Interest is not the most important thing.
The most important thing is hard-working.
*Hard-working is the key of **success.***
Keep in your mind that you can achieve any with the patience

Projects and Researches

Presenter:
Abduraimjonov Abdurakhmon

Agenda

I. **Researches and Projects in Onycom**
   1. Loading Time Project
   2. Creating Automation Testing Tool for mobile applications
   3. Detecting clickable objects in mobile applications
   4. Creating Auto-labeling tool for labeling all objects (buttons, text buttons, image and etc..) on Image
   5. Multi-devices controlling in the same time

II. **Researches and Projects in Roborus**
   1. AI Projects and Researches
   2. GUI application Development

III. **Researches and Projects in GREW Creative Lab**
   1. CSONG Development

IV. **Researches and Projects in CVPR Lab**
   1. Computer Vision Researches
   2. AI Researches and Paper summary

Loading Time Project

I. Creating Loading time **detector** and **classifier**.

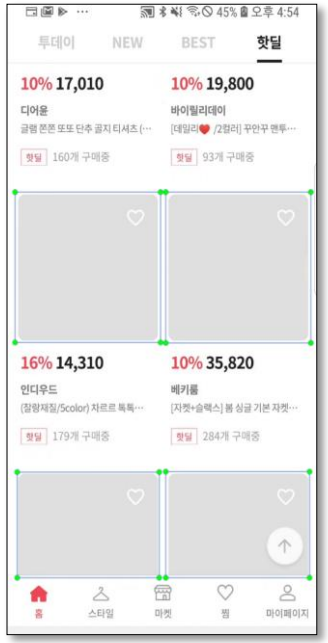
II. Creating Loading time API.

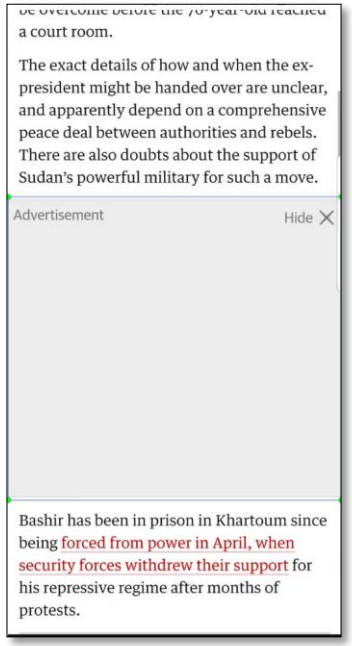
Step 1 > Collecting dataset: Take screenshots using **150** mobile apps

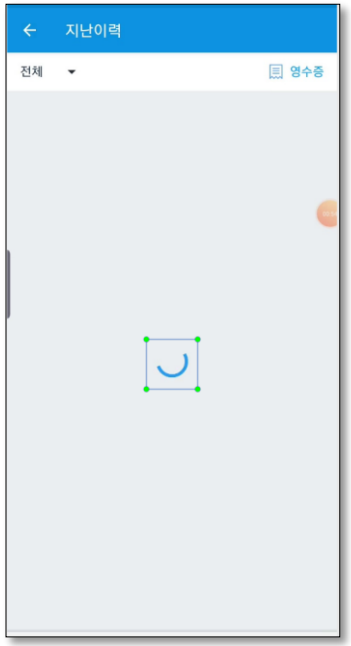
Step 2 > Labeling dataset using **LabelImg** tool (link: https://github.com/tzutalin/labelImg)

class names = [loading]

Loading image examples:



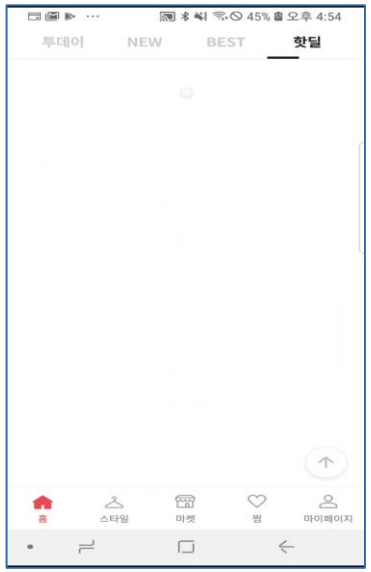


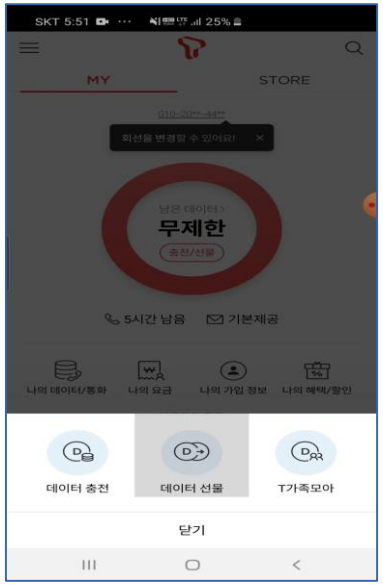


Loading Time Project

Creating Loading time detector and classifier.
Creating Loading time API.

Step 1 > Collecting dataset: Take screenshots using **150** mobile apps
Step 2 > Labeling dataset using **LabelImg** tool (link: https://github.com/tzutalin/labelImg)
Step 3 > Sorting dataset for classification:
class names = [loading, normal]





Loading Time Project : Loading Time API

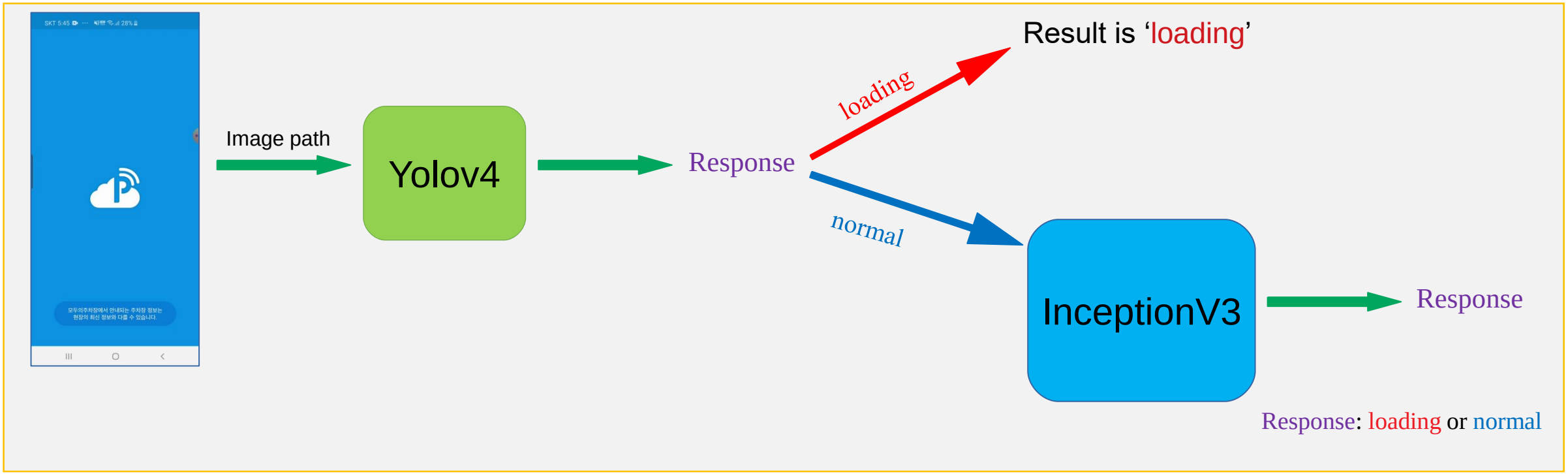
Creating Loading time detector and classifier.
Creating Loading time API.

Step 1 > Collecting dataset: Take screenshots using **150** mobile apps
Step 2 > Labeling dataset using **LabelImg** tool (link: https://github.com/tzutalin/labelImg)
Step 3 > Sorting dataset for classification
Step 4 > Making detection and classification model:
Detection model: Training dataset with Yolov3 and Yolov4 using Darknet (link: https://github.com/AlexeyAB/darknet)

Classification model: Training dataset with Inception v3 pre-trained model (link: https://tfhub.dev/google/imagenet/inception_v3/feature_vector/3)
Step 5 > Creating loading time **API** using Flask

Loading Time Project : Loading Time API

Working process of Loading Time API:



In the above architecture: Image will be sent to Yolov4, if the response from Yolov4 is 'loading', *result is loading*.
If the response from Yolov4 is 'normal', the image will be sent to **InceptionV3**. **InceptionV3** gives the final result.

Please enter to download video for knowing how **Loading Time API** works:
Video link : https://github.com/aavuzb/project_videos/raw/main/Loading_Time_API.mp4

Creating Automation Testing Tool for mobile applications

I tried to create **Automation Testing Tool** which is used for *testing mobile application automatically*.

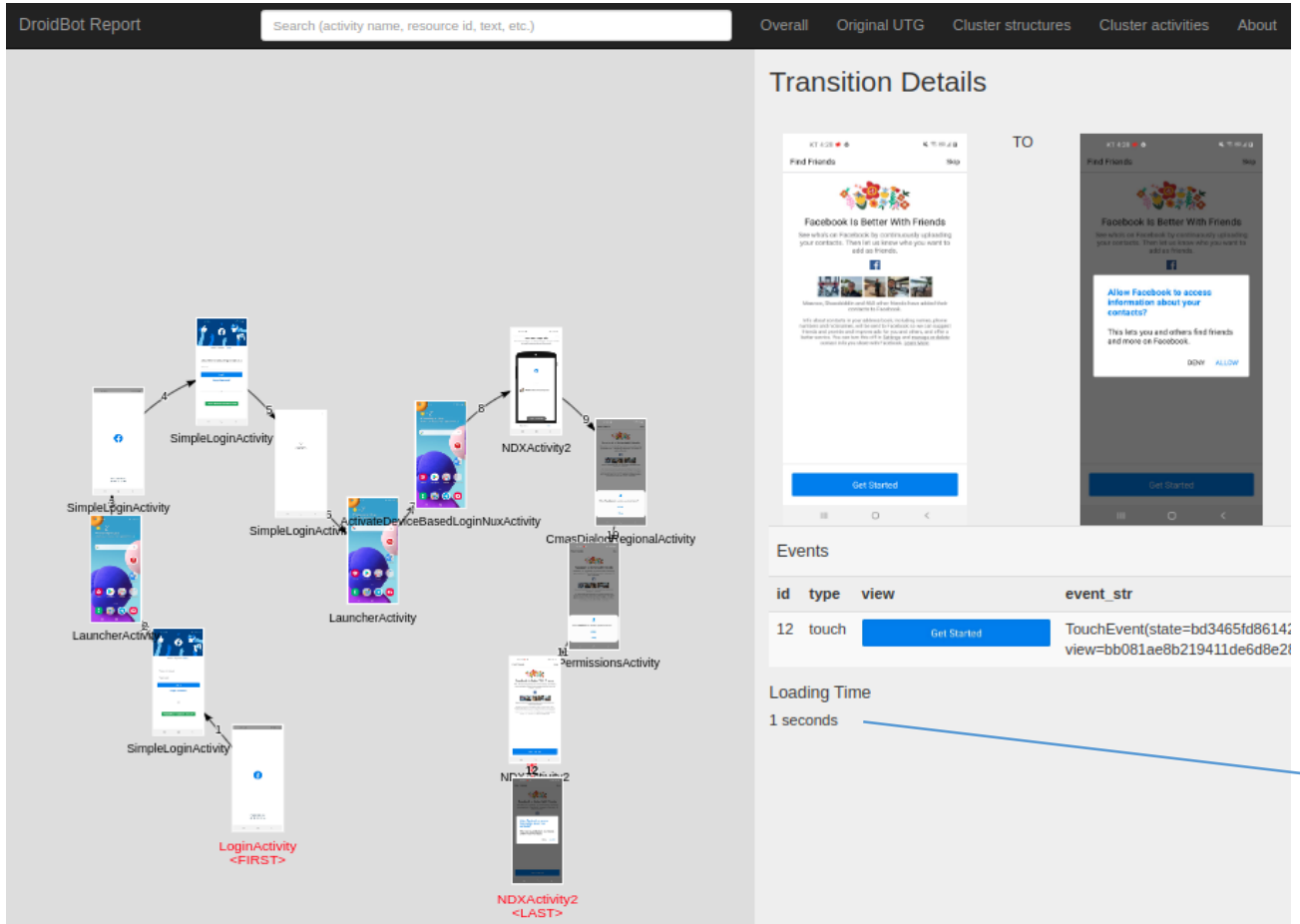
To create Automation Testing Tool, I combined **DroidBot** (link: https://github.com/honeynet/droidbot) and **Loading Time API**

Responsibilities of **Automation Testing Tool**:

- Installing mobile app
- Touching all clickable objects (image, icon, …)
- Finding total loading time (Connecting with Loading Time API)
- Showing all result after testing mobile application

Creating Automation Testing Tool for mobile applications

Output result of Automation Tool



Output result of "facebook.apk" after testing by **DroidBot** and **Loading Time API**

State 11 to State 12

Result = 1 seconds

Result means that Loading time API found only *one loading image* from **State 11** to **State 12**

Please enter to download video for knowing how Automation Tool works (Instagram is used for testing):
Video link : https://github.com/aavuzb/project_videos/raw/main/DroidBot_Loading_Time_API.avi

Detecting clickable objects in mobile applications

Step 1 => **Collecting dataset**: Take screenshots using **150** mobile apps

Step 2 => **Labeling dataset** using **LabelImg** tool (link: https://github.com/tzutalin/labelImg)

class names = [icon_button, image, input_box, clickable_item, button, text_button]

dataset examples:

Detecting clickable objects in mobile applications

Step 1 => **Collecting dataset**: Take screenshots using **150** mobile apps
Step 2 => **Labeling dataset** using **LabelImg** tool (link: https://github.com/tzutalin/labelImg)
Step 3 => Making detection model:
Detection model: Training dataset with Yolov3 and Yolov4 using Darknet (link: https://github.com/AlexeyAB/darknet)
Step 4 => Creating loading time **API** using Flask

Please enter to download video for knowing how **Detection Model** works :
Video link : https://github.com/aavuzb/project_videos/raw/main/Detection_model_6_classes.mp4

Creating Auto-labeling tool for labeling all objects (buttons, text buttons, image and etc..) on Image

The ***goal*** of this task is to *label all objects* (with **their names**) on images.
To creating **Auto-labeling**:

Auto-labeling Tool **1** (**ALT1**) = UI Automator + OCR + Icon classification model
Auto-labeling Tool **2** (**ALT2**) = Detection Model + OCR + Icon classification model

UI Automator**:** https://github.com/openatx/uiautomator2

Detection Model: which is explained previous pages

OCR: **Easy OCR** open-source project (link: https://github.com/JaidedAI/EasyOCR)

Icon classification model: dataset is download from **Kaggle** and dataset was training by **VGG** model

Creating Auto-labeling tool for labeling all objects (buttons, text buttons, image and etc..) on Image

input

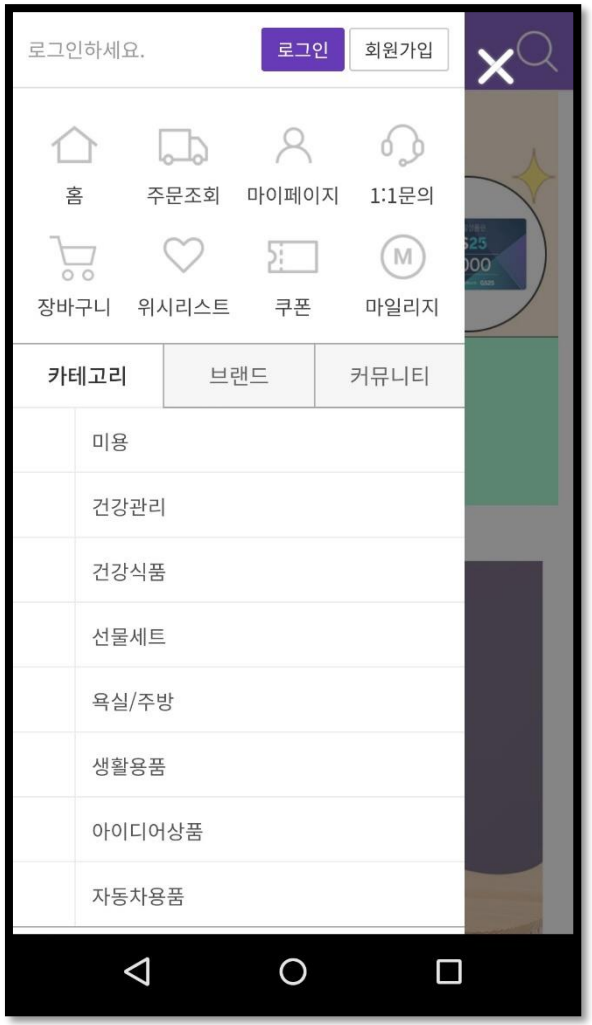
ALT1 output

ALT2 output

Abduraimjonov Abdurakhmon

Creating Auto-labeling tool for labeling all objects (buttons, text buttons, image and etc..) on Image

input



ALT1 output

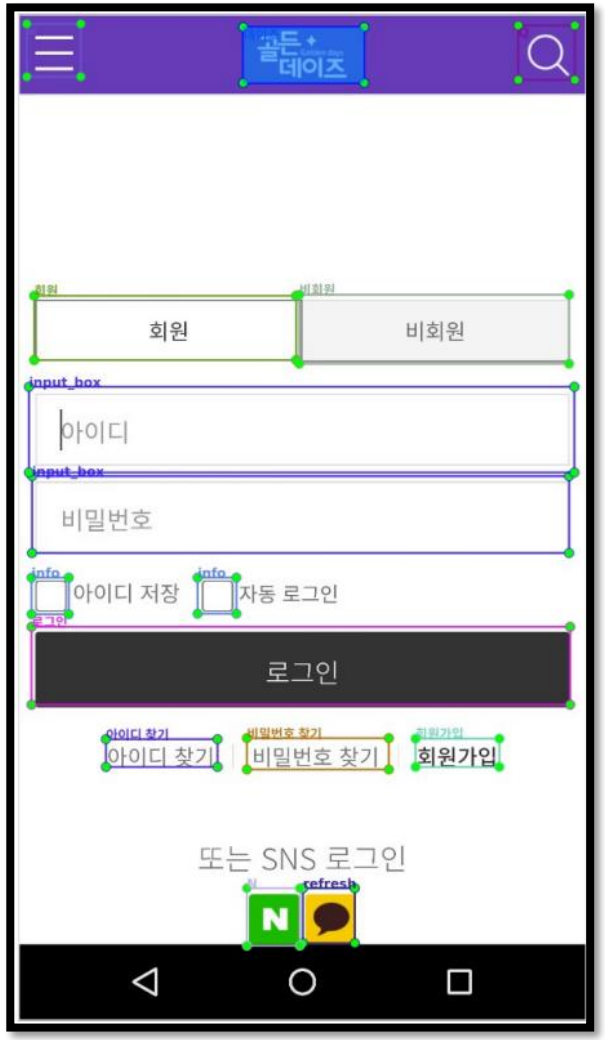
ALT2 output

Creating Auto-labeling tool for labeling all objects (buttons, text buttons, image and etc..) on Image

input

ALT1 output

ALT2 output



Multi-devices controlling in the same time

The goal of this task is to **control multiple devices in the same time**.

To implement this task, I used *"Multi-scale template matching"* open-source code (link: https://github.com/agiledots/multiscale-template-matching),

But I have created **new formula** which help to decrease the matching time.

I have implemented this task in **Python** and **C++** .

Please watch below videos for knowing what is **multi-device controlling** and how it works:

Phone with other devices: link => https://github.com/aavuzb/project_videos/raw/main/multi-devices-controlling_Phone.mp4

Tablet with other devices: link => https://github.com/aavuzb/project_videos/raw/main/multi-devices-controlling_Tablet.mp4

Café Cleanliness

Using Python with Tensorflow and Keras

AI Researches

Related to Face Recognition by using FaceNet
Image Enhancement by GANs and Computer Vision

AI Project

Café cleanliness using Transfer Learning

Inception, Mobile Net and Efficient Nets [B0, B7]
Best result on Efficient Net B3 and Efficient Net B6

Unoccupied + Empty

Unoccupied + not_Empty

Occupied + not_Empty

Occupied + Empty

Café Cleanliness : Result

The goal of this Project is :

Improving the quality of service.

Please enter to download video for knowing how **Café Cleanliness project** works :

Video link : https://github.com/aavuzb/project_videos/raw/main/Caf%C3%A9%20Cleanlines.mp4

It is real-time testing by using a Camera. This model always warn the waiter or Café or Restaurant employees about the table is empty or not, or a customer came or not

Self-Ordering Application

GUI projects

Using C++ with Openframeworks

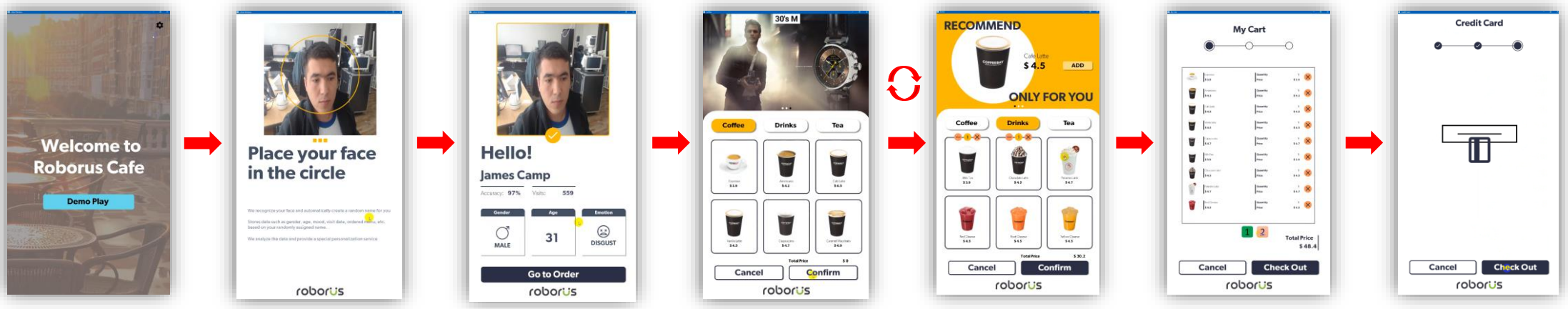
Creating Self-Ordering applications to sell **Coffee**, **Drinks**, and **Tea**
Those GUI applications are written for the KIOSK Machine as bellow images:

Self-Ordering Application

UI projects

I have created two types of self-ordering applications.
The first type of application has a face capturing window. It is very interesting for customers.
Because, Kiosk recommends a product and ads after recognizing Gender, Age, and Emotion

Following images are steps of the first type of self-ordering application



Initial window

Face capturing And sending to captured face to Server

Gender, Age and Emotion recognition result

Kiosk recommends ads or a product after Gender, Age and Emotion recognition

Ordered Products list

Payment process

Please watch the videos on the next page, then you totally understand the working process of this application

Self-Ordering Application : Result

Please enter to download video for knowing how **Self-Ordering Application** works :

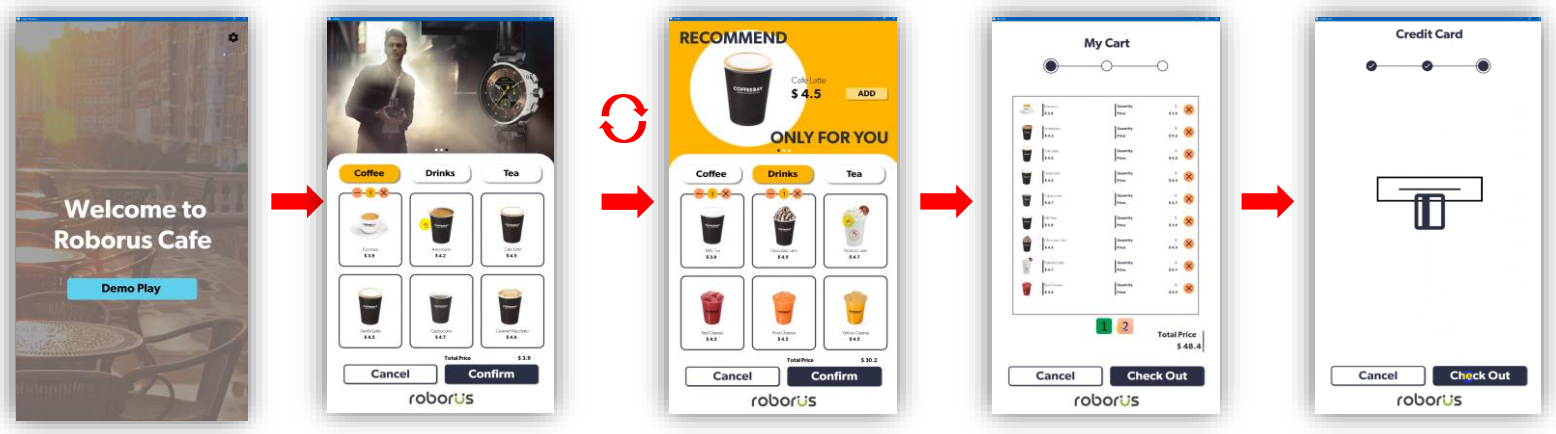
Video link : https://github.com/aavuzb/project_videos/raw/main/Self-Ordering%20Application%201.mp4

Self-ordering application after Gender, Age, and Emotion recognition

GUI projects

The second type of application has no Gender, Age, and Emotion recognition part. Customers can order without Face capturing.

Following images are steps of the second type of self-ordering application



Initial window

Kiosk recommends ads or a product after Gender, Age and Emotion recognition

Ordered Products list

Payment process

Please watch the videos on the next page, then you totally understand the working process of this application

Self-Ordering Application : Result

Self-ordering application without face capturing

Please enter to download video for knowing how **Self-Ordering Application 2** works :
Video link : https://github.com/aavuzb/project_videos/raw/main/Self-Ordering%20Application%202.mp4

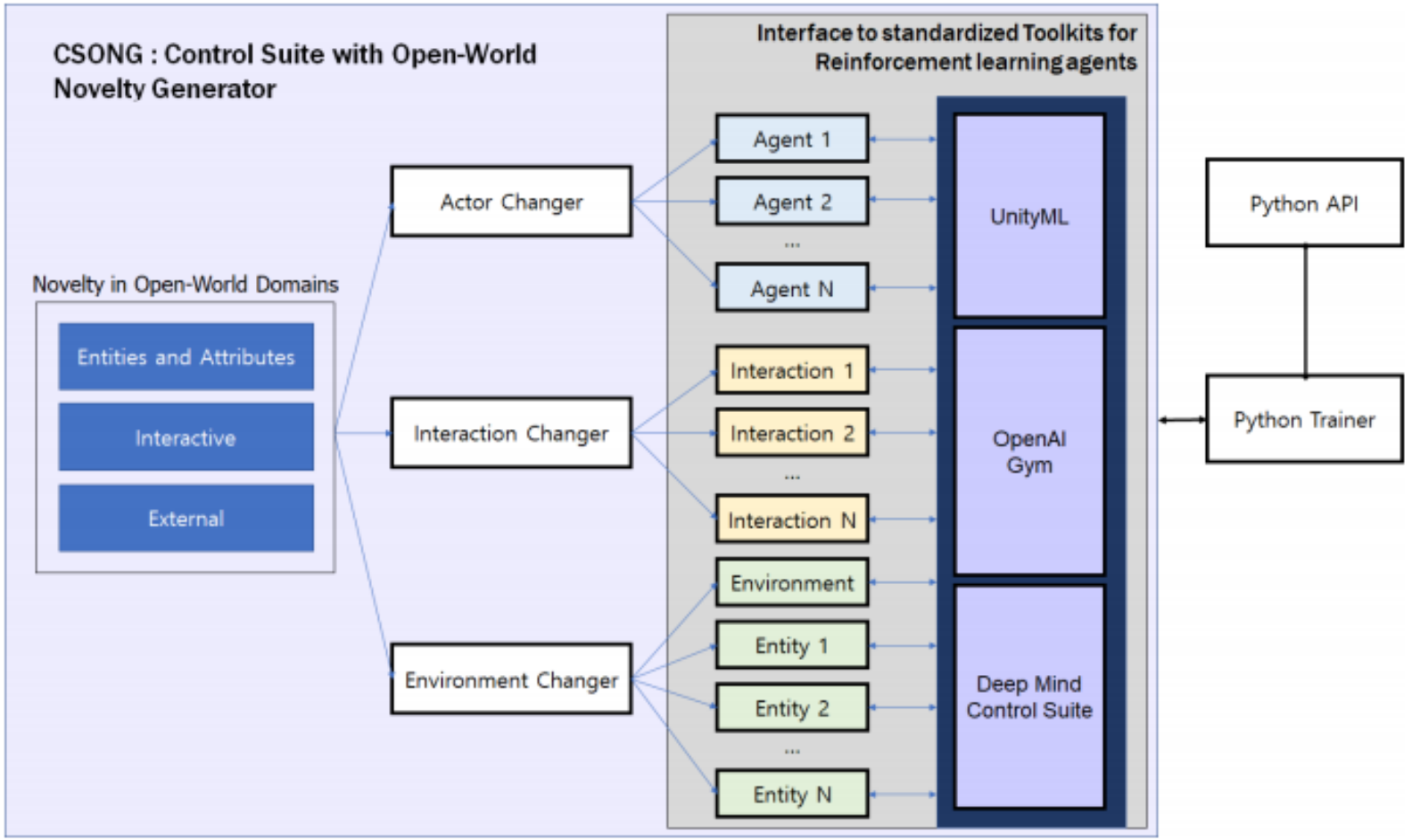
Self-Ordering Application : Result

Similar to the second type of self-ordering application.
However, the design is different from the previous one.

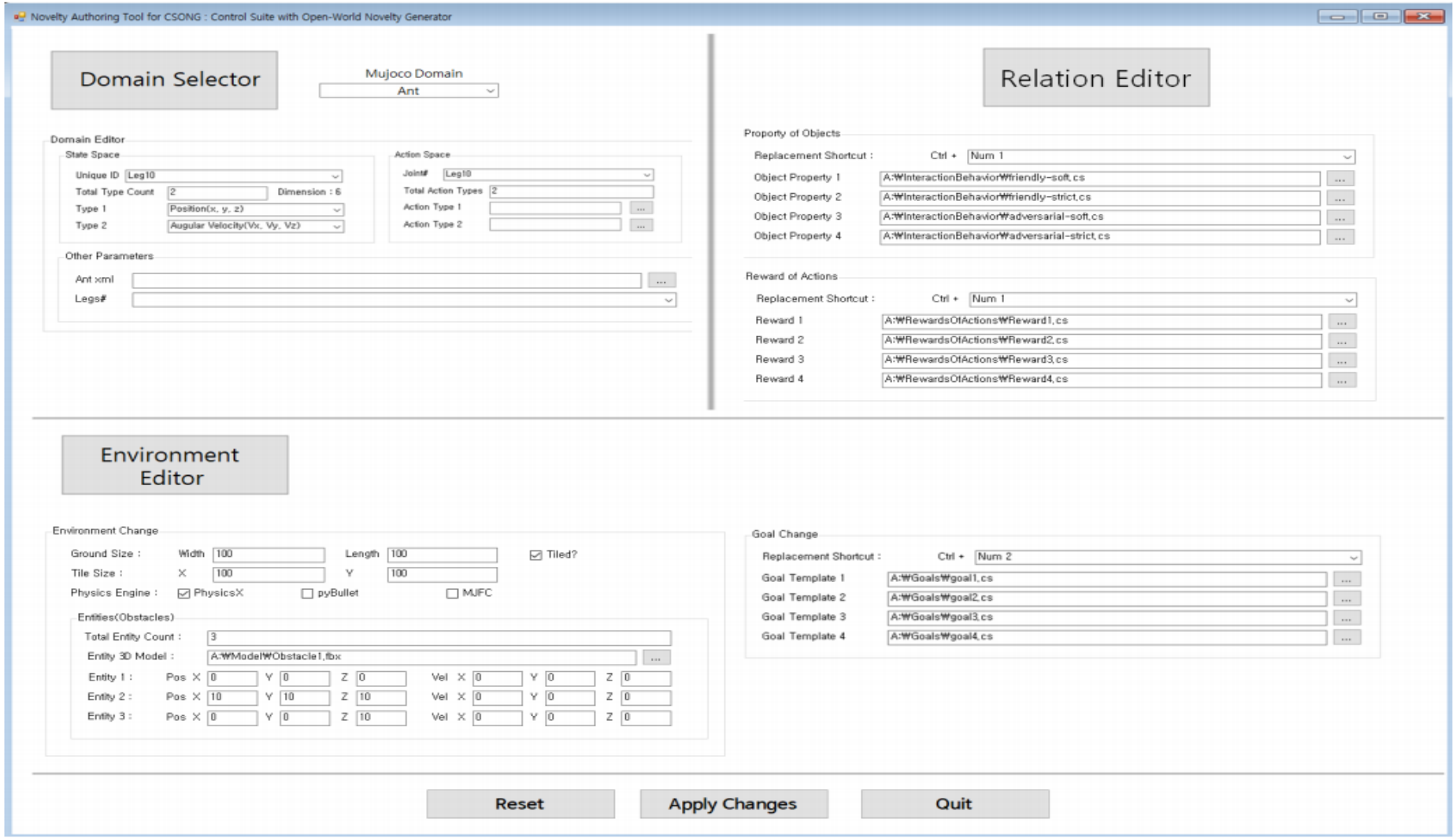
Please enter to download video for knowing how **Self-Ordering Application 3** works :
Video link : https://github.com/aavuzb/project_videos/raw/main/Self-Ordering%20Application%203.mp4

CSONG Development

CSONG : RL Control Suite with Open-World Novelty Generator

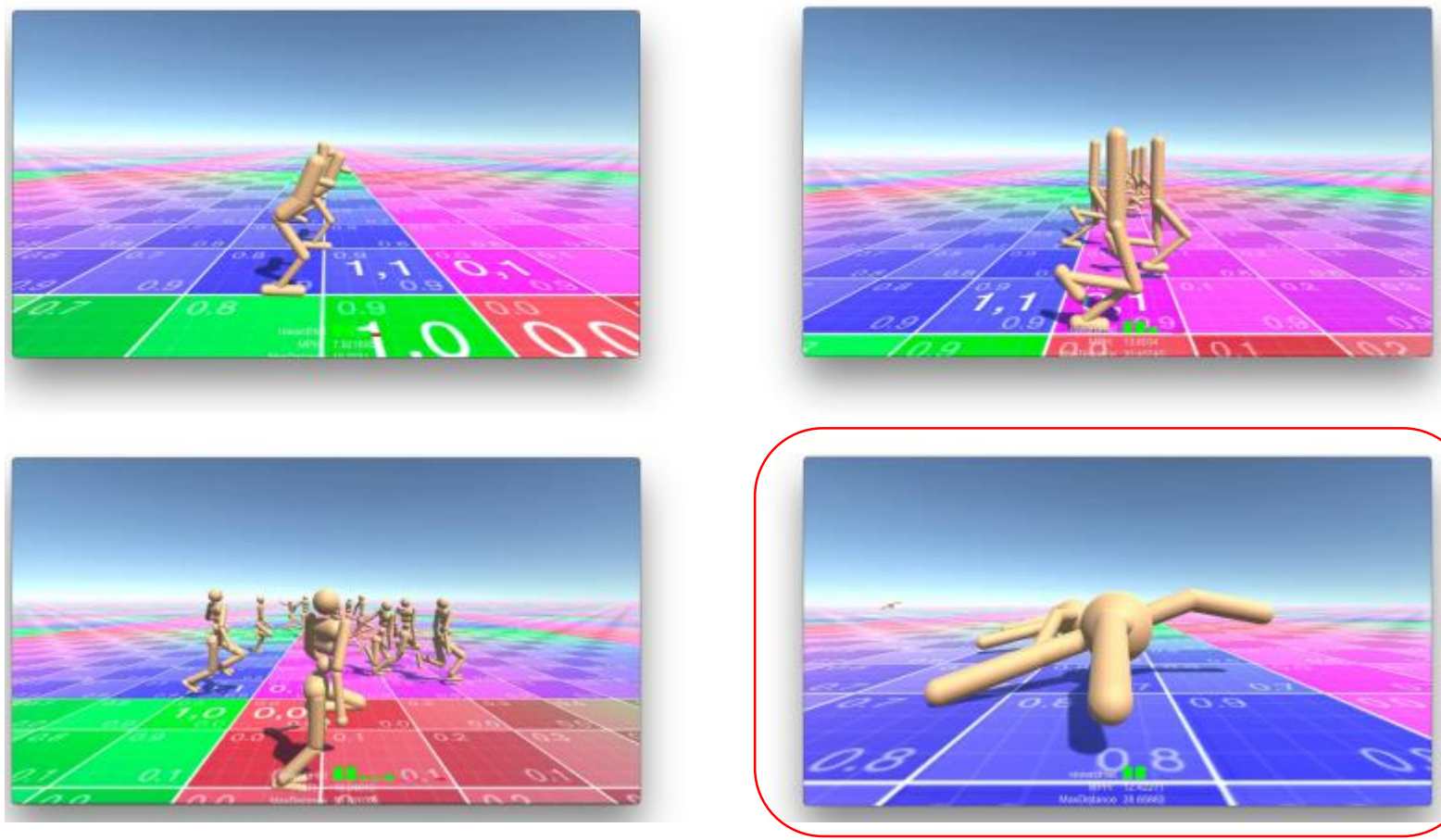


CSONG Development : Server side of CSONG



Server Side is Windows Form Application by C# (.NET Framework)

CSONG Development : Client side of CSONG



I implemented
CSONG for Ant Model.

In future, CSONG will be
Implemented for all Models

Unity ML-Agents. The Marathon Environments

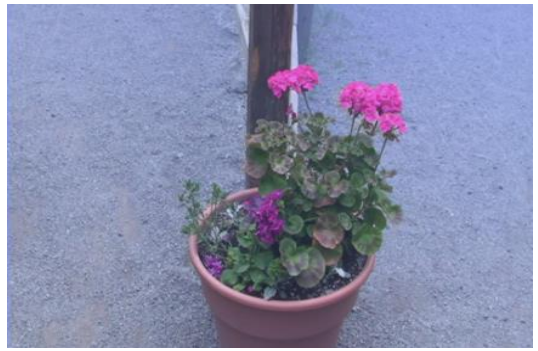
CSONG Development : Final Result

Please enter to download video for knowing how **CSONG** works :

Video link : https://github.com/aavuzb/project_videos/raw/main/CSONG_Result.avi

Computer Vision Researches

Increasing the quality of image using “Non-local image dehazing” algorithm and improving it

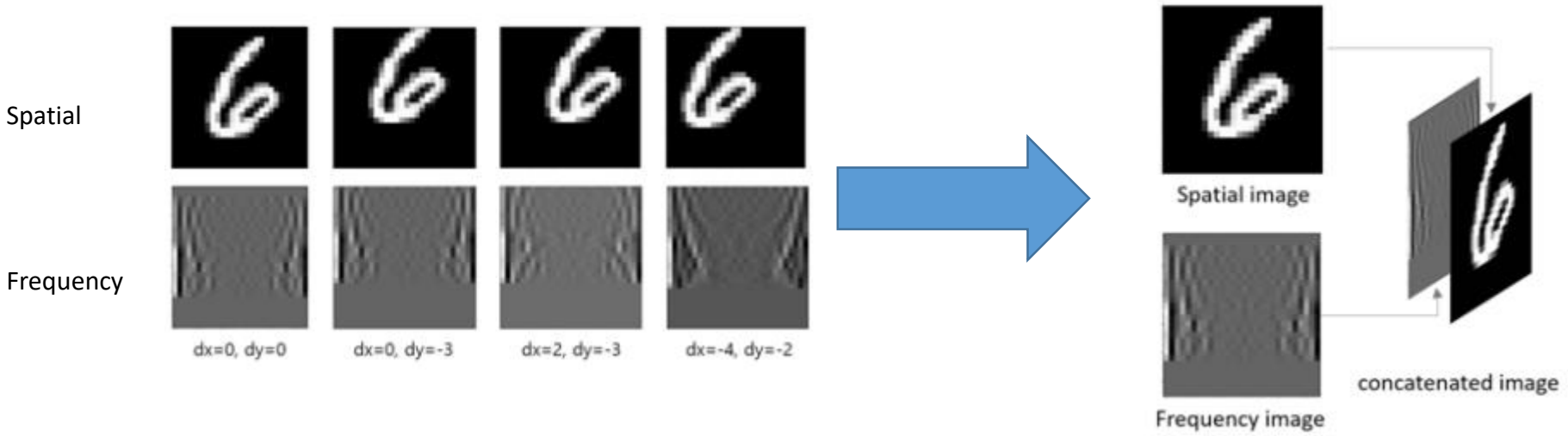


AI Researches and Paper summary

Paper Name: Extending Input Channel Using Global Feature Image for Convolutional Neural Networks

Paper link : https://ieeexplore.ieee.org/document/8960966

Short explanation about paper: All digits are centralized in MNIST. However, if the digit is shifted from the center, the accuracy will decrease. Fast Fourier Transform (FFT) can produces similar frequency image from shifted images.



Paper Name: Extending Input Channel Using Global Feature Image for Convolutional Neural Networks

Paper link : https://ieeexplore.ieee.org/document/8960966

Result: The following table is the accuracy comparison result among spatial, frequency, and concatenated image. You can see, mostly the concatenated achieve higher accuracy than others

Translation (dx, dy)	Spatial image	Frequency image	Concatenated image
0,0	**99.53**	97.72	99.51
0,-3	93.17	84.63	**94.37**
2,-3	92.13	79.60	**93.51**
4,0	92.15	76.39	**93.42**
2,3	91.21	74.51	**93.39**
0,3	94.66	86.33	**94.95**
-4,3	88.77	60.26	**91.07**
-5,0	85.20	61.16	**88.89**
-4,2	82.75	60.43	**88.67**
-3,-2	91.54	71.37	**94.37**
Total	91.11	75.24	**93.21**
deviation	4.71	12.44	**3.15**

Dear CEO/CTO,

Thank you very much for reading my portfolio. I am so sorry if you didn't satisfy my explanations.

However, if you invite me for the interview, I will try to explain all pages more clearly and deeply.

Thank you so much for your attention 

Abduraimjonov Abdurakhmon