

PROJECT DOCUMENTATION

TEAM MEMBERS

1. Jonathan
2. Henry
3. Marvin Christian
4. Nicky Nicholas

TABLE OF CONTENTS

Team Members	1
Table of Contents	1
Project Background	2
Aims of Solution	2
Usage	3
• Predict X-Ray Results	3
• Change Language	4
Repo Directory	4
How It Works and How To Deploy	5
Model Overview	5
App Flowchart	5
Running On Local Machine	6
Deploy To Google Cloud Run	7
Datasets	9
Tools and Programming Language	10
Additional Notes	10

PROJECT BACKGROUND

Pneumonia is one of the biggest health problems in the world that is not limited to developed countries. Ever since the COVID-19 pandemic, the mortality rate for patients infected with pneumonia has increased by 35%-50%. One way to diagnose cases of pneumonia is to perform X-ray scans of a patient's lungs, and have an expert examine the symptoms and treatments required for said patient. However, after interviewing with several physicians, we found that not all of them are knowledgeable in diagnosing X-ray results. Moreover, we found out that there were cases of experts not being available on-site to help with the examination of X-ray results. Additionally, misdiagnosis or delay in diagnosis by medical experts can lead to serious complications or even death in extreme cases. That's why we believe this is a phenomenon that can be explored to uncover potential solutions.

AIMS OF SOLUTION

Our team developed a web application called ParuParuKu. ParuParuKu is a Flask-based application for detecting pneumonia from X-ray images. It utilizes artificial intelligence (AI) that was built on a deep learning model that has been trained on datasets containing chest X-ray scans of patients with and without pneumonia.

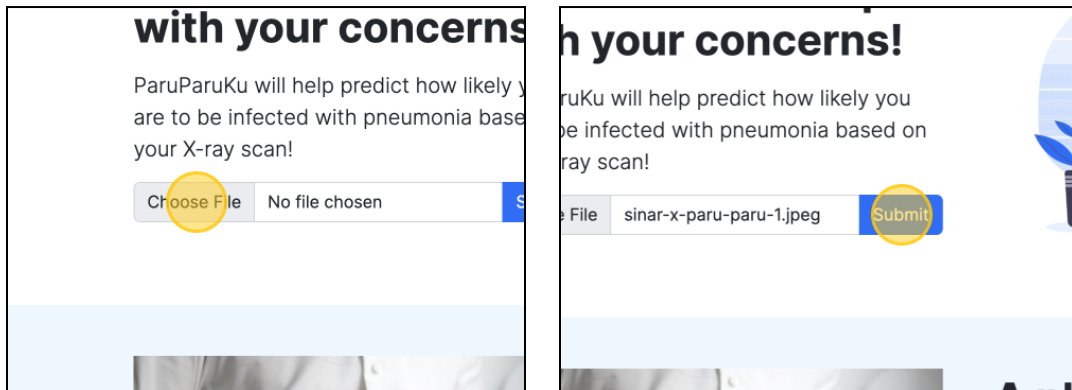
Our aim is to accurately and efficiently analyze X-ray images and provide predictions of whether or not pneumonia is present in a patient. We also aim to reduce the possibility for human error and give proper advice based on the predictions to fulfill the role as an expert system.

It's important to note that ParuParuKu is not intended to replace the needs for proper medical examinations. We believe ParuParuKu is here to assist users' decision-making process. In the end, we hope that by combining the ever-increasing power of technology, and the experiences of medical experts, we can contribute to the health of humanity for the better.

USAGE

- **PREDICT X-RAY RESULTS**

1. First, make sure the app is up and running on your local machine by checking out [this section](#). After the app successfully runs, you may proceed.
2. On the main page, you can click the input box like the image below. After choosing your X-ray images, you may click submit.



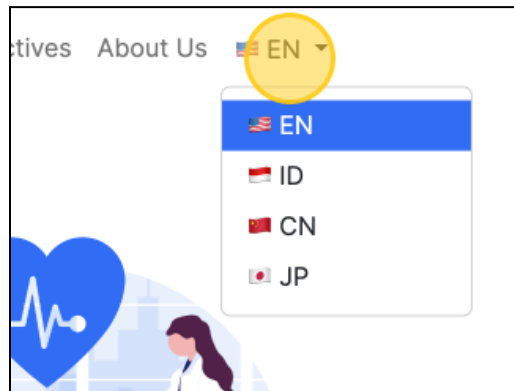
Notes:

You can try to use the images below as your test images. These images were not used to train the model.

- a. [Chest X-ray images](#)
 - b. [Random file](#)
3. After submitting, please wait a few seconds for the prediction results.
 4. You will get a set of results including:
 - a. Prediction results
 - b. GradCAM visualizations (Results visualizations)
 - c. Action recommendation
 5. If you want to rediagnose your results, you may upload it once more from the input box on the bottom of the page.

- **CHANGE LANGUAGE**

To change the language of the page, you can click on the language button at the top right of the navigation bar.



REPO DIRECTORY

- **models**

- **training-results**

This folder contains model training results, including confusion matrix, training and testing plot, and a table to recap all the results.

- **notebooks**

This folder contains the notebooks that were used to train the models.

- **model-outputs**

This folder contains all the notebook output in .h5 format.

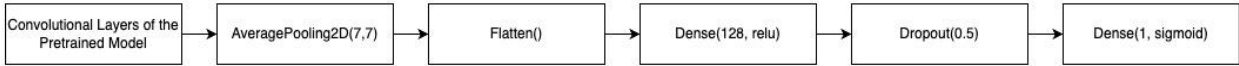
- **paruparuku-webapp**

This folder contains the main application. A more detailed explanation can be found in the [next section](#).

- **project-documentation.pdf** (this file)

HOW IT WORKS AND HOW TO DEPLOY

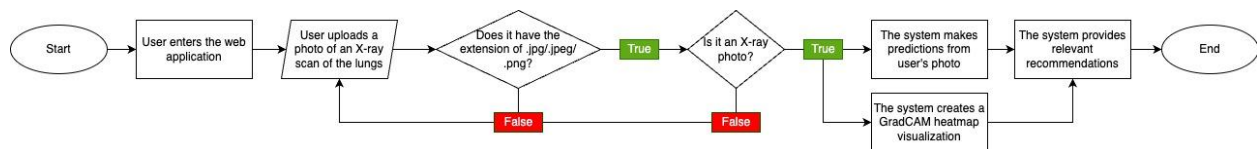
MODEL OVERVIEW



Our models were built on top of existing pre-trained models, namely DenseNet201, MobileNetV2, and NASNetMobile. By comparison with other models that we have trained, we have decided to only use these three models as they deliver the best results out of all. The same setup was used to train random image detector models, but we have decided to only use the MobileNetV2 model. This is because it delivered the same results as the other two, but with a smaller size in comparison.

Training was done on Google Colaboratory with detailed specifications contained in the notebooks. All training results and summary can be found on the repo, detailed location can be referred back to the [repo directory](#) section.

APP FLOWCHART



There are three main components of the prediction capabilities:

1. Initial File Check

There is a saying that you should “never trust user inputs” and we take that seriously. The app will check for a valid input as an image with the extension of .png, .jpg, or .jpeg. After that, it will check if the image is an X-ray image using a random-image detector that was explained previously. If either of those results return false, the user will be prompted to upload a new file.

2. Prediction and Visualization

As stated before, prediction will be run on all our three models to provide various results as a comparison. GradCAM heatmap will also be generated based on the results from the models' prediction. Prediction results and visualizations will be displayed on the result page.

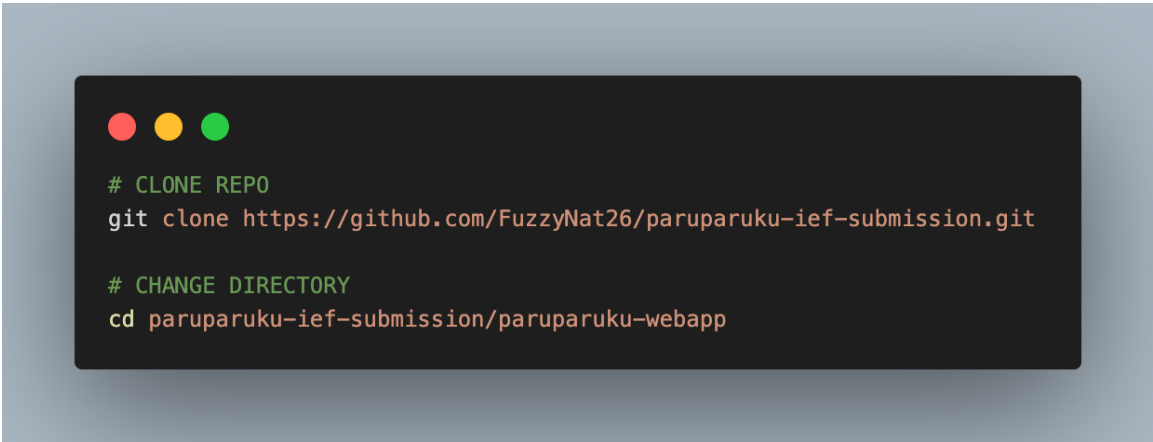
3. Result page

This component will provide the necessary information for the users, including the prediction results of each model, the models' visualization, and the call-to-action based on the severity of the pneumonia.

RUNNING ON LOCAL MACHINE

As our app is built using Flask, a series of packages is needed to run the app on the local machine. We suggest that an environment is to be used to manage the packages on your local machine. In this documentation, we used Visual Studio Code and Anaconda Navigator. You will need an internet connection for the installation process.

1. First, open Visual Studio Code or your preferred text editor.
2. Next, open your terminal and type:

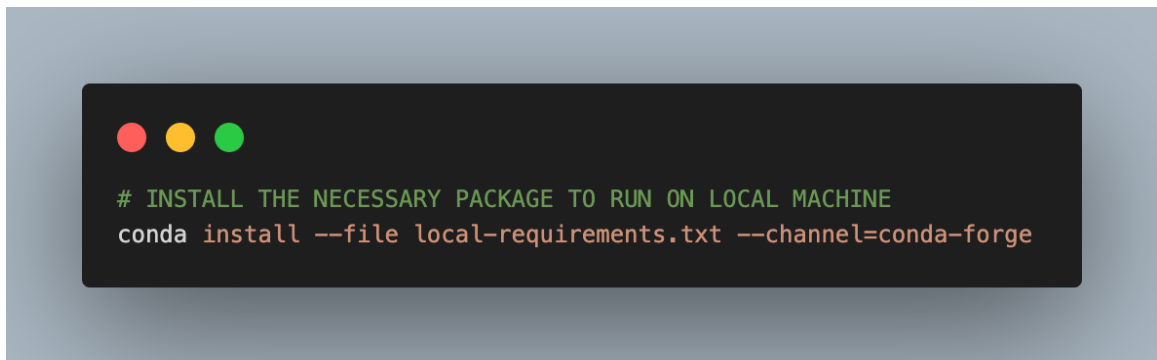
A screenshot of a terminal window with a dark background and light-colored text. At the top left, there are three colored circles (red, yellow, green) representing window control buttons. The terminal contains two lines of commands: the first line is a comment '# CLONE REPO' followed by 'git clone https://github.com/FuzzyNat26/paruparuku-ief-submission.git'; the second line is a comment '# CHANGE DIRECTORY' followed by 'cd paruparuku-ief-submission/paruparuku-webapp'.

```
# CLONE REPO
git clone https://github.com/FuzzyNat26/paruparuku-ief-submission.git

# CHANGE DIRECTORY
cd paruparuku-ief-submission/paruparuku-webapp
```

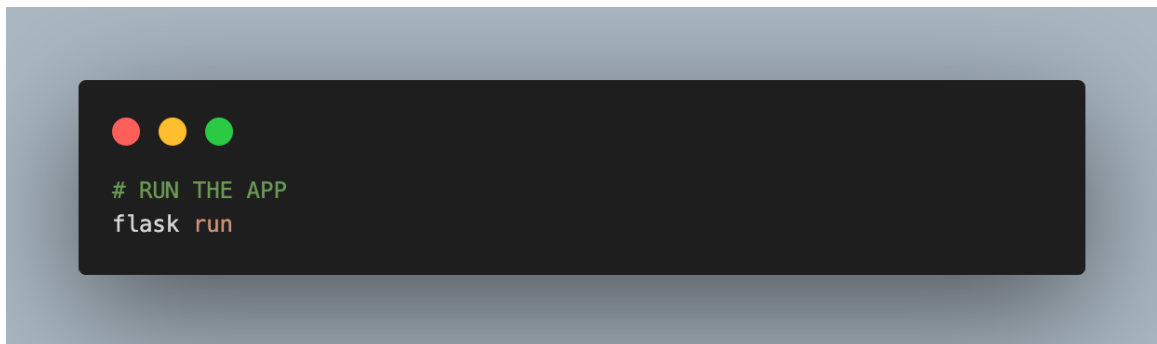
If you have cloned the repo, you only need to change the directory to **paruparuku-webapp**.

3. After that, activate your environment and install the packages by running the following command. Then, press “y” to proceed with the installation process.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The text inside the terminal is as follows:

```
# INSTALL THE NECESSARY PACKAGE TO RUN ON LOCAL MACHINE  
conda install --file local-requirements.txt --channel=conda-forge
```

4. After all installations are completed, type the following command on your terminal.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The text inside the terminal is as follows:

```
# RUN THE APP  
flask run
```

5. You should start to see the app is up and running on your local machine. Open your preferred browser and try it out!

DEPLOY TO GOOGLE CLOUD RUN

Our source code can also be deployed to serverless service. In this specific example, we will host it on Google Cloud Run. Our repository already contains necessary files to deploy the app to Google Cloud Run. These deployment steps are referenced to this [Google Cloud Documentation](#) with extra information. You will also need an internet connection for the deployment process.

1. Before starting the deployment, try running the app on your local machine by following [this section](#).

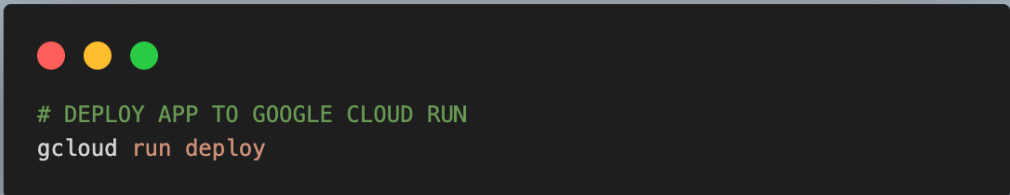
2. After everything is checked out, visit <https://console.cloud.google.com/>. If you're new to Google Cloud, create a new account first.
3. Next, in the Google Cloud Console, on the project selector page, create a Google Cloud project.
4. Make sure that billing is enabled for your Cloud project. Detailed instructions can be found on [this link](#).
5. Before continuing, make sure your directory is in **paruparuku-webapp**.
6. Install the [Google Cloud CLI](#).
7. To initialize the Google Cloud CLI, run the following command:



```
# INITIALIZE GOOGLE CLOUD CLI
gcloud init

# SET DEFAULT PROJECT FOR GOOGLE CLOUD RUN SERVICE
# REPLACE PROJECT_ID WITH THE NAME OF THE PROJECT
# THAT YOU CREATED ON 3RD STEP
gcloud config set project PROJECT_ID
```

8. To deploy this code, run the following command:



```
# DEPLOY APP TO GOOGLE CLOUD RUN
gcloud run deploy
```

There will be a few prompts to deploy this app.

- When you are prompted for the source code location, press **“Enter”** to deploy the current folder.
 - When you are prompted for the service name, press **“Enter”** to accept the default name.
 - If you are prompted to enable the Artifact Registry API or to allow creation of Artifact Registry repository, respond by pressing **“y”**.
 - When you are prompted for a region: select the region of your choice, for example **“asian-southeast1”**.
 - You will be prompted to allow unauthenticated invocations: respond by pressing **“y”**.
9. After the deployment process is done, there should be a link that is similar to <https://project-id-as.a.run.app>.

DATASETS

In order to train our deep learning model, we used the following datasets with the respective links below.

Name	Descriptions	Link
Chest X-ray Images (Pneumonia)	<p>Datasets were used to train pneumonia diagnosis models.</p> <p>The datasets consisted of 5856 scans (1583 healthy and 4273 diagnosed with pneumonia). Due to imbalanced data, we organized the images into 80% for training, 10% for validation, and 10% for testing.</p>	Chest X-Ray Images (Pneumonia) Kaggle
Image Captioning Dataset, Random Images	<p>Datasets were used to train user input detections.</p> <p>We used 1000 images from each dataset (totalling to 2000 random images).</p>	image captioning dataset, random images Kaggle
Unsplash Random Images Collection	<p>These datasets are compared with 2000 scans of chest x-ray (1000 for each category). Like</p>	Unsplash random images collection Kaggle

	previous datasets, we organized the images into 80% for training, 10% for validation, and 10% for testing.	
--	------------------------------------------------------------------------------------------------------------	--

TOOLS AND PROGRAMMING LANGUAGE

Tools

1. [Visual Studio Code](#), as code editor.
2. [Anaconda Navigator](#), as packages and environments manager.

Platform

1. [Browser](#), as developing target platform.
2. [Google Colaboratory](#), as a platform to train the deep learning models.

Frameworks

1. [Flask](#), as web application frameworks.
2. [Bootstrap](#), as application UI frameworks.

Programming Language

1. [Python](#), as the main programming language to build the AI models and application backbones.
2. [HTML/CSS/Javascript](#), to build the application layouts.

ADDITIONAL NOTES

If there is any problem or error regarding this submission, please contact us via email address / phone number.