Strike Stroke

Lee Seungsu
dept. Computer Science
Korea
2019034702
mqm0051@gmail.com

Park Geonryul
dept. Computer Science
Korea
2019040564
geonryul0131@gmail.com

Elia Ayoub
dept. Computer Science
France
9170420231
elia-ayoub@outlook.com

Ryan Jabbour
dept. Computer Science
France
9191820235
jabbourryan2@gmail.com

***Abstract*—Stroke is an acute and severe disease worldwide ranked as the [1] fourth cause of death in South Korea and [2] the fifth cause of death in the United States of America.**

[3] The most effective time to remove the blood clots caused by this disease is within an hour of the stroke, but of course, the sooner, the better. Otherwise, treatment should be administered within at least three hours in order to minimize potential complications. Since the hospital needs to conduct an MRI and a CT scan for diagnosis, we can consider [4] the best time to arrive at the hospital after a stroke would be one hour. However, in case of the onset of prognostic symptoms, the time of stroke cannot be predicted, so it is recommended that you go to the hospital right away.

Fortunately, [5] stroke has a reliable pre-hospital diagnostic method called BE-FAST (Balance, Eyes, Face, Arm, Speech, Terrible headache). Facial expression changes caused by paralysis of facial muscles are a very obvious symptom of stroke to detect, so you can make a relatively accurate diagnosis about it.

That's where we're putting our focus.

Of course, there are currently many applications that can be used to diagnose patients using this method. But they're all "passive" applications where you should turn on the app and take pictures of yourself.

However, the importance of the software we're creating lies in the "active" part. Home appliances, typically refrigerators and TVs, are used daily by people. We know eventually that everyone opens their fridge at least once a day, even if there's no specific reason, or turn on their TV. According to a 2012 Consumer Electronics Industry Survey, a family of four opens the refrigerator on average 40 times a day and it means 10 times a day per person. Putting their attention on this stat, [6] LG Electronics not only achieved great results with the design of "Magic Space" but also significantly reduced electricity use at home. [7] In addition, Americans open their refrigerators an average of 33 times a day, according to ENERGY STAR, a program run by the U.S. Environmental Protection Agency and the U.S. Department of Energy.

Considering everything cited above, a brief intro of our project: the refrigerator would be equipped with a camera to scan the users' facial expressions when standing face to it. If a risk of stroke is detected, a speaker implemented informs the user of his condition right away.

If this health check service was supported by every home appliance, we could imagine a household that actively protects our health in real-time daily and not just a passive household that passively neglect dangerous health issues.

TABLE I
A LIST OF ROLE ASSIGNMENT

Role	Name	Task Description
Development Manager	Lee Seungsu	Lee Seungsu was responsible for designing the overall concept of the software and finding the basis for various claims.
Software Development	Park Geonryul	Park Geonryul was in charge of the actual implementation of systems as well as machine learning.
Customer	Elia Ayoub	Elia, as a customer and user of the products and servies provided by the team, had to deliver an objective feedback on them accordingly.
Project Analysis	Ryan Jabbour	Ryan was given the task of checking all the documentation to present and the logical development process of the project.

I. Introduction

A. *Motivation*

- **The Problem**

  [8] According to the National Statistical Office of the Republic of Korea, nearly 60,000 of the total 120,000 stroke patients in 2021 were not transferred to the emergency room until more than six hours after the outbreak. Fewer than 15% of the people arrived at the emergency room in less than an hour, and half of them were patients living in the Seoul-Gyeonggi area. [9] The number of patients at risk is increasing in provinces and the medical infrastructure is insufficient, so initial diagnosis or prevention is not possible, and even if it could be, follow-up will inevitably be delayed. After all, time is the lifeblood of a stroke and every minute counts. You need to quickly notice the signs but rarely do patients check signs of stroke daily.

  With the development of technology and healthcare in the world, life expectancy has gradually increased over the years and is approaching a staggering 80 years old. Health is one of the most important factors in a person's life. However, there are cases where the elderly are reluctant to go to the hospital due to their habits or due to their misunderstandings arising from their experiences. There are cases in which the right time to prevent or treat the person had already passed and it was already too late. Then, in this context, not worrying about one's health and not taking preemptive measures can lead to serious social problems.

This problem is not only defined to the older generation. [8] According to the National Medical Center, the incidence of stroke among people in their 20s and 30s is rising every year. The stereotype that stroke is a disease only the elderly can have can often push people to neglect it even when having premonitory symptoms.

Moreover, a growing number of people have started living alone, especially in South Korea, and similar problems can appear in these single-person households. Because of living alone, ones can't point out their unhealthy habits. And if this person gets an acute disease, they won't be able to take the proper measures leading to serious health problems. [10] As the proportion of single-person households in Korea approaches 34.5%, chances for them to recognize acute diseases in the early stages are also decreasing.

In order to become a better society, we must overcome all those problems. The reality is that people's recognition rate of early stroke symptoms is very low. [11] According to the Korea Centers for Disease Control and Prevention, only 54% of all respondents were correct for early stroke symptoms. Although awareness reached a high of 61% during the pandemic in 2019 - presumably because people cared a lot about health issues due to pandemic - it has been low since 2019.

Let's summarize everything. First, people aren't as wary or concerned about strokes as we might think. Second, when a stroke occurs, it is quite rare for patients to arrive at the hospital within the recommended time, and most of them actually live in the metropolitan area where the medical infrastructure doesn't meet quality requirements. Third, the elderly, who are at high risk of getting the disease, are concentrated in provinces with low-quality healthcare institutions, and have low awareness about the disease so it is unlikely for them to respond to premonitory symptoms. Fourth, although the incidence rate of stroke for people in their 20s and 30s is rising, their vigilance is still very low. Fifth, as the number of single-person households is increasing, ones are not able to detect stroke beforehand and initial responses are becoming insufficient.

In order for the passive detectors to be effective, people of all ages must be aware of stroke on their own and check it periodically. However, no method, from promotions to campaigns, seems to be able to enhance this phenomenon. If this was the case, [12] the prognostic indicator for stroke should have been higher.

Therefore, what we need is an active, every day, stroke checker.

- **The Solution**

  As we said at the beginning, [7] people open their refrigerator approximately once a day. Our concept benefits from this habit. There are already many refrigerators equipped with IoT technology, so our idea is to equip our home appliances with cameras. The refrigerator detects the person's face and his landmark through computer vision when he stands in front of it.

  Using BE-FAST diagnostic methods, if a specific facial expression such as paralysis of one facial muscle is detected, "Strike Stroke" will notify the user right away. The notification method would be through a push-message on your phone application (Thin-Q) or by the use of an AI speaker (NUGU).

  Afterwards, it will guide you to the nearest hospital or emergency center where first aid for stroke is available. It will offer users automatic connection to 119 if they approve it.

  Since fixed cameras may not respond appropriately depending on the users' physical characteristics, [13] multi-angle vision technology is applied to detect them from various angles. This creates a true daily active detector, beyond the limits of difference in home structure and physical characteristics of each user.

- **Future Expectations**

  As the artificial intelligence field is rapidly developing and computer vision is a technology that occupies a large proportion of it, it is highly likely to detect user behavior and develop it into various medical diagnosis. If these technologies are included in each of LG Electronics' home appliances, which currently have a huge share compared to other competitors, users will be able to continue to actively detect their diseases in their homes, whether in the living room, the kitchen or even the bedroom. This will allow the home to become an active diagnostic center for individuals rather than just passive living space. If this one day becomes our reality, we expect a very big paradigm shift. [14] We think home diagnostics self care, which is developing recently, is a very important technology field, and the synergy will be great if it is combined with home appliances.

B. Research on Related Materials

- **Project MONAI**

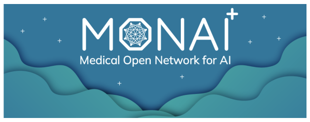


Fig. 1. MONAI project

MONAI is an initiative started by NVIDIA and King's College London to establish an inclusive community of AI researchers to develop and exchange best practices for AI in healthcare. This collaboration has expanded to include academic and industry leaders throughout the medical field.

This project is similar to our project because it is simply analyzing MRI or CT photographs with AI, but the methods used are different.

- **BASLER**

This company actually provides an overall solution for the vision system. Their products support hardware and software at the same time and can analyse images based on machine learning. However, their cameras and sensors are very expensive, so it would be difficult to apply them to home appliances as they are presented in this project.

- **Kaggle Project**

Fig. 2. Kaggle

It is a stroke detection project undertaken by Kaggle. It can be used as an AI model for our project but since the algorithm used in this project is based on 2D images, it differs from the 3D recognition we need to use in our project.

- **Related Papers**

We researched numerous papers in order to study the theoretical part of our project.

1. Multi-Angle detector [15]

This paper introduces lightweight deep network and combining key point feature positioning for multi-angle facial expression recognition. Using robot dog to recognize facial expressions will be affected by distance and angle. To solve this problem, this paper proposes a method for facial expression recognition at different distances and angles, which solved the larger distance and deflection angle of facial expression recognition accuracy and real-time issues.

2. Raspberry Pi Based Emotion Recognition using OpenCV, TensorFlow, and Keras [16]

In this tutorial, they implement an Emotion Recognition System or a Facial Expression Recognition System on a Raspberry Pi 4. The apply a pre-trained model in order to recognize the facial expression of a person from a real-time video stream. The “FER2013” dataset is used to train the model with the help of a VGG-like Convolutional Neural Network (CNN).

3. Connect a Raspberry Pi or other device with AWS [17]

This step-by-step tutorial guides through all the steps you need to take in order to connect a Raspberry Pi or any other device with AWS. It tells you how to set up the device, install the required tools and libraries for the AWS IoT Device SDK, install AWS IoT Device SDK, install and run the sample app, as well as view the messages from the sample app in the AWS IoT console.

4. Realtime Facial Emotion Recognition [18]

This repository demonstrates an end-to-end pipeline for real-time Facial emotion recognition application through full-stack development. The front-end is developed in react.js and the back-end is developed in FastAPI. The emotion prediction model is built with Tensorflow Keras, and for real-time face detection with animation on the front-end, Tensorflow.js have been used.

5. Kaggle FER-2013 DataSet [19]

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image.

The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 28,709 examples and the public test set consists of 3,589 examples.

6. Facial landmarks with dlib, OpenCV, and Python [20]

This post explains line by line a source code and demonstrates in details what are face landmarks and how to detect facial landmarks using dlib, OpenCV, and Python. Also, it introduces alternative facial landmark detectors such as ones coming from the MediaPipe library which is capable of computing a 3D face mesh.

II. Requirements

A. *Training AI model*

It is a process of training an artificial intelligence model based on data. It learns through images with face drooping of stroke patients and images of normal people who do not exist. The image is converted into a Tensor form and flatten through an image preprocessing process. The model should be a classification-capable model, and includes Support Vector Machine, Visual Transformer, and Naive bayesian classifier.

B. *Saving trained AI model*

Since it is difficult to train the AI model on a web server, the model should be trained from the outside based on data and the file is stored in the local computer. Pickle or joblib can be used.

C. *Loading trained AI model*

The stored trained artificial intelligence model should be able to be retrieved from the appropriate location. Because the location to import is a virtual machine, like AWS Lightsail, the 'scp' command to move files from the local computer to the virtual machine will be used.

D. *Classifying Image with trained AI model*

Similar to training an artificial intelligence model, the image preprocessing process comes first. Preprocessed image is input to the trained artificial intelligence model to receive predicted values. At this time, since it is medical information such as stroke, hard-classification such as 0 or 1 is not enough. The probability of each predicted value is also predicted.

E. *Returning the result*

The predicted value returned by trained AI model is sent via web communication. For this purpose, it goes through a process of converting to an appropriate format. The classified results and the probability for each result value must be included and converted to JSON format and transmitted.

F. *Get Image with API*

Receive image files using the functions of the web framework. The image file itself is delivered using the Post method, and is passed as a parameter to the artificial intelligence model through the preprocessing process mentioned above.

G. *Post the result with API*

In 'Returning the result' part above, the predicted value is converted to JSON format and the result value is returned to the place where the Post request was sent. This function is implemented using the functions of the web framework.

H. *Encapsulate the AI model*

For convenience of implementation, the contents mentioned above are gathered and encapsulated into one function in one file.

I. *Run the Web Server*

Deploy the web server by running the above file. Opens to external IP to enable connection.

J. *AWS*

Fig. 3. Amazon Web Service

It is a server for the deployment and training of the artificial intelligence model. It is built on an Ubuntu-based x86-64 architecture. We created a virtual instance using EC2 and configured security settings to make it accessible via an Elastic IP.

K. *Raspberry Pi*

Fig. 4. Raspberry Pi

It controls a camera that captures user photos and enables communication with a web server. The artificial intelligence model is responsible for determining the presence of a stroke based on the photos and relaying the results back to the user. This process can be conveyed either audibly through NUGU speakers or visually through a dashboard. We use this to create a real IoT program.

L. SSH

To enhance coding productivity on Raspberry Pi, we use SSH. This allows us to remotely connect to the Raspberry Pi from our local desktop and use the same IDE for increased efficiency. SSH enables the connection between the Raspberry Pi and the local desktop when they are on the same Wi-Fi network.

M. Camera

To execute our project, a camera is essential. Since we are planning to implement IoT technology using Raspberry Pi, we opted for the highly compatible camera, PiCamera2. This allows us to automatically forward the camera to camera-related functions used in OpenCV.

N. LED module

We use the LED module for privacy protection. When the device connects to the server or the internet, the LED lights up, allowing users to be aware of the current status. This is to prevent crimes resulting from hacking.

O. OpenCV

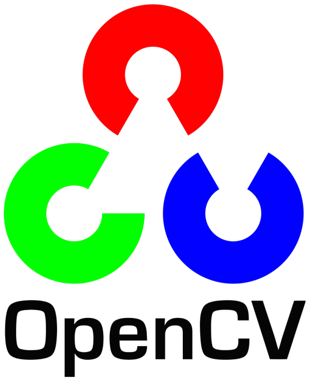


Fig. 5. OpenCV

OpenCV is a powerful library widely used for computer vision and image processing tasks. It allows us to perform various operations, such as reading and saving images. Additionally, it provides the capability to perform color space transformations, which is crucial for efficiency. Resizing images is also possible, and OpenCV offers a wide range of algorithms and functions for tasks like face pattern detection. Furthermore, it allows for feature point extraction and image processing, making it a versatile tool for a variety of tasks.

Fig. 6. TensorFlow

P. Tensorflow Serving

When developing an artificial intelligence model based on TensorFlow, it is necessary for streamlined deployment.

Q. Privacy Protection

Privacy policy is of utmost importance in our project, given that it involves capturing a user's everyday life. We aim to ensure privacy through a two-step approach.

*1) **Background Blur**:* First, we will apply blurring to the background, excluding the face.

*2) **Two-level detection and LED module**:* Second, we will implement a real-time detection algorithm that operates locally without internet or server connectivity in normal circumstances. In the event of a detected risk of stroke, the system will establish a connection to the server, capturing an accurate photo of the user to utilize a trained artificial intelligence model. If connected to the server or the internet, an LED module located next to the camera will activate, allowing the user to visually confirm the device's status.

R. NUGU

Fig. 7. Nugu AI Speaker

Using an artificial intelligence speaker gives the ability to audibly relay stroke information sent from the Raspberry Pi to the user. Additionally, the trigger could be used by the user to initiate photo capture when desired.

S. *Dashboard*

It visually expresses whether you have a stroke or not. It displays stroke information and its probability sent from the Raspberry Pi, along with health-related useful information generated by ChatGPT, to the user. Examples of useful information include foods and lifestyle habits that can be beneficial in the event of a stroke, as well as guidance on what to do if a stroke occurs.

*1) **Probability of Stroke**:* It expresses the probability value of stroke from the value transmitted through the API. It transmits probabilities to users by using as a motif the speed dashboard of a car.

*2) **User's Image**:* It displays the photo of the user, allowing the user to objectively assess their own condition and potentially raise awareness of their health.

*3) **Treatment Options**:* It shows treatment options obtained using ChatGPT. If the user has a high probability of stroke, these treatment options can be highlighted in order to draw even more attention to the user.

*4) **Prevention and Information**:* It tells users how to prevent themselves of gettting a stroke obtained and this information is also obtained by using ChatGPT. Even if the probability of stroke is low, it still informs the user of preventions and informs him on the behavioral guidelines to have in suspicion of stroke. This enables daily active health care.

T. *ChatGPT*

Fig. 8. ChatGPT

It is a generative AI that generates useful information based on the user's stroke status and probability. It uses API to send questions and receive answers. The answer is notified to the user through the Dashboard or the NUGU Speaker.

*1) **Question**:* When asking questions related to strokes to ChatGPT, you only receive information suggesting an immediate need to go to the hospital. By extracting the probability from the results returned by the artificial intelligence model, we can inquire directly about treatment options, preventive measures, and guidance on selecting a hospital.

*2) **Answer**:* After using the question format above, the answer received from ChatGPT can be passed to the Dashboard or the NUGU Speaker then passed to the user in a visual or auditory representation.

U. *Database*

To enhance the accuracy of the artificial intelligence model, a database for storing images may be required. However, since this can involve sensitive personal information, the project can also be implemented without a database.

III. Development Environment

A. *OS*

We set Ubuntu, a type of Linux, as the default OS. This Ubuntu is the OS of the virtual machine required when the learned AI model is deployed through a web server. I chose Ubuntu because it is a CLI-based, lightweight, fast, and familiar operating system for programmers.

B. *Raspberry Pi*

Our goal is to integrate artificial intelligence into home appliances to create a smart IoT system. To achieve true IoT implementation, we have chosen to use Raspberry Pi. We aim to implement genuine IoT technology by controlling home appliances with an independent computer. While we can develop the intended program on a desktop, we believe it's not suitable for IoT because of the significant differences in performance between a desktop and the board used in home appliances.

In practice, we found that a program that worked well on a desktop did not run smoothly on a Raspberry Pi due to several factors. These factors include differences in the operating system, such as bit and Linux version, variations in default libraries, differences in the camera recognition mechanism, variations in hardware performance, such as CPU and GPU, heat-related issues, and performance optimization. By using Raspberry Pi, we have come to realize aspects that were not carefully considered during desktop programming, allowing us to create a program truly suited for home appliances.

To program on Raspberry Pi, we utilized SSH. We accessed the Raspberry Pi via SSH from Mac OS and wrote the code. The reason for using SSH was convenience. Programming directly on Raspberry Pi would require installing a separate editor and libraries for efficient coding. Additionally, the response time for keyboard input is not very fast, making it less convenient. Therefore, we connected to the same Wi-Fi network and used the SSH server via the designated IP address.

Fig. 9. Python

C. *Languages*

*1) **Python**:* It is the language used to design AI models and is mainly used for programming. The reasons for creating an AI model in Python are as follows.

First, various libraries. When designing various AI models, you can conveniently use useful libraries through Python. For example, there are several libraries available, such as scikit-learn that can be used when implementing machine learning models such as support vector machine, and numpy, which helps with various numerical calculations.

Second, deep learning frameworks are easy to use. By using frameworks that help implement deep learning, such as PyTorch or Keras, you can modularize each step and handle back-propagation especially easily. Lastly, you can create pythonic code by using FastAPI in the API required to communicate with the web server. For this reason, Python was used as the main language to implement the AI model.

TABLE II
A VERSION OF SOFTWARE/LANGUAGE/TOOL

Name	Version
Raspberry Pi	Raspi 4B 4GB
Rasbian	Devian BookWorm
Camera	PiCamera 2
Ubuntu	22.04
Uvicorn	0.23.2 with CPython 3.10.12 on Linux
Python	3.10.12
OpenCV	4.5.5
Dlib	19.23.2
NUGU	SKT Jan. 18 Release

D. *Environment Resources*

*1) **AWS Lightsail**:*

- OS: Ubuntu 22.04.1 LTS (GNU/Linux 6.2.0-1014-aws x86_64)
- CPU: Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz
- RAM: 1GB RAM
- Storage: 40GB SSD

*2) **Google Colab Pro**:*

- GPU: T4 GPU
- GPU RAM: 15GB
- System RAM: 12GB

E. *Estimated Costs*

- AWS Lightsail: $3.5/month
- Colab Pro: $9.99/month
- Raspberry Pi 4B 4GB: $70

IV. SOFTWARE IN USE

A. *AWS*

We carried out tasks such as building a server with Amazon Web Service and used the functions below.

*1) **AWS Lightsail**:* We built a virtual environment to run a web server using AWS Lightsail. To set up a virtual environment in AWS, you can also use EC2, but the reasons for using AWS Lightsail are as follows.
First, it is a small-scale project. It is a simpler version than EC2 and has relatively limitations, but it is sufficient to operate as a web server and is more stable and simple than those with many unused functions.
Second, it is a monthly billing system. EC2 charges for the amount of instances used, but Lightsail operates at a fixed rate, so we decided that Lightsail would be more suitable as a server virtual machine that is turned on 24 hours a day. For the above reasons, the virtual machine used to deploy the AI model used AWS's Lightsail.

*2) **AWS** :*

B. *scp*

scp stands for "Secure Copy Protocol," and it is a command-line tool used to securely transfer files and directories between a local host and a remote host or between two remote hosts over a network. scp is a part of the SSH (Secure Shell) suite of network protocols and provides a secure way to copy files and data from one location to another.

C. *joblib*

joblib is a Python library used for serialization, especially for efficiently storing and loading Python objects, often used for handling large Numpy arrays or complex data structures. It is commonly employed in data analysis, machine learning, and scientific research to save and load Python objects or share objects between processes. There are many advantages:
First, Efficient Serialization. joblib can serialize Python objects into binary format and store them on disk efficiently. This is particularly useful for handling large data.
Second, Numpy Array Support. joblib supports a variety of Python data structures, including Numpy arrays, and it can compress and store them. We use 'joblib' as below usage. Storing trained models and their parameters to reuse them later or for model deployment.

D. scikit-learn

[21] scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a NumFOCUS fiscally sponsored project.

*1) **skimage.io**:* scikit-image (a.k.a. skimage) is a collection of algorithms for image processing and computer vision. The main package of skimage only provides a few utilities for converting between image data types; for most features, you need to import one of the following subpackages:

E. PIL

[22] Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux. The latest version of PIL is 1.1.7, was released in September 2009 and supports Python 1.5.2–2.7. Development of the original project, known as PIL, was discontinued in 2011.
Subsequently, a successor project named Pillow forked the PIL repository and added Python 3.x support This fork has been adopted as a replacement for the original PIL in Linux distributions including Debian[5] and Ubuntu (since 13.04).

F. NumPy

[23] NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The predecessor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers.

NumPy is open-source software and has many contributors. NumPy is a NumFOCUS fiscally sponsored project. NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents due to the absence of compiler optimization. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays; using these requires rewriting some code, mostly inner loops, using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality. Although matlab can perform sparse matrix operations, numpy alone cannot perform such operations and requires the use of the scipy.sparse library. Internally, both MATLAB and NumPy rely on BLAS and LAPACK for efficient linear algebra computations.

Python bindings of the widely used computer vision library OpenCV utilize NumPy arrays to store and operate on data. Since images with multiple channels are simply represented as three-dimensional arrays, indexing, slicing or masking with other arrays are very efficient ways to access specific pixels of an image. The NumPy array as universal data structure in OpenCV for images, extracted feature points, filter kernels and many more vastly simplifies the programming workflow and debugging.

G. Google Colab Pro

Google Colab is a cloud-based Jupyter notebook environment provided by Google. This free service allows you to write and run Python code through a web browser, and can be used for a variety of tasks, including data analysis, machine learning model development, and research.

By utilizing this Google Colab, you are not restricted by the performance and capacity limitations of individual users' computers and can proceed with AI model learning independently. In addition, it provides an environment where GPU or TPU can be remotely connected and used, so it is useful for computationally intensive operations such as deep learning model training. Lastly, in order to write Python code on a personal user computer, there is the need to install a virtual environment and various libraries. With Google Colab, users can free themselves from this hassle and focus entirely on implementation.

H. FastAPI

FastAPI is a web framework for building fast, modern web applications and APIs using Python. FastAPI is easy to use, has excellent performance, and supports Python 3.6 or higher. The reasons for choosing FastAPI when building a web server are as follows.

First, the Python Framework. FastAPI is closely related to Python, the language used to create AI models with the Python Web Framework. Therefore, bugs can be reduced by maintaining consistency when creating a web server.

Second, you can create interactive API documentation. It is convenient because developers who create AI models can also carry out debugging on their own through interactive API documents. For this reason, I chose FastAPI.

*1) **File**:* The File class in FastAPI is used to represent an uploaded file in your application. It's commonly used as a parameter in your route's function to handle file uploads. Here is an example python code:

```
from fastapi import FastAPI, File

app = FastAPI()

@app.post("/uploadfile/")
async def upload_file(file: UploadFile):
    # Do something with the uploaded file
    return {"filename": file.filename}
```

Attributes of 'File'

- filename: This attribute contains the name of the uploaded file.
- content_type: The MIME content type of the uploaded file.
- file: The actual file data, which can be read or processed.

Handling the Uploaded file You can perform various operations on the uploaded file using the attributes. For example, you can save the file to disk, read its contents, check the content type, or perform any other custom logic.

*2) **UploadFile**:* The UploadFile class is a specialized class provided by FastAPI for handling file uploads. It contains additional functionality for managing the uploaded files. Here is an example python code:

```
from fastapi import FastAPI, UploadFile

app = FastAPI()

@app.post("/uploadfile/")
async def upload_file(file: UploadFile):
    # Do something with the uploaded file
    return {"filename": file.filename}
```

Attributes of 'UploadFile'

- filename: This attribute contains the name of the uploaded file.
- content_type: The MIME content type of the uploaded file.
- file: The actual file data, which can be read or processed.
- read(): Allows you to read the content of the uploaded file as bytes.
- save(): You can use this method to save the uploaded file to a specified location on your server.

*I. **Uvicorn***

Uvicorn is a popular ASGI (Asynchronous Server Gateway Interface) server that is commonly used to deploy web applications, particularly FastAPI applications, in the Python ecosystem. It's designed for high-performance, asynchronous web serving, and it's often used in conjunction with ASGI frameworks like FastAPI and Starlette.

*J. **Deep learning Framework***

When implementing an AI model, if you create a deep learning model, the scikit-learn library is not enough. A deep learning framework that modularizes various layers and facilitates back-propagation is needed. In this project, Pytorch was used as shown below.

*1) **Pytorch**:* PyTorch is an open source machine learning library for deep learning and machine learning, primarily used to develop and train models using the Python language. PyTorch is widely used by many researchers and companies. The reasons for using Pytorch among various Deep Learning Frameworks are as follows.

First, the dynamic computation graph. One of the best features of PyTorch is its use of dynamic computation graphs. This refers to the way the graph is constructed when defining and calculating models. This makes it much easier to dynamically change and debug models.

Second, AI model learning using GPU. When implementing a machine learning model using scikit-learn, learning was performed using only the CPU. PyTorch provides the ability to accelerate model training and inference using NVIDIA GPUs, enabling faster training.

Third, automatic differentiation is possible. PyTorch supports automatic differentiation, making it easy to calculate gradients. This makes it easier to implement learning algorithms such as back-propagation and gradient descent. In fact, when implementing the Transformer algorithm, which will be described later, it was implemented using Pytorch, which has the above strengths.

*K. **Support Vector Machine***

[24] In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis.

SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or VC theory proposed by Vapnik (1982, 1995) and Chervonenkis (1974). Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier (although methods such as Platt scaling exist to use SVM in a probabilistic classification setting).

SVM maps training examples to points in space so as to maximise the width of the gap between the two categories. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

The support vector clustering algorithm, created by Hava Siegelmann and Vladimir Vapnik, applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data. These data sets require unsupervised learning approaches, which attempt to find natural clustering of the data to groups and, then, to map new data according to these clusters.

*1) **Motivation**:* Classifying data is a common task in machine learning.

Suppose some given data points each belong to one of two classes, and the goal is to decide which class a "new" data point will be in. In the case of support vector machines, a data point is viewed as a p - dimensional vector (a list of p numbers), and we want to know whether we can separate such points with a $(p-1)$-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the 'maximum-margin hyperplane' and the linear classifier it defines is known as a 'margin classifier'; or equivalently, the perceptron of optimal stability.

More formally, a support vector machine constructs a hyperplane or set of hyperplanes in a high or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the classifier. A lower generalization error means that the implementer is less likely to experience overfitting.

Whereas the original problem may be stated in a finite-dimensional space, it often happens that the sets to discriminate are not linearly separable in that space. For this reason, it was proposed that the original finite-dimensional space be mapped into a much higher-dimensional space, presumably making the separation easier in that space. To keep the computational load reasonable, the mappings used by SVM schemes are designed to ensure that dot products of pairs of input data vectors may be computed easily in terms of the variables in the original space, by defining them in terms of a kernel function $k(x, y)$ selected to suit the problem. The hyperplanes in the higher-dimensional space are defined as the set of points whose dot product with a vector in that space is constant, where such a set of vectors is an orthogonal (and thus minimal) set of vectors that defines a hyperplane. The vectors defining the hyperplanes can be chosen to be linear combinations with parameters α_i of images of feature vectors x_i that occur in the data base. With this choice of a hyperplane, the points x in the feature space that are mapped into the hyperplane are defined by the relation $\sum_i \alpha_i k(x_i, x) = \textbf{constant}$. Note that if $k(x, y)$ becomes small as y grows further away from x, each term in the sum measures the degree of closeness of the test point x to the corresponding data base point x_i. In this way, the sum of kernels above can be used to measure the relative nearness of each test point to the data points originating in one or the other of the sets to be discriminated. Note the fact that the set of points x mapped into any hyperplane can be quite convoluted as a result, allowing much more complex discrimination between sets that are not convex at all in the original space.

*2) **Linear SVM**:* We are given a training dataset of n points of the form

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n),$$

where the y_i are either 1 or 1, each indicating the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a p-dimensional real vector. We want to find the "maximum-margin hyperplane" that divides the group of points $\mathbf{x}_i$ for which $y_i = 1$ from the group of points for which $y_i = -1$, which is defined so that the distance between the hyperplane and the nearest point $\mathbf{x}_i$ from either group is maximized.

Any hyperplane can be written as the set of points $\mathbf{x}$ satisfying

$$\mathbf{w}^{\mathsf{T}}\mathbf{x} - b = 0,$$

where $\mathbf{w}$ is the (not necessarily normalized) normal vector to the hyperplane. This is much like Hesse normal form, except that $\mathbf{w}$ is not necessarily a unit vector. The parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

Hard-margin

If the training data is linearly separable, we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible. The region bounded by these two hyperplanes is called the ”margin”, and the maximum-margin hyperplane is the hyperplane that lies halfway between them. With a normalized or standardized dataset, these hyperplanes can be described by the equations $\mathbf{w}^\mathsf{T}\mathbf{x} - b = 1$ (anything on or above this boundary is of one class, with label 1) and $\mathbf{w}^\mathsf{T}\mathbf{x} - b = -1$(anything on or below this boundary is of the other class, with label 1).

Geometrically, the distance between these two hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, so to maximize the distance between the planes we want to minimize $\|\mathbf{w}\|$. The distance is computed using the distance from a point to a plane equation. We also have to prevent data points from falling into the margin, we add the following constraint: for each i either

$$\mathbf{w}^\mathsf{T}\mathbf{x}_i - b \geq 1\,,\ \text{if } y_i = 1,$$

or

$$\mathbf{w}^\mathsf{T}\mathbf{x}_i - b \leq -1\,,\ \text{if } y_i = -1.$$

These constraints state that each data point must lie on the correct side of the margin.

This can be rewritten as

$$y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n. \tag{1}$$

We can put this together to get the optimization problem:

$$\underset{\mathbf{w},\, b}{\text{minimize}} \quad \|\mathbf{w}\|_2^2 \tag{2}$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top\mathbf{x}_i - b) \geq 1 \quad \forall i \in \{1, \dots, n\} \tag{3}$$

The $\mathbf{w}$ and b that solve this problem determine our classifier, $\mathbf{x} \mapsto \operatorname{sgn}(\mathbf{w}^\mathsf{T}\mathbf{x} - b)$ where $\operatorname{sgn}(\cdot)$ is the sign function.

An important consequence of this geometric description is that the max-margin hyperplane is completely determined by those $\mathbf{x}_i$ that lie nearest to it. These $\mathbf{x}_i$ are called ’support vectors’.

Soft-margin

To extend SVM to cases in which the data are not linearly separable, the ’hinge loss’ function is helpful

$$\max\left(0, 1 - y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b)\right).$$

Note that y_i is the ’i’-th target (i.e., in this case, 1 or 1), and $\mathbf{w}^\mathsf{T}\mathbf{x}_i - b$ is the ”i”-th output.

This function is zero if the constraint in (1) is satisfied, in other words, if $\mathbf{x}_i$ lies on the correct side of the margin. For data on the wrong side of the margin, the function’s value is proportional to the distance from the margin.

The goal of the optimization then is to minimize

$$\lambda\|\mathbf{w}\|^2 + \left[\frac{1}{n}\sum_{i=1}^n \max\left(0, 1 - y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b)\right)\right],$$

where the parameter $\lambda > 0$ determines the trade-off between increasing the margin size and ensuring that the $\mathbf{x}_i <$ lie on the correct side of the margin. By deconstructing the hinge loss, this optimization problem can be massaged into the following:

$$\underset{\mathbf{w},\, b,\, \zeta}{\text{minimize}} \quad \|\mathbf{w}\|_2^2 + C\sum_{i=1}^n \zeta_i \tag{4}$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top\mathbf{x}_i - b) \geq 1 - \zeta_i, \quad \zeta_i \geq 0 \quad \forall i \in \{1, \dots, n\} \tag{5}$$

Thus, for large values of C, it will behave similar to the hard-margin SVM, if the input data are linearly classifiable, but will still learn if a classification rule is viable or not. (λ is inversely related to C, e.g. in ’LIBSVM’.)

*3) **Non-linear kernel:*** We use rbf kernel to deal with non-linear problem. These are introduction about kernel method. The algorithm is formally similar, except that every dot product is replaced by a nonlinear kernel function. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high-dimensional; although the classifier is a hyperplane in the transformed feature space, it may be nonlinear in the original input space.

It is noteworthy that working in a higher-dimensional feature space increases the generalization error of support vector machines, although given enough samples the algorithm still performs well.

Some common kernels include:
* Polynomial (homogeneous):

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j)^d$$

Particularly, when $d = 1$, this becomes the linear kernel.

* Polynomial(inhomogeneous):

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + r)^d$$

* Gaussian radial basis function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right)$$

for $\gamma > 0$. Sometimes parametrized using $\gamma = 1/(2\sigma^2)$.

* Sigmoid function (Hyperbolic tangent):

$$k(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\kappa\mathbf{x_i} \cdot \mathbf{x}_j + c)$$

for some (not every) $\kappa > 0$ and $c < 0$

*4) **Computing the SVM classifier:*** Computing the (soft-margin) SVM classifier amounts to minimizing an expression of the form

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b)\right)\right] + \lambda\|\mathbf{w}\|^2$$

.

We focus on the soft-margin classifier since, as noted above, choosing a sufficiently small value for λ yields the hard-margin classifier for linearly classifiable input data. The classical approach, which involves reducing above term to a quadratic programming problem, is detailed below. Then, more recent approaches such as sub-gradient descent and coordinate descent will be discussed.

Primal

Minimizing above term can be rewritten as a constrained optimization problem with a differentiable objective function in the following way.

For each $i \in \{1, \ldots, n\}$ we introduce a variable $\zeta_i = \max\left(0, 1 - y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b)\right)$. Note that ζ is the smallest nonnegative number satisfying $y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b) \geq 1 - \zeta_i$.

Thus we can rewrite the optimization problem as follows

$$\text{minimize } \frac{1}{n}\sum_{i=1}^{n}\zeta_i + \lambda\|\mathbf{w}\|^2 \tag{6}$$

$$\text{subject to } y_i\left(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b\right) \geq 1 - \zeta_i \text{ and } \zeta_i \geq 0, \text{ for all } i. \tag{7}$$

This is called the ”primal” problem.

Dual

By solving for the Lagrangian dual of the above problem, one obtains the simplified problem

$$\text{maximize } f(c_1 \ldots c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i(\mathbf{x}_i^\mathsf{T}\mathbf{x}_j) y_j c_j, \tag{8}$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i. \tag{9}$$

This is called the ”dual” problem. Since the dual maximization problem is a quadratic function of the c_i subject to linear constraints, it is efficiently solvable by quadratic programming algorithms.

Here, the variables c_i are defined such that

$$\mathbf{w} = \sum_{i=1}^{n} c_i y_i \mathbf{x}_i.$$

Moreover, $c_i = 0$ exactly when $\mathbf{x}_i$ lies on the correct side of the margin, and $0 < c_i < (2n\lambda)^{-1}$ when $\mathbf{x}_i$lies on the margin's boundary. It follows that $\mathbf{w}$ can be written as a linear combination of the support vectors.

The offset, b, can be recovered by finding an $\mathbf{x}_i$ on the margin's boundary and solving

$$y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i - b) = 1 \iff b = \mathbf{w}^\mathsf{T}\mathbf{x}_i - y_i.$$

(Note that $y_i^{-1} = y_i$ since $y_i = \pm 1$.)

Kernel Trick

Suppose now that we would like to learn a nonlinear classification rule which corresponds to a linear classification rule for the transformed data points $\varphi(\mathbf{x}_i)$. Moreover, we are given a kernel function k which satisfies $k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)$.

We know the classification vector $\mathbf{w}$in the transformed space satisfies

$$\mathbf{w} = \sum_{i=1}^{n} c_i y_i \varphi(\mathbf{x}_i),$$

where, the c_i are obtained by solving the optimization problem

$$\max\ f(\mathbf{c}) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i(\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)) y_j c_j \tag{10}$$

$$= \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i k(\mathbf{x}_i, \mathbf{x}_j) y_j c_j \tag{11}$$

$$\text{subject to } \sum_{i=1}^{n} c_i y_i = 0, \text{ and } 0 \leq c_i \leq \frac{1}{2n\lambda} \text{ for all } i. \tag{12}$$

The coefficients c_i can be solved for using quadratic programming, as before. Again, we can find some index i such that $0 < c_i < (2n\lambda)^{-1}$, so that $\varphi(\mathbf{x}_i)$lies on the boundary of the margin in the transformed space, and then solve

$$b = \mathbf{w}^\mathsf{T}\varphi(\mathbf{x}_i) - y_i = \left[\sum_{j=1}^{n} c_j y_j \varphi(\mathbf{x}_j) \cdot \varphi(\mathbf{x}_i)\right] - y_i \tag{13}$$

$$= \left[\sum_{j=1}^{n} c_j y_j k(\mathbf{x}_j, \mathbf{x}_i)\right] - y_i. \tag{14}$$

Finally,

$$\mathbf{z} \mapsto (\mathbf{w}^\mathsf{T}\varphi(\mathbf{z}) - b) = \left(\left[\sum_{i=1}^{n} c_i y_i k(\mathbf{x}_i, \mathbf{z})\right] - b\right).$$

*L. **Transformer***

[25] A transformer is a deep learning architecture, initially proposed in 2017, that relies on the parallel multi-head attention mechanism. It is notable for requiring less training time than previous recurrent neural architectures, such as long short-term memory (LSTM), and its later variation has been prevalently adopted for training large language models on large (language) datasets, such as the Wikipedia corpus

and Common Crawl, by virtue of the parallelized processing of input sequence. Input text is split into n-grams encoded as tokens and each token is converted into a vector via looking up from a word embedding table. At each layer, each token is then contextualized within the scope of the context window with other (unmasked) tokens via a parallel multi-head attention mechanism allowing the signal for key tokens to be amplified and less important tokens to be diminished.

This architecture is now used not only in natural language processing and computer vision, but also in audio and multi-modal processing. It has also led to the development of pre-trained systems, such as generative pre-trained transformers (GPTs) and BERT (Bidirectional Encoder Representations from Transformers).

*1) **Architecture**:* All transformers have the same primary components:

- Tokenizers, which convert text into tokens.
- A single embedding layer, which convert tokens and positions of the tokens into vector representations.
- Transformer layers, which carry out repeated transformations on the vector representations, extracting more and more linguistic information. These consist of alternating attention and feedforward layers.
- (optional) Un-embedding layer, which converts the final vector representations back to a probability distribution over the tokens.

Transformer layers can be one of two types, ”encoder” and ”decoder”. In the original paper both of them were used, while later models included only one type of them. BERT is an example of encoder-only model; GPT are decoder-only models.

Input
The input text is parsed into tokens by a tokenizer, most often a byte pair encoding tokenizer, and each token is converted into a vector via looking up from a word embedding table. Then, positional information of the token is added to the word embedding.

Encoder/decoder architecture
Like earlier seq2seq models, the original transformer model used an ”'encoder/decoder”' architecture. The encoder consists of encoding layers that process the input tokens iteratively one layer after another, while the decoder consists of decoding layers that iteratively process the encoder's output as well as the decoder output's tokens so far.

The function of each encoder layer is to generate contextualized token representations, where each representation corresponds to a token that ”mixes” information from other input tokens via self-attention mechanism. Each decoder layer contains two attention sublayers:
(1) cross-attention for incorporating the output of encoder (contextualized input token representations), and (2) self-attention for ”mixing” information among the input tokens to the decoder (i.e., the tokens generated so far during inference time).

Both the encoder and decoder layers have a [[Feedforward neural network—feed-forward neural network]] for additional processing of the outputs and contain residual connections and layer normalization steps.

Scaled dot-product attention
The transformer building blocks are scaled dot-product attention units. For each attention unit, the transformer model learns three weight matrices: the query weights W_Q, the key weights W_K, and the value weights W_V. For each token i, the input token representation x_i is multiplied with each of the three weight matrices to produce a query vector $q_i = x_i W_Q$, a key vector $k_i = x_i W_K$, and a value vector $v_i = x_i W_V$. Attention weights are calculated using the query and key vectors: the attention weight a_{ij} from token $i$to token $j$is the dot product between q_i and k_j. The attention weights are divided by the square root of the dimension of the key vectors, $\sqrt{d_k}$, which stabilizes gradients during training, and passed through a softmax which normalizes the weights.

The fact that W_Q and $W_K$are different matrices allows attention to be non-symmetric: if token i attends to token j(i.e. $q_i \cdot k_j$ is large), this does not necessarily mean that token $j$will attend to token i (i.e. $q_j \cdot k_i$ could be small). The output of the attention unit for token $i$is the weighted sum of the value vectors of all tokens, weighted by a_{ij}, the attention from token i to each token.

The attention calculation for all tokens can be expressed as one large matrix calculation using the softmax, which is useful for training due to computational matrix operation optimizations that quickly compute matrix operations. The matrices Q, K and V are defined as the matrices where the ith rows are vectors q_i, k_i, and v_i respectively. Then we can represent the attention as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\sqrt{d_k}}\right) V \tag{15}$$

where softmax is taken over the horizontal axis.

Multi-head attention
One set of (W_Q, W_K, W_V) matrices is called an ”attention head”, and each layer in a transformer model has multiple attention heads. While each attention head attends to the tokens that are relevant to each token, multiple attention heads allow the model to do this for different definitions of ”relevance”. In addition, the influence field representing

relevance can become progressively dilated in successive layers. Many transformer attention heads encode relevance relations that are meaningful to humans. For example, some attention heads can attend mostly to the next word, while others mainly attend from verbs to their direct objects.

The computations for each attention head can be performed in parallel, which allows for fast processing. The outputs for the attention layer are concatenated to pass into the feed-forward neural network layers.

Concretely, let the multiple attention heads be indexed by i, then we have

$$\begin{aligned} &\text{MultiheadedAttention}(Q, K, V) \\ &= \text{Concat}_{i\in[\#heads]}(\text{Attention}(XW_i^Q, XW_i^K, XW_i^V))W^O \end{aligned} \tag{16}$$

where the matrix X is the concatenation of word embeddings, and the matrices W_i^Q, W_i^K, W_i^V are ”projection matrices” owned by individual attention head i, and W^O is a final projection matrix owned by the whole multi-headed attention head.

Masked attention

It may be necessary to cut out attention links between some word-pairs. For example, the decoder for token position t should not have access to token position $t + 1$. This may be accomplished before the softmax stage by adding a mask matrix Mthat is $-\infty$ at entries where the attention link must be cut, and 0 at other places:

Encoder

Each encoder consists of two major components: a self-attention mechanism and a feed-forward neural network. The self-attention mechanism accepts input encodings from the previous encoder and weights their relevance to each other to generate output encodings. The feed-forward neural network further processes each output encoding individually. These output encodings are then passed to the next encoder as its input, as well as to the decoders.

The first encoder takes positional information and embeddings of the input sequence as its input, rather than encodings. The positional information is necessary for the transformer to make use of the order of the sequence, because no other part of the transformer makes use of this.

The encoder is bidirectional. Attention can be placed on tokens before and after the current token. Tokens are used instead of words to account for polysemy.

Positional encoding

A positional encoding is a fixed-size vector representation that encapsulates the relative positions of tokens within a target sequence: it provides the transformer model with information about ”where” the words are in the input sequence.

The positional encoding is defined as a function of type $f : \mathbb{R} \to \mathbb{R}^d; d \in \mathbb{Z}, d > 0$, where $d$is a positive even integer. The full positional encoding – as defined in the original paper – is given by the equation:

$$(f(t)_{2k}, f(t)_{2k+1}) = (\sin(\theta), \cos(\theta)) \quad \forall k \in \{0, 1, \ldots, d/2-1\}$$

where $\theta = \frac{t}{r^k}, r = N^{2/d}$.

Here, $N$is a free parameter that should be significantly larger than the biggest $k$that would be input into the positional encoding function. In the original paper, the authors chose $N = 10000$.

The function is in a simpler form when written as a complex function of type $f : \mathbb{R} \to \mathbb{C}^{d/2}$

$$f(t) = \left(e^{it/r^k}\right)_{k=0,1,\ldots,\frac{d}{2}-1}$$

where $r = N^{2/d}$.

The main reason the authors chose this as the positional encoding function is that it allows one to perform shifts as linear transformations:

$$f(t + \Delta t) = \text{diag}(f(\Delta t))f(t)$$

where $\Delta t \in R$ is the distance one wishes to shift. This allows the transformer to take any encoded position, and find the encoding of the position n-steps-ahead or n-steps-behind, by a matrix multiplication.

By taking a linear sum, any convolution can also be implemented as linear transformations:

$$\sum_j c_j f(t + \Delta t_j) = \left(\sum_j c_j \,\text{diag}(f(\Delta t_j))\right) f(t)$$

for any constants c_j. This allows the transformer to take any encoded position and find a linear sum of the encoded locations of its neighbors. This sum of encoded positions, when fed into the attention mechanism, would create attention weights on its neighbors, much like what happens in a convolutional neural network language model. In the author’s words, ”we hypothesized it would allow the model to easily learn to attend by relative position”.

In typical implementations, all operations are done over the real numbers, not the complex numbers, but since complex multiplication can be implemented as real 2-by-2 matrix multiplication, this is a mere notational difference.

Decoder

Each decoder consists of three major components: a self-attention mechanism, an attention mechanism over the encodings, and a feed-forward neural network. The decoder functions in a similar fashion to the encoder, but an additional attention mechanism is inserted which instead draws relevant information from the encodings generated by the encoders.

This mechanism can also be called the ”encoder-decoder attention”.

Like the first encoder, the first decoder takes positional information and embeddings of the output sequence as its input, rather than encodings. The transformer must not use the current or future output to predict an output, so the output sequence must be partially masked to prevent this reverse information flow. This allows for autoregressive text generation. For all attention heads, attention can’t be placed on following tokens. The last decoder is followed by a final linear transformation and softmax layer, to produce the output probabilities over the vocabulary.

All members of OpenAI’s GPT series have a decoder-only architecture.

*2) **Attention:*** [26] Machine learning-based attention is a mechanism mimicking cognitive attention. It calculates ”soft” weights for each word, more precisely for its embedding, in the context window. It can do it either in parallel (such as in transformers) or sequentially (such as recurrent neural networks). ”Soft” weights can change during each runtime, in contrast to ”hard” weights, which are (pre-)trained and fine-tuned and remain frozen afterwards.

Attention was developed to address the weaknesses of recurrent neural networks, where words in a sentence are slowly processed one at a time. Recurrent neural networks favor more recent words at the end of a sentence while earlier words fade away in volatile neural activations. Attention gives all words equal access to any part of a sentence in a faster parallel scheme and no longer suffers the wait time of serial processing. Earlier uses attached this mechanism to a serial recurrent neural network’s language translation system (below), but later uses in Transformers large language models removed the recurrent neural network and relied heavily on the faster parallel attention scheme.

Core calculation
The attention network was designed to identify the highest correlations amongst words within a sentence, assuming that it has learned those patterns from the training corpus. This correlation is captured in neuronal weights through back-propagation from unsupervised pretraining.

The example below shows how correlations are identified once a network has been trained and has the right weights. When looking at the word ”that” in the sentence ”see that girl run”, the network should be able to identify ”girl” as a highly correlated word. For simplicity this example focuses on the word ”that”, but in actuality all words receive this treatment in parallel and the resulting soft-weights and context vectors are stacked into matrices for further task- specific use.

The query vector is compared (via dot product) with each word in the keys. This helps the model discover the most relevant word for the query word. In this case ”girl” was determined to be the most relevant word for ”that”. The result (size 4 in this case) is run through the softmax function, producing a vector of size 4 with probabilities summing to 1. Multiplying this against the value matrix effectively amplifies the signal for the most important words in the sentence and diminishes the signal for less important words.

The structure of the input data is captured in the $Q_{\mathbf{w}}$and $K_{\mathbf{w}}$ weights, and the $V_{\mathbf{w}}$ weights express that structure in terms of more meaningful features for the task being trained for. For this reason, the attention head components are called Query (Q), Key (K), and Value (V)—a loose and possibly misleading analogy with relational database systems.

Note that the context vector for ”that” does not rely on context vectors for the other words; therefore the context vectors of all words can be calculated using the whole matrix math—X, which includes all the word embeddings, instead of a single word’s embedding vector **x** in the formula above, thus parallelizing the calculations. Now, the softmax can be interpreted as a matrix softmax acting on separate rows. This is a huge advantage over recurrent networks which must operate sequentially.

*3) **ViT:*** [27] ”’Vision Transformer”’ (”’ViT”’) is a transformer designed for computer vision. Transformers were introduced in 2017, The basic structure is to break down input images as a series of patches, then tokenized, before applying the tokens to a standard Transformer architecture.

The attention mechanism in a ViT repeatedly transforms representation vectors of image patches, incorporating more and more semantic relations between image patches in an image. This is analogous to how in natural language processing, as representation vectors flow through a Transformers, they incorporate more and more semantic relations between words, from syntax to semantics.

ViT has found applications in image recognition, image segmentation, and autonomous driving.

Architecture
The basic architecture, used by the original 2020 paper, is as follows. In summary, it is a BERT-like encoder-only Transformer. The input image is of type $\mathbb{R}^{H\times W\times C}$, where H, W, C are height, width, channel RGB. It is then split into square-shaped patches of type $\mathbb{R}^{P\times P\times C}$.

For each patch, the patch is pushed through a linear operator, to obtain a vector (”patch embedding”). The position of the patch is also transformed into a vector by ”position encoding”. The two vectors are added, then pushed through several Transformer encoders.

Classification

The above architecture turns an image into a sequence of vector representations. To use the vector representation for downstream applications, one needs to add some network modules on top of it.

For example, to use it for classification, one can add a shallow MLP on top of it that outputs a probability distribution over classes. The original paper uses a linear-GeLU-linear-softmax network.

Vision Transformer

Transformers found their initial applications in natural language processing tasks, as demonstrated by language models such as BERT (language model) and GPT-3. By contrast the typical image processing system uses a convolutional neural network (CNN). Well-known projects include Xception, ResNet, EfficientNet, DenseNet, and Inception

Transformers measure the relationships between pairs of input tokens (words in the case of text strings), termed attention. The cost is quadratic in the number of tokens. For images, the basic unit of analysis is the pixel. However, computing relationships for every pixel pair in a typical image is prohibitive in terms of memory and computation. Instead, ViT computes relationships among pixels in various small sections of the image (e.g., 16x16 pixels), at a drastically reduced cost. The sections (with positional embeddings) are placed in a sequence. The embeddings are learnable vectors. Each section is arranged into a linear sequence and multiplied by the embedding matrix. The result, with the position embedding is fed to the transformer.

As in the case of BERT, a fundamental role in classification tasks is played by the class token. A special token that is used as the only input of the final MLP Head as it has been influenced by all the others.

The architecture for image classification is the most common and uses only the Transformer Encoder in order to transform the various input tokens. However, there are also other applications in which the decoder part of the traditional Transformer Architecture is also used.

In Masked Autoencoder, there are two ViTs put end-to-end. The first one takes in image patches with positional encoding, and outputs vectors representing each patch. The second one takes in vectors with positional encoding and outputs image patches again. During training, both ViTs are used. An image is cut into patches, and only 25% of the patches are put into the first ViT. The second ViT takes the encoded vectors and outputs a reconstruction of the full image. During use, only the first ViT is used.

M. OpenCV

[28] OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS. OpenCV leans mostly towards real-time vision applications and takes advantage of MMX and SSE instructions when available. A full-featured CUDAand OpenCL interfaces are being actively developed right now. There are over 500 algorithms and about 10 times as many functions that compose or support those algorithms. OpenCV is written natively in C++ and has a templated interface that works seamlessly with STL containers. Since our goal is to diagnose strokes through facial expressions, we have to utilize computer vision technology. We implemented functions like image-to-vector conversion using OpenCV, one of the most renowned libraries in computer vision.

N. dlib

[29] Dlib is a general purpose cross-platform software library written in the programming language C++. Its design is heavily influenced by ideas from design by contract and component-based software engineering. Thus it is, first and foremost, a set of independent software components. It is open-source software released under a Boost Software License.

Since development began in 2002, Dlib has grown to include a wide variety of tools. As of 2016, it contains software components for dealing with networking, threads, graphical user interfaces, data structures, linear algebra, machine learning, image processing, data mining, XML and text parsing, numerical optimization, Bayesian networks, and many other tasks. In recent years, much of the development has been focused on creating a broad set of statistical machine learning tools and in 2009 Dlib was published in the Journal of Machine Learning Research. Since then it has been used in a wide range of domains.

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments. Dlib's open source licensing allows you to use it in any application, free of charge.

Unlike a lot of open source projects, this one provides complete and precise documentation for every class and function. There are also debugging modes that check the documented preconditions for functions. When this is enabled it will catch the vast majority of bugs caused by calling functions incorrectly or using objects in an incorrect manner. We used dlib for Face recognition and Face landmark detection. Dlib provides functions for these purposes, which helped us progress the project more efficiently.

O. face_utils

[30] this is an opensource wrapper library for the most common face detection models. It also provides multiple face utilities such as face cropping. Supported detection models. first, face_recognition (hog and cnn), second, retina face model third, haar cascade face detection. We used the face_utils library from imutils to resize images.

P. pip

[31] pip (also known by Python 3's alias pip3) is a package-management system written in Python and is used to install and manage software packages. The Python Software Foundation recommends using pip for installing Python applications and its dependencies during deployment.
Pip connects to an online repository of public packages, called the Python Package Index. Pip can be configured to connect to other package repositories (local or remote), provided that they comply to Python Enhancement Proposal 503. The fields of AI and computer vision have become highly active, particularly in the context of Python. One can easily and quickly install and use libraries for these purposes through pip.

Q. ssh

[32] The Secure Shell Protocol (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. Its most notable applications are remote login and command-line execution.

SSH applications are based on a client–server architecture, connecting an SSH client instance with an SSH server. SSH operates as a layered protocol suite comprising three principal hierarchical components: the transport layer provides server authentication, confidentiality, and integrity; the user authentication protocol validates the user to the server; and the connection protocol multiplexes the encrypted tunnel into multiple logical communication channels.

SSH was designed on Unix-like operating systems, as a replacement for Telnet and for unsecured remote Unix shell protocols, such as the Berkeley Remote Shell (rsh) and the related rlogin and rexec protocols, which all use insecure, plaintext transmission of authentication tokens.

Subsequent development of the protocol suite proceeded in several developer groups, producing several variants of implementation. The protocol specification distinguishes two major versions, referred to as SSH-1 and SSH-2. The most commonly implemented software stack is OpenSSH, released in 1999 as open-source software by the OpenBSD developers. Implementations are distributed for all types of operating systems in common use, including embedded systems.

We utilized SSH for efficient coding on Raspberry Pi and to establish a connection between the local environment and AWS.

R. VSCode

[33] Visual Studio Code, also commonly referred to as VS Code, is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add functionality.

In the Stack Overflow 2023 Developer Survey, Visual Studio Code was ranked the most popular developer environment tool among 86,544 respondents, with 73.71% reporting that they use it. It increased its use among those learning to code versus those developing as a profession.

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including C, C#, C++, Fortran, Go, Java, JavaScript, Node.js, Python, Rust, and Julia. It is based on the Electron framework, which is used to develop Node.js web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed ”Monaco”) used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

Out of the box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

VSCode is currently the most popular code editor. It is not only clean but also supports various languages and extension packages. We used this editor to enhance efficiency when programming.

S. aws-iot

[34] AWS IoT provides cloud services for connecting IoT devices to other devices and AWS cloud services. It offers device software that aids in integrating IoT devices into AWS IoT-based solutions. When devices are connected to AWS IoT, they can be linked to cloud services provided by AWS. AWS IoT allows you to choose the latest technologies that best suit your solution.

In the field of managing and supporting IoT devices, AWS IoT Core supports the following protocols: MQTT, MQTT over WSS (WebSockets), HTTPS, and LoRaWAN. AWS IoT Core's message broker supports devices and clients that use MQTT and MQTT over WSS protocols for publishing and subscribing to messages. It also supports devices and clients that use the HTTPS protocol for message publication.

AWS IoT Core for LoRaWAN enables the connection and management of wireless LoRaWAN (Low-Power Wide-Area Network) devices. It eliminates the need to develop and operate a LoRaWAN Network Server (LNS). To effectively utilize AWS on Raspberry Pi, you have installed AWS's dedicated IoT SDK.

T. Wi-fi

[35] Wi-Fi is a family of wireless network protocols based on the IEEE 802.11 family of standards, which are commonly used for local area networking of devices and Internet access, allowing nearby digital devices to exchange data by radio waves.

These are the most widely used computer networks, used globally in home and small office networks to link devices and to provide Internet access with wireless routers and wireless access points in public places such as coffee shops, hotels, libraries, and airports to provide visitors.

Wi-Fi technology may be used to provide local network and Internet access to devices that are within Wi-Fi range of one or more routers that are connected to the Internet. The coverage of one or more interconnected access points can extend from an area as small as a few rooms to as large as many square kilometres. Coverage in the larger area may require a group of access points with overlapping coverage. For example, public outdoor Wi-Fi technology has been used successfully in wireless mesh networks in London. An international example is Fon.

We use Raspberry Pi to implement true IoT technology. To connect Raspberry Pi to the internet, we have the option of either directly plugging in an Ethernet LAN cable or using Wi-Fi. Given that we designed the project with the consideration of it being integrated into home appliances, we chose to use Wi-Fi for wireless internet connectivity instead of a LAN cable.

U. Github

[36] GitHub is a platform and cloud-based service for software development and version control using Git, allowing developers to store and manage their code.

It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project. Headquartered in California, it has been a subsidiary of Microsoft since 2018.

We use GitHub for efficient collaboration. We manage repositories and files remotely, create branches to manage different versions, and allow team members to easily see what new changes or additions they've made.

V. NUGU

The NUGU AI speaker integration with the Raspberry Pi requires a NUGU developer account and access to the NUGU developer documentation. The integration begins with connecting to the NUGU API, an essential interface for communication between the Raspberry Pi and the NUGU AI speaker.

As part of the integration prerequisites, developers must obtain the necessary API keys or credentials from the NUGU developer portal. These authentication mechanisms are critical for securing communication between the Raspberry Pi and the NUGU speaker, ensuring that only authorized requests are processed. The next phase involves implementing the interaction logic. A script running on the Raspberry Pi will enable triggering the NUGU speaker with predefined messages upon stroke detection by the machine learning model.

Once the integration is complete, the NUGU AI speaker can be used to alert the user or call for help in the event of a stroke. The speaker can also be used to provide the user with information about stroke prevention and treatment.

W. Task distribution

*1) **Park Geonryul**:* Park implemented an AI model and trained it using Google Colab and Pytorch. He was in charge of building a virtual machine using AWS Lightsail and deploying the AI model to a web server using FastAPI.

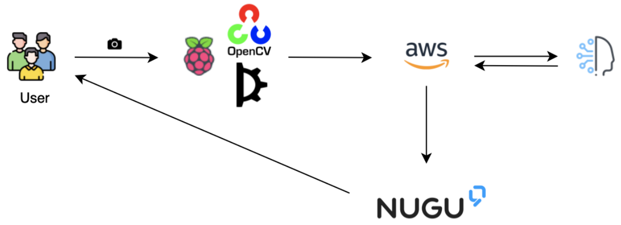
*2) **Lee Seungsu**:* Lee have planned this project and implemented focusing on overall IoT technology using Raspberry Pi. He developed algorithms for capturing faces using OpenCV and Dlib. Additionally, he created a two-level detection model for privacy. The process involves capturing images on the Raspberry Pi, detecting face changes in real-time, blurring the background, sending them to the server, and receiving the results to inform the customer.

*3) **Elia Ayoub**:* Elia did researches on the AI NUGU Speaker. He tried to find how to connect the speaker to the Rasberry Pi as well as how to deliver the condition found through the AI model to the users.

*4) **Ryan Jabbour**:* Ryan worked jointly with Elia on the AI NUGU Speaker and worked on keeping a logical progress of the project for the group while keeping everything in order such as documentation and team work and meetings.

V. SPECIFICATIONS

Fig. 10. Project Architecture



A. Training AI model

This is the process of learning an artificial intelligence model based on data. The data is a 38MB image from kaggle's 'Face Images of Acute Stroke and Non Acute Stroke'. Training is conducted with a total of 4,000 pieces of data, including about 1,600 stroke face images and 2,400 non-stroke face images. Image preprocessing during the learning process proceeds as follows. Convert to a tensor of (200x200x3) and flatten. The label indicates the stroke face as 1 and the non-stroke face as 0, and the SVM and Transformer models are trained in Google Colab under the above conditions.

*1) **Support Vector Machine**:*

- C(Regularization Parameter): C is a parameter that controls the penalty for misclassifications. A small C value sets a low penalty for misclassifications, making the model fit the training data more. A large C value sets a high penalty for misclassifications, encouraging the model to achieve higher accuracy on the training data. The appropriate value for C is determined through cross-validation.
- Kernel: SVM supports various kernel functions. The most commonly used kernels include the linear kernel, polynomial kernel, and Radial Basis Function (RBF) kernel. The choice of kernel depends on the data's characteristics and the problem at hand.
- Gamma: When using the RBF kernel, Gamma controls the flexibility of the decision boundary. A small Gamma value makes the decision boundary smoother, while a large Gamma value makes the decision boundary more complex. The appropriate value for Gamma is also determined through cross-validation.

*2) **ViT**:* [37] ViT is composed of various modules. The module is composed as follows.

Patchify

The transformer encoder was developed with sequence data in mind, such as English sentences. However, an image is not a sequence. So we have to "sequencify" an image. We break it into multiple sub-images and map each sub-image to a vector.

We do so by simply reshaping our input, which has size (N, C, H, W), to size (N, #Patches, Patch dimensionality), where the dimensionality of a patch is adjusted accordingly.

Adding classification tokens

If you look closely at the architecture picture, you will notice that also a "v_class" token is passed to the Transformer Encoder.

Simply put, this is a special token that we add to our model that has the role of capturing information about the other tokens. This will happen with the MSA block (later on). When information about all other tokens will be present here, we will be able to classify the image using only this special token. The initial value of the special token (the one fed to the transformer encoder) is a parameter of the model that needs to be learned.

If we wanted to do another downstream task, we would just need to add another special token for the other downstream task (for example, classifying a digit as higher than 5 or lower) and a classifier that takes as input this new token.

Positional Encoding

As anticipated, positional encoding allows the model to understand where each patch would be placed in the original image. While it is theoretically possible to learn such positional embeddings, previous work by Vaswani et. al. suggests that we can just add sines and cosines waves.

In particular, positional encoding adds high-frequency values to the first dimensions and low-frequency values to the latter dimensions.

In each sequence, for token i we add to its j-th coordinate the following value:

$$p_{i,j} = \begin{cases} sin\left(\frac{i}{10000^{\frac{j}{d_{emb\_dim}}}}\right) \\ cos\left(\frac{i}{10000^{\frac{j}{d_{emb\_dim}}}}\right) \end{cases}$$

This positional embedding is a function of the number of elements in the sequence and the dimensionality of each element. Thus, it is always a 2-dimensional tensor or “rectangle”.

Here’s a simple function that, given the number of tokens and the dimensionality of each of them, outputs a matrix where each coordinate (i,j) is the value to be added to token i in dimension j.

Layer Normalization Layer normalization is a popular block that, given an input, subtracts its mean and divides by the standard deviation.

However, we commonly apply layer normalization to an (N, d) input, where d is the dimensionality. Luckily, also the Layer Normalization module generalizes to multiple dimensions.

Layer normalization is applied to the last dimension only. We can thus make each of our 50x8 matrices (representing a single sequence) have mean 0 and std 1.

Multi-head Self Attention

Simply put: we want, for a single image, each patch to get updated based on some similarity measure with the other patches. We do so by linearly mapping each patch to 3 distinct vectors: $\mathbf{q}, \mathbf{k}$, and $\mathbf{v}$ (query, key, value).

Then, for a single patch, we are going to compute the dot product between its q vector with all of the k vectors, divide by the square root of the dimensionality of these vectors, softmax these so-called attention cues, and finally multiply each attention cue with the v vectors associated with the different k vectors and sum all up.

In this way, each patch assumes a new value that is based on its similarity (after the linear mapping to $\mathbf{q}, \mathbf{k}$ and $\mathbf{v}$) with other patches. This whole procedure, however, is carried out H times on H sub-vectors of our current 8-dimensional patches, where H is the number of Heads. If you’re unfamiliar with the attention and multi-head attention mechanisms, I suggest you read this nice post by Yasuto Tamura.

Once all results are obtained, they are concatenated together. Finally, the result is passed through a linear layer (for good measure).

The intuitive idea behind attention is that it allows modeling the relationship between the inputs. What makes a ‘0’ a zero are not the individual pixel values, but how they relate to each other.

Residual Connection

A residual connection consists in just adding the original input to the result of some computation. This, intuitively, allows a network to become more powerful while also preserving the set of possible functions that the model can approximate.

With this self-attention mechanism, the class token (first token of each of the N sequences) now has information regarding all other tokens.

Classfication MLP

Finally, we can extract just the classification token (first token) out of our N sequences, and use each token to get N classifications.

B. *Saving trained AI model*

The trained artificial intelligence model is saved as a file so that it can be distributed to a web server. There are various saving formats. The method we used is joblib. joblib was chosen because it allows you to simply save and load models with the dump() and load() commands. Save the artificial intelligence model that has completed training in Google Colab to Google Drive using the joblib.dump() command and download it to your local computer.

C. *Loading trained AI model*

Just like how you saved it, you can load the trained artificial intelligence model from a file using the joblib library. Train artificial intelligence in Google Colab and move the saved model locally to AWS Lightsail using the scp command. This completes the preparation to deploy the model as an AWS Lightsail virtual machine using a web server.

D. *Classifying Image with trained AI model*

Images are preprocessed similarly to when training an artificial intelligence model. Convert the image to (200x200x3) tensor. Process it by calling the resize function of the ’skimage’ library. Afterwards, image preprocessing is completed by flattening. Call the predict method with the loaded artificial intelligence model and return the predicted value by passing the preprocessed image as a parameter. At this time, not only the classified result of 0 or 1 is returned, but also the probability value is returned to inform the user of the basis for the judgment.

E. *Returning the result*

Because the returned value needs to be sent back to the Raspberry Pi via web communication, it is changed to a JSON object. It consists of a total of three key-value values. The key is as follows. ’prediction’, ’probability_0’, and ’probability_1’ represent the predicted value, non-stroke probability, and stroke probability, respectively. Example objects are as follows:

```
{
    'prediction': result_list[0],
    'probability_0': probability[0][0],
    'probability_1': probability[0][1]
}
```

F. *Get Image with API*

Using FastAPI’s File and UploadFile libraries, images sent through a web server can be processed within the program. When used in combination with the ’imread’ method of the skimage.io library, images can be transmitted to the artificial intelligence model using the same logic as in the training process. Example code is as follows:

```
{
    img_array = imread(file.file)
}
```

G. *Post the result with API*

In ’Returning the result’ above, the predicted value is converted to JSON format and the resulting value is sent back to the Raspberry Pi. When you return from a function with the @app.post(”/classify/”) annotation, a response is sent to the place where the image file was sent, so if you attach the annotation to the function that performs classification and return the above Json format, you can Post.

H. *Encapsulate the AI model*

Everything mentioned above can be implemented in one file. You can create an app.py file, create image preprocessing, prediction from an artificial intelligence model, and return value, and attach an annotation on top of this function to encapsulate it so that all tasks are processed at once.

I. *Run the Web Server*

You can run the app.py file with Uvicorn. At this time, enter and execute the additional command as shown below to enable connection to the external IP.

```
    uvicorn app:app --reload --host=0.0.0.0
```

J. *Raspberry Pi*

*1) **Raspberry OS installation and connection**:* To download the Raspberry Pi OS image, you will need a micro SD card. Recently, with the introduction of the Raspberry Pi Imager, downloading and installing the OS has become more convenient. First, download the Raspberry Pi Imager on your local desktop. Then, insert the micro SD card into your desktop. In the Raspberry Pi Imager, select the SD card from the Storage tab and install the Raspberry Pi OS. After that, insert the SD card into the Raspberry Pi, and connect the power using a USB-C port. Your Raspberry Pi will power on.

*2) **Wi-fi**:* While Raspberry Pi does have a built-in Wi-Fi module, it doesn’t automatically connect during boot. To enable this functionality, you need to modify the ’wpa_supplicant.conf’ file. Insert your Wi-Fi ID and password as shown below, and use ’scan_ssid = 1’ to allow detection of hidden networks. By making these changes to the ’wpa_supplicant.conf’ file, Raspberry Pi will automatically connect to Wi-Fi during boot. Wi-Fi connection is crucial not only for internet access but also for SSH usage.

*3) **SSH**:* You can configure SSH in the Raspberry Pi settings. Enabling SSH automatically completes the necessary settings. After that, when you first establish an SSH connection, set the connection password, and you are ready to use SSH.

SSH primarily relies on IP for forwarding, so you need to know the current IP address of the Raspberry Pi, which you can check by entering ’ifconfig’ in the terminal. Additionally, the local and remote devices you want to connect must be connected to the same Wi-Fi network by default. After connecting your local desktop to the same Wi-Fi network as the Raspberry Pi, you can complete the SSH connection by entering ’ssh userID@xxx.xx.xx.x,’ providing your username and IP address, and entering the password.

*4) **Camera**:* With the update of the Raspberry Pi OS to ’Bookworm,’ there have been changes in the overall software for using the camera. The update involved transitioning to the ’libcamera’ library, which posed challenges when applying it to essential camera functions, such as ’VideoCapture’ in OpenCV. Therefore, it is necessary to revert to the previous version.

In the ’config.txt’ file, you should provide the ’start_x=1’ parameter and turn off the ’camera_auto_detect’ feature.

*5) **Virtual Environment for Python**:* Raspberry Pi is an externally managed environment, and as such, it restricts the use of ’pip,’ allowing only ’apt’ for package management. However, creating a Python virtual environment can provide more flexibility within these constraints. A significant reason for using Python is the availability of a wide range of libraries. To conveniently install these libraries, it is advisable to create a virtual environment.

*6) **Swap size**:* OpenCV is a sizable library, and to ensure stable operation, it’s important to allocate sufficient disk space. You can determine the current swap size in use by using ’free -m.’ Then, you can adjust the swap size by modifying ’/etc/dphys-swapfile.’ The default setting is 100, but to ensure normal operation, it’s recommended to set it to 4096 or higher. After adjusting the swap size, you can proceed with the installation of OpenCV

K. *OpenCV*

*1) **video_capture**:* to use the ’VideoCapture’ function properly, you need to downgrade the camera library and disable ’camera_auto_detect’ to use OpenCV/V4L2. Additionally, increasing the GPU memory on the Raspberry Pi is recommended for improved video processing.

*2) **Color scale converting**:* Using ’cv2.cvtColor,’ the image being captured is converted to grayscale. This is used for the following reasons:

- Information Compression: Grayscale images have only one color channel per pixel, representing only brightness information. This reduces image size, saving memory and reducing computational requirements for storage and processing.

- Computational Efficiency: Grayscale image processing is much faster and more efficient compared to color image processing. Handling a single color channel is faster and more efficient than multiple color channels.

- Feature Extraction and Object Detection: Grayscale images are primarily used in image processing tasks such as edge detection, feature extraction, and object detection. Edge detection algorithms often rely on brightness information, making grayscale images more suitable for these tasks.

- Memory Savings: Grayscale images require less memory compared to color images, which can be advantageous for memory-intensive tasks, such as deep learning models.

Facial recognition is possible with grayscale images, so changing to grayscale helps increase computational efficiency.

L. NUGU

*1) **NUGU Developer Account and Documentation Access**:* Before integrating the NUGU AI speaker with the Raspberry Pi, developers must create a NUGU developer account and access the NUGU developer documentation. The NUGU API/SDK provides a secure and efficient way to connect the Raspberry Pi to the NUGU AI speaker and exchange information between the two devices.

*2) **Connection to NUGU API/SDK**:* Once the Raspberry Pi is connected to the NUGU API/SDK, developers must obtain the necessary API keys or credentials from the NUGU developer portal. These credentials authenticate requests and ensure that only authorized devices can communicate with the NUGU AI speaker.

*3) **Logic for Interaction - Triggering NUGU Speaker**:* After obtaining the necessary API keys or credentials, developers can develop the interaction logic for the integration. This involves writing a script on the Raspberry Pi to trigger the NUGU speaker with predefined messages when the machine learning model detects a stroke. The script should integrate with the NUGU API/SDK to communicate effectively with the NUGU AI speaker.

M. Privacy Policy

To ensure personal data protection, we implement a two-step process:

1) The first step involves blurring the background screen.
   a) Backgroung Blur

2) The second step uses a computer vision model that operates exclusively on the local device for initial assessment. Only if detection is made, the connection to the server is established. we called it 'Two-level detection'. And if connected to the internet, an LED module lights up next to the camera, allowing customers to visually confirm whether the camera is online or not.

   Two-level detection: [38] Through the following research paper, we obtained the gradient difference at the corner of the mouth for stroke diagnosis using face detection. Utilizing the coordinates of each landmark obtained above, we issue a warning when the gradient exceeds a certain threshold.

   a) **When No Detection Occurs in Fisrt-level:**

      In the absence of detection in real-time, the system continues to operate actively within the user's daily life without any specific signals.

   b) **When Detection Occurs in First-level:**

      Upon detecting risk factors in real-time, the system establishes a connection to the server and captures a more accurate photo for further detection using a trained artificial intelligence model (e.g., SVM, Transformer), ensuring higher accuracy.

   c) **When the Diagnosis indicates a Stroke in Second-level:**

      In the event of a stroke diagnosis, the user is promptly informed. After the notification, the system seeks the user's consent, and upon receiving it, automatically initiates a 119 emergency call.

   d) **When the Diagnosis Indicates No Stroke:**

      If no detection occurs in the initial assessment, the system continues its operation. However, in the secondary assessment, if no detection takes place, the user is informed to provide reassurance.

References

[1] National Statistical Office. Cause of death statistics results. 2022.

[2] American Stroke Association. About stroke.

[3] Alejandro M Spiotta, Jan Vargas, Raymond Turner, M Imran Chaudry, Holly Battenhouse, and Aquilla S Turk. The golden hour of stroke intervention: effect of thrombectomy procedural time in acute ischemic stroke on outcome. *Journal of neurointerventional surgery*, 6(7):511–516, 2014.

[4] 위키백과. 뇌졸중.

[5] Faten El Ammar, Agnieszka Ardelt, Victor J Del Brutto, Andrea Loggini, Zachary Bulwa, Raisa C Martinez, Cedric J McKoy, James Brorson, Ali Mansour, and Fernando D Goldenberg. Be-fast: a sensitive screening tool to identify in-hospital acute ischemic stroke. *Journal of stroke and cerebrovascular diseases*, 29(7):104821, 2020.

[6] 중앙일보. 60% 더 비싼 '문안의 문' 냉장고 불티 왜. 2012.

[7] ELIZABETHTOWN GAS. Cold facts about your refrigerator.

[8] 국립중앙의료원. 응급의료현황통계, 뇌졸중 환자의 발병 후 응급실 도착 소요시간 현황. *Kosis*, 2021.

[9] 국립중앙의료원. 공공의료기관현황, 시도별 공공의료기관 전문의 현황. *Kosis*, 2021.

[10] 통계청. 인구총조사, 가구주의 성, 연령 및 세대구성별 가구. *Kosis*, 2022.

[11] 질병관리청. 지역사회건강조사, 시도별 뇌졸중 조기증상 인지율. *Kosis*, 2022.

[12] 국립중앙의료원. 응급의료현황통계 - 뇌졸중 환자의 응급진료 결과 현황. *Kosis*, 2021.

[13] Han Gao, Amir Ali Mokhtarzadeh, Shaofan Li, Hongyan Fei, Junzuo Geng, and Deye Wang. Multi-angle face expression recognition based on integration of lightweight deep network and key point feature positioning. In *Journal of Physics: Conference Series*, volume 2467, page 012008. IOP Publishing, 2023.

[14] MARGA ROMA. The rise of home diagnostics and proactive self-care. 2022.

[15] Han Gao, Amir Ali Mokhtarzadeh, Shaofan Li, Hongyan Fei, Junzuo Geng, and Deye Wang. Multi-angle face expression recognition based on integration of lightweight deep network and key point feature positioning. *Journal of Physics: Conference Series*, 2467, 2023.

[16] JOYDIP DUTTA. https://circuitdigest.com/microcontroller-projects/raspberry-pi-based-emotion-recognition-using-opencv-tensorflow-and-keras.

[17] AWS. https://docs.aws.amazon.com/iot/latest/developerguide/connecting-to-existing-device.html.

[18] victor369basu. https://github.com/victor369basu/facial-emotion-recognition.

[19] MANAS SAMBARE. https://www.kaggle.com/datasets/msambare/fer2013.

[20] Adrian Rosebrock. https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/.

[21] Wikipedia contributors. Scikit-learn — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Scikit-learn&oldid=1165753997, 2023. [Online; accessed 1-November-2023].

[22] Wikipedia contributors. Python imaging library — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Python_Imaging_Library&oldid=1180608377, 2023. [Online; accessed 1-November-2023].

[23] Wikipedia contributors. Numpy — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=NumPy&oldid=1182528270, 2023. [Online; accessed 1-November-2023].

[24] Wikipedia contributors. Support vector machine — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Support_vector_machine&oldid=1181915689, 2023. [Online; accessed 2-November-2023].

[25] Wikipedia contributors. Transformer (machine learning model) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Transformer_(machine_learning_model)&oldid=1182841289, 2023. [Online; accessed 2-November-2023].

[26] Wikipedia contributors. Attention (machine learning) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Attention_(machine_learning)&oldid=1181687523, 2023. [Online; accessed 2-November-2023].

[27] Wikipedia contributors. Vision transformer — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Vision_transformer&oldid=1182845224, 2023. [Online; accessed 2-November-2023].

[28] OpenCV. https://opencv.org/about/.

[29] Wikipedia contributors. Dlib — Wikipedia, the free encyclopedia, 2023. [Online; accessed 1-November-2023].

[30] PyPi. https://pypi.org/project/face-utils/.

[31] Wikipedia contributors. Pip (package manager) — Wikipedia, the free encyclopedia, 2023. [Online; accessed 1-November-2023].

[32] Wikipedia contributors. Secure shell — Wikipedia, the free encyclopedia, 2023. [Online; accessed 1-November-2023].

[33] Wikipedia contributors. Visual studio code — Wikipedia, the free encyclopedia, 2023. [Online; accessed 1-November-2023].

[34] AWS. https://docs.aws.amazon.com/ko_kr/iot/.

[35] Wikipedia contributors. Wi-fi — Wikipedia, the free encyclopedia, 2023. [Online; accessed 1-November-2023].

[36] Wikipedia contributors. Github — Wikipedia, the free encyclopedia, 2023. [Online; accessed 1-November-2023].

[37] Brian Pulfer. Vision transformers from scratch: A step-by-step guide, 2022.

[38] Oi-Mean Foong, Kah-Wing Hong, and Suet-Peng Yong. Droopy mouth detection model in stroke warning. *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, 2016.