())
    model.add(layers.Dense(32, activation='relu'))
    model.add(layers.Dropout(0.2))
    model.add(layers.Dense(16, activation='relu'))
    model.add(layers.Dense(3))


    model.compile(optimizer=opt,loss="sparse_categorical_crossentropy",metrics=["accuracy"])
    print(opt)
    model.fit(x_train, y_train, epochs=50,batch_size=10,verbose=1, validation_data=(x_cv,y_cv))
    loss,accuracy=model.evaluate(x_test,y_test)
    print(f"Test loss - {opt} : {loss:.4f}")
    print(f"Test accuracy - {opt}: {accuracy:.4f}")
    # model.save("model.h5")
```

RMS Prop

Results

```
In [145]: CNN(x_train,x_test,y_train,y_test,x_cv,y_cv,'RMSprop')
          - val_accuracy: 1.0000
          Epoch 46/50
          115/115 [==============================] - 8s 67ms/step - loss: 0.0373 - accuracy: 0.9661 - val_loss: 2.3842e-07
          - val_accuracy: 1.0000
          Epoch 47/50
          115/115 [==============================] - 8s 70ms/step - loss: 0.0392 - accuracy: 0.9643 - val_loss: 2.3842e-07
          - val_accuracy: 1.0000
          Epoch 48/50
          115/115 [==============================] - 8s 70ms/step - loss: 0.0459 - accuracy: 0.9582 - val_loss: 2.3842e-07
          - val_accuracy: 1.0000
          Epoch 49/50
          115/115 [==============================] - 8s 71ms/step - loss: 0.0488 - accuracy: 0.9556 - val_loss: 2.3842e-07
          - val_accuracy: 1.0000
          Epoch 50/50
          115/115 [==============================] - 8s 71ms/step - loss: 0.0507 - accuracy: 0.9539 - val_loss: 2.3842e-07
          - val_accuracy: 1.0000
          18/18 [==============================] - 1s 50ms/step - loss: 2.3842e-07 - accuracy: 1.0000
          Test loss - RMSprop : 0.0000
          Test accuracy - RMSprop: 1.0000
```

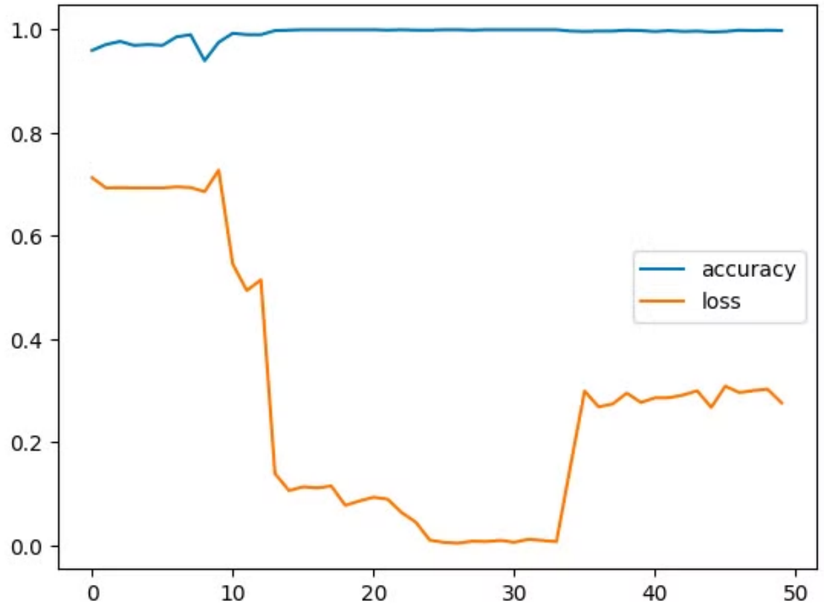


RMSprop $\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t$

Adam

Results

```
In [141]: CNN(x_train,x_test,y_train,y_test,x_cv,y_cv,'Adam')
-05 - val_accuracy: 1.0000
Epoch 46/50
115/115 [==============================] - 8s 70ms/step - loss: 1.4708e-05 - accuracy: 1.0000 - val_loss: 1.9920e
-05 - val_accuracy: 1.0000
Epoch 47/50
115/115 [==============================] - 8s 72ms/step - loss: 1.4643e-05 - accuracy: 1.0000 - val_loss: 1.9778e
-05 - val_accuracy: 1.0000
Epoch 48/50
115/115 [==============================] - 8s 71ms/step - loss: 1.4547e-05 - accuracy: 1.0000 - val_loss: 1.9621e
-05 - val_accuracy: 1.0000
Epoch 49/50
115/115 [==============================] - 8s 71ms/step - loss: 1.4461e-05 - accuracy: 1.0000 - val_loss: 1.9481e
-05 - val_accuracy: 1.0000
Epoch 50/50
115/115 [==============================] - 8s 70ms/step - loss: 1.4467e-05 - accuracy: 1.0000 - val_loss: 1.9317e
-05 - val_accuracy: 1.0000
18/18 [==============================] - 1s 53ms/step - loss: 1.9634e-05 - accuracy: 1.0000
Test loss - Adagrad : 0.0000
Test accuracy - Adagrad: 1.0000
```

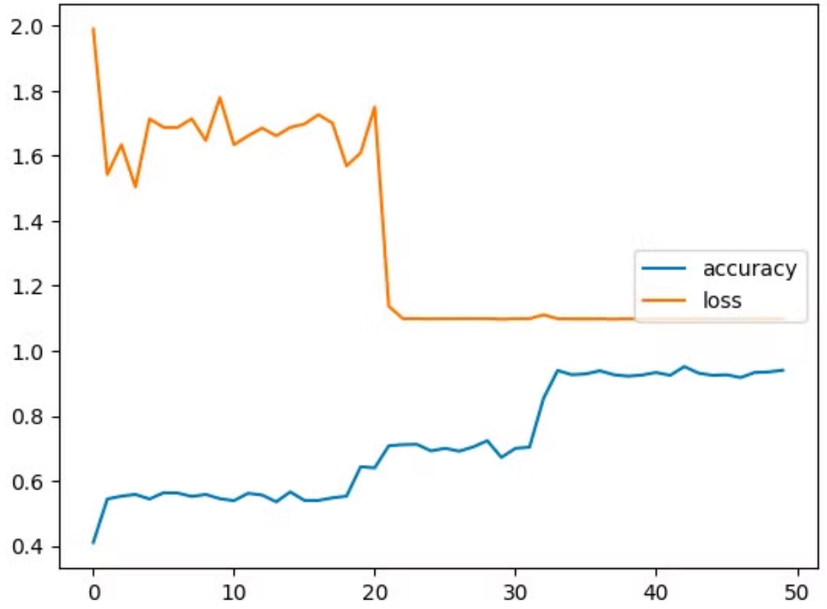


Adam $$\Delta\theta_t = -\frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

SGD

Results

```
In [146]: CNN(x_train,x_test,y_train,y_test,x_cv,y_cv,'SGD')
l_accuracy: 1.0000
Epoch 46/50
115/115 [==============================] - 7s 65ms/step - loss: 0.6931 - accuracy: 0.9452 - val_loss: 0.6931 - va
l_accuracy: 1.0000
Epoch 47/50
115/115 [==============================] - 8s 72ms/step - loss: 0.6931 - accuracy: 0.9460 - val_loss: 0.6931 - va
l_accuracy: 1.0000
Epoch 48/50
115/115 [==============================] - 7s 63ms/step - loss: 0.6931 - accuracy: 0.9556 - val_loss: 0.6931 - va
l_accuracy: 1.0000
Epoch 49/50
115/115 [==============================] - 8s 69ms/step - loss: 0.6931 - accuracy: 0.9460 - val_loss: 0.6931 - va
l_accuracy: 1.0000
Epoch 50/50
115/115 [==============================] - 8s 67ms/step - loss: 0.6931 - accuracy: 0.9399 - val_loss: 0.6931 - va
l_accuracy: 1.0000
18/18 [==============================] - 1s 49ms/step - loss: 0.6931 - accuracy: 1.0000
Test loss - SGD : 0.6931
Test accuracy - SGD: 1.0000
```



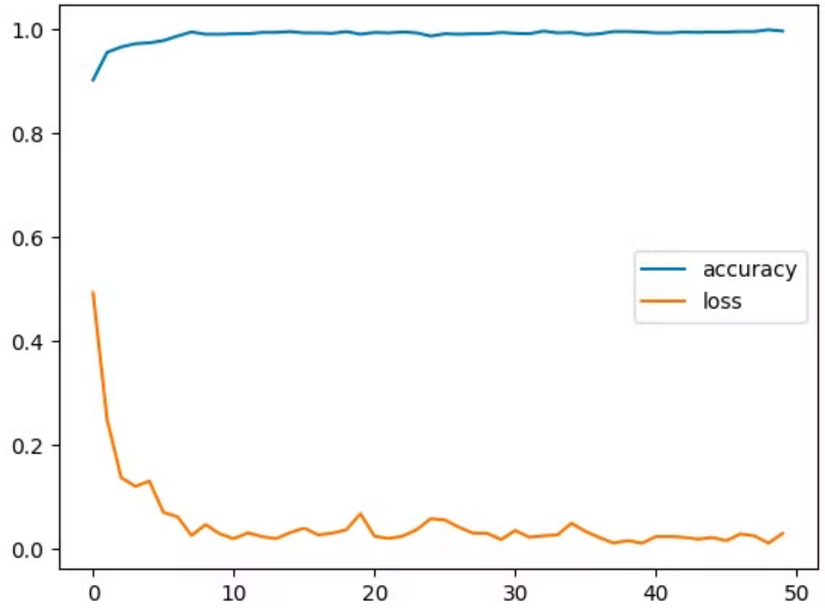
SGD

$$g_t = \nabla_{\theta_t} J(\theta_t)$$
$$\Delta\theta_t = -\eta \cdot g_t$$
$$\theta_t = \theta_t + \Delta\theta_t$$

Adadelta

Results

```
In [147]: CNN(x_train,x_test,y_train,y_test,x_cv,y_cv,'Adadelta')
-07 - val_accuracy: 1.0000
Epoch 46/50
115/115 [==============================] - 8s 66ms/step - loss: 4.8608e-06 - accuracy: 1.0000 - val_loss: 4.4023e
-07 - val_accuracy: 1.0000
Epoch 47/50
115/115 [==============================] - 8s 68ms/step - loss: 1.9556e-04 - accuracy: 1.0000 - val_loss: 4.3647e
-07 - val_accuracy: 1.0000
Epoch 48/50
115/115 [==============================] - 8s 67ms/step - loss: 4.4737e-07 - accuracy: 1.0000 - val_loss: 4.3647e
-07 - val_accuracy: 1.0000
Epoch 49/50
115/115 [==============================] - 8s 68ms/step - loss: 8.6613e-04 - accuracy: 1.0000 - val_loss: 4.3835e
-07 - val_accuracy: 1.0000
Epoch 50/50
115/115 [==============================] - 8s 66ms/step - loss: 1.3629e-04 - accuracy: 1.0000 - val_loss: 4.2709e
-07 - val_accuracy: 1.0000
18/18 [==============================] - 1s 47ms/step - loss: 4.4088e-07 - accuracy: 1.0000
Test loss - Adadelta : 0.0000
Test accuracy - Adadelta: 1.0000
```

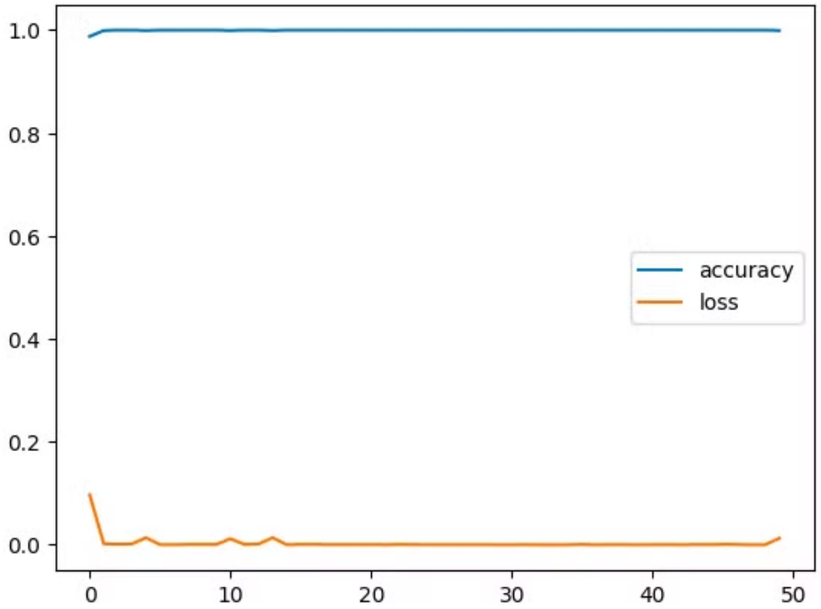


Adadelta $$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t$$

Adagrad

Results

```
In [145]: CNN(x_train,x_test,y_train,y_test,x_cv,y_cv,'RMSprop')
- val_accuracy: 1.0000
Epoch 46/50
115/115 [==============================] - 8s 67ms/step - loss: 0.0373 - accuracy: 0.9661 - val_loss: 2.3842e-07
- val_accuracy: 1.0000
Epoch 47/50
115/115 [==============================] - 8s 70ms/step - loss: 0.0392 - accuracy: 0.9643 - val_loss: 2.3842e-07
- val_accuracy: 1.0000
Epoch 48/50
115/115 [==============================] - 8s 70ms/step - loss: 0.0459 - accuracy: 0.9582 - val_loss: 2.3842e-07
- val_accuracy: 1.0000
Epoch 49/50
115/115 [==============================] - 8s 71ms/step - loss: 0.0488 - accuracy: 0.9556 - val_loss: 2.3842e-07
- val_accuracy: 1.0000
Epoch 50/50
115/115 [==============================] - 8s 71ms/step - loss: 0.0507 - accuracy: 0.9539 - val_loss: 2.3842e-07
- val_accuracy: 1.0000
18/18 [==============================] - 1s 50ms/step - loss: 2.3842e-07 - accuracy: 1.0000
Test loss - RMSprop : 0.0000
Test accuracy - RMSprop: 1.0000
```



Adagrad $$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t$$

5 ——

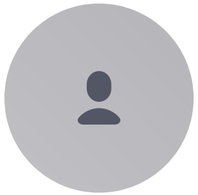
Questions

Reference

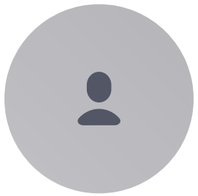
1. **Dataset resource**: https://www.kaggle.com/datasets/sid321axn/covid-cxr-image-dataset-research
2. **Coding resource**: https://chat.openai.com/, https://www.kaggle.com/code/sid321axn/covid-detection-from-cxr-using-explainable-cnn#Model-Building
3. **Visualization of CNN**: https://www.kaggle.com/getting-started/253300
4. **Formula for optimization method**: https://www.sravikiran.com/GSOC18//2018/05/16/optimizers/

The last slide

Thank you!

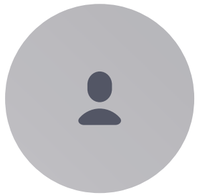


Weiting Lin
MS
wtin@ucdavis.edu



Yuxi Jiang
MS
yuijiang@ucdavis.edu

Jaehyeon Park
MS
jpapark@ucdavis.edu



Hungta Chen
MS
htcchen@ucdavis.com

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It's free)