

目錄

1 CV

這是我的 CV，包含我的學歷、經歷、專案、目前會的技能等等

2 大學部專題

我們設計了一個 Scalable Serving System for LLM with Kubernetes and Nvidia MIG，依據 inference 的 runtime behavior 動態調整資源用量，並透過 MIG 技術解決 resources fragmentation 的問題

3 專案－乳房腫塊辨識

利用新穎的系統架構來識別乳房腫塊，使用 YOLO 作為 preliminary detector，初步辨別後將結果交給 CNN 做最後的預測

4 專案－液壓手臂顏色分類

我們設計了自己的軟硬體架構，透過電腦進行顏色辨識，將結果 encode 後傳送到 FPGA，再透過 FPGA 控制液壓機械手臂，將不同顏色的方塊夾到指定位置

5 英文檢定

多益檢定證書，我拿到了 915 的成績


6 專業課程成績

- 三主科：計算機結構、計算方法設計、作業系統
- 實驗：軟體設計與實驗、硬體設計與實驗
- 專題領域相關：資料庫系統概論、計算機系統管理
- 其他專業領域：編譯器設計、機器學習概論、

6 課外活動

- 國立清華大學資訊工程學系系學會會長
- 個人網頁
- 國立清華大學通識教育中心通識優秀學生作品獎

BIN LUN LI

 [mike911209](#) |  [my site](#) |  mike911209@gmail.com |  +886 (09) 82-880-498

A passionate programmer who is eager to learn, seeking an opportunity to leverage all my strength into resolving real-world challenges.

EDUCATION

National Tsing Hua University (NTHU)

Sep 2021 - Jun 2025

B.S. in Computer Science

Average **GPA 4.27/4.3** on last 2 semesters.

EXPERIENCE

Teaching Assistant of Operating System

Sep 2024 - Dec 2024

- **Designed** the specifications for Machine problem.
- **Setup** the server environment for class students.

Teaching Assistant of Hardware Design and Lab

Sep 2024 - Dec 2024

- **Developed** the specifications for course materials.
- **Resolved** issues caused by both hardware and software, thereby enhancing my debugging skills.

Director of Student Association of Dept. of CS, NTHU

Sep 2022 - Aug 2023

- **Led a team of 10 people** to organize everything related to CS students.
- **Collaborated** with faculty, industry professionals, and student groups.

PROJECTS

Scalable Serving System for LLM with Kubernetes and Nvidia MIG

- **Scaling** up or out automatically according to LLM inference server's runtime resources usage.
- **Reconfigure** hardware to resolve resources fragmentation.
- **System design**, a complete system that receives HTTP requests and inference automatically, then response to users.

Mass detection in Mammography image

[Link to Demo](#)

- **Brand new architecture (YOLO + CNN)** for mass detection, seamlessly integrating both front end and back end components for our model's inference.
- **Achieve over 70%** precision in the experiment.

Automatic color classification by hydraulic robotic arm on FPGA

[Link to Demo](#)

- **Machine Learning**, utilizing python to recognize colors.
- **Encode** color message into bitstream and transmit them to Digilent Basys3 (an FPGA board).
- **Hardware design**, novel hardware architecture which leverage syringe pump to operate a robotic arm to grasp objects and transport them to designated locations.

SKILLS

Programming Languages

C/C++, Python, Java, Javascript, Verilog, Go

Tools & Libraries

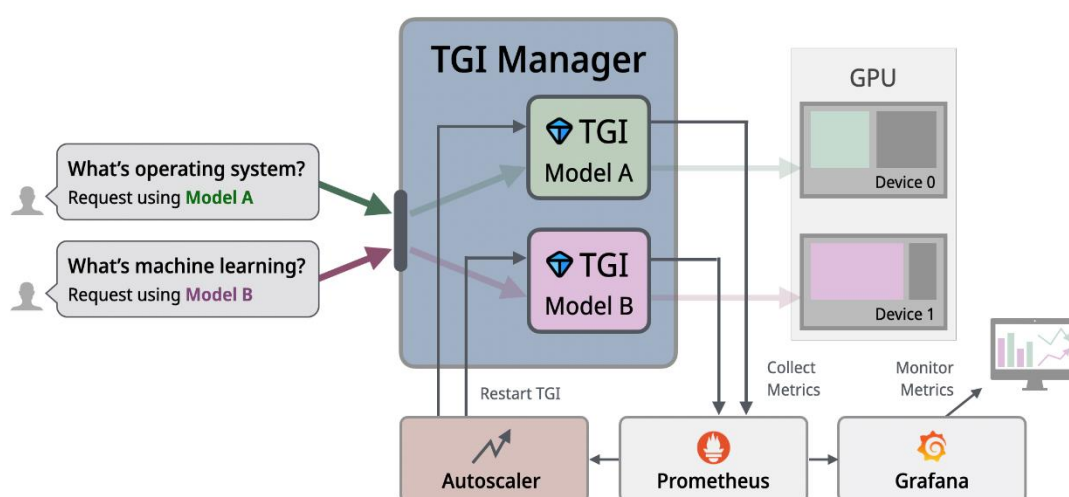
Git, Unix-like shells, Docker, Kubernetes, FPGA, React, SQL, L^AT_EX

Languages

TOEIC Listening and Reading Test 915/990

Scalable Serving System for LLMs with Kubernetes and NVIDIA MIG

專題實作架構圖



專題摘要（實作細節請參考附錄 1）

隨著單一 GPU 算力愈發強大，現今許多 job 無法 fully utilize GPU 導致資源使用率低下，造成 **internal resource fragmentation**，因此這份 project 旨在實作一個 LLM serving system，利用 NVIDIA MIG 技術，physically partition GPU，使多個 inference job share 單一 GPU，提高資源使用率。

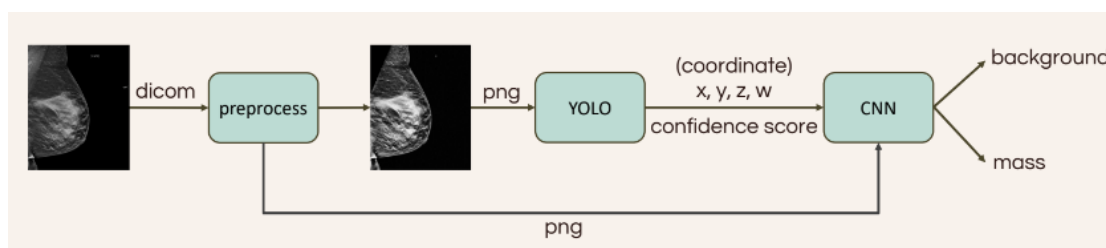
另一方面，LLM inference 一直是雲端上常見的 workload，然而雲端的 workload distribution 通常不 balance，LLM inference 的 KV cache 又會隨著 inference 的過程而持續增大，因此需要一個 auto scaling 技術：當 inference server 的資源使用率過低時，降低分配的資源量；而當許多的 inference requests 到來時，也需要及時分配更多資源，以維持 performance。

因此，我們實作了一個 **Scalable Serving System for LLMs with Kubernetes and Nvidia MIG**。此系統使用 Hugging Face 的 Text generation inference (TGI) framework，接受 user requests，inference 時監測資源使用狀況，**動態調整分配給 inference server 的資源量**，並且透過 Nvidia MIG 技術，我們解決了在 Cloud computing 上常見的 **internal resource fragmentation** 問題，同時也 ensure inference 時的 fault tolerance。

在過程中，我們遇到了諸多挑戰，像是 CUDA process 只能使用一張 MIG GPU，因此我們在設計 scaling policy 時遇到許多限制。又因為 MIG 是非常新穎且特別的 physically GPU partitioning 技術，先前並未有人嘗試過將 Kubernetes 與 MIG 做結合，因此在此份專題中，有許多需要我們自己摸索嘗試。而我們最後也成功實作出這樣一個 system，不僅可以動態調整資源，也解決了 internal resource fragmentation 的問題。

Mass detection in Mammogram images

專案實作架構圖



專案摘要（實作細節請參考附錄 2）

此份專案旨在接收 X 光照片 input，輸出潛在的腫塊座標。為此，我們設計了一個嶄新的系統架構，結合 **YOLO 與傳統 CNN 進行雙階段模型預測**，使用 YOLO 作為 preliminary detector，初步辨識後，將輸出結果交給 CNN，CNN 再進一步專注在更精細的範圍上進行辨識。

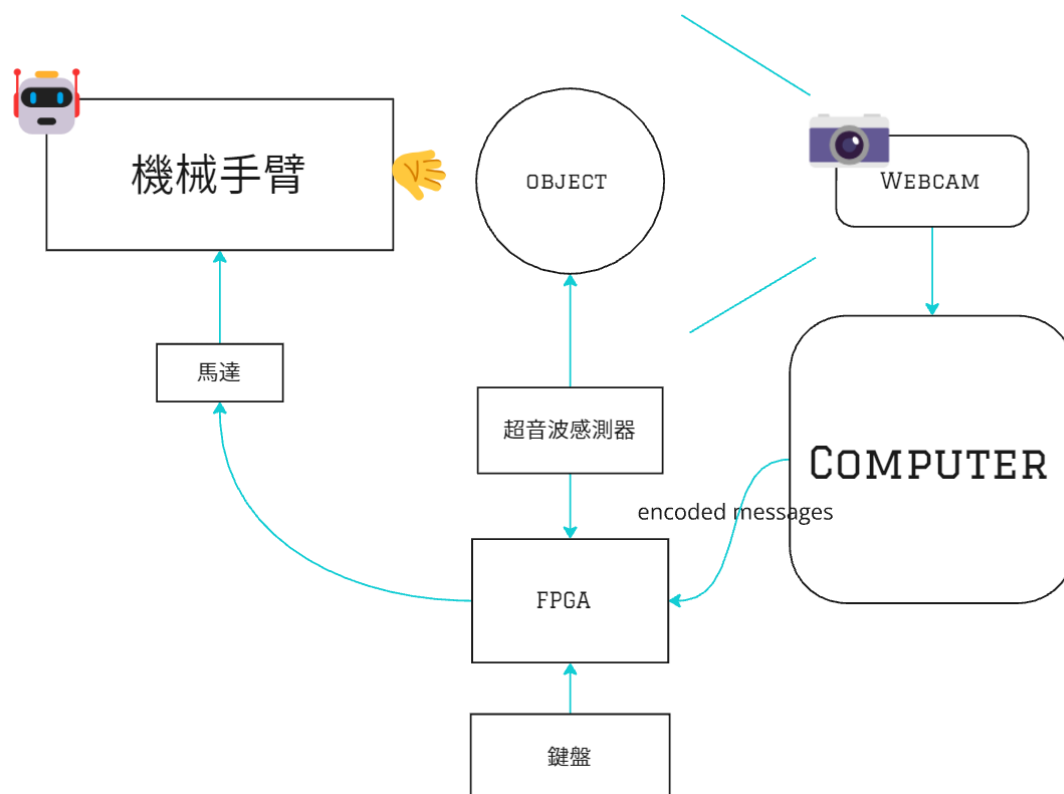
此份專案主要可以分為 3 大部分：圖片預處理、YOLO 模型訓練與 CNN 模型訓練。我在團隊中負責 **YOLO 部分**，嘗試各種圖片預處理，訓練模型後觀察結果準確率。

就結果而言，我們最後的 f1-score 是 0.422，不及傳統的單一模型預測，我們推測原因有 2：其一為 dataset 有許多的錯誤 label，導致模型訓練不佳；其二為 YOLO 經過初步篩選後的 output 對於 CNN 而言過於相像，導致 CNN 被混淆，無法提高準確度。然而，這個創新的想法也給了我們可以嘗試更多不同的技術以優化結果的機會，我們也藉此在實作的過程中有非常多的進步。

全自動液壓手臂顏色分類

Demo: <https://drive.google.com/file/d/1ia0PHZth9sduqlt7a2sNz5kfgA6AjJln>

專案實作架構圖



專案摘要（實作細節請參考附錄3）

在此專案中我們設計了自己的軟硬體架構，硬體架構設計、3D 列印液壓結構、設計電腦與 FPGA 傳輸協定、控制馬達移動邏輯等，我們實作出一個可以自動辨識顏色方塊並將方塊放置於指定位置的全自動液壓機械手臂。

英文檢定

ETS

TOEIC

LISTENING AND READING

OFFICIAL SCORE CERTIFICATE

TOEIC® 臺灣區總代理 忠欣股份有限公司

2F, No. 45, Sec. 2, Fuxing S. Rd., Da'an Dist.

Taipei City 106472, Taiwan (R.O.C.)

李秉綸

LI BING-LUN

Name

2002/12/09

Date of Birth

(yyyy/mm/dd)

24319737

Registration Number

2024/08/25

Test Date

(yyyy/mm/dd)

Individual (August 2024)

Client

LISTENING

470

Your Score

5

495

READING

445

Your Score

5

495

TOTAL SCORE

915

Copyright © 2024 by ETS. All rights reserved.

ETS, the ETS logos, and TOEIC are registered trademarks of ETS in the United States and other countries, used under license in Taiwan.

TOEIC® 臺灣區總代理 忠欣股份有限公司為讓成績使用單位辨識本成績單之真實性，提供智慧手機使用之專屬應用程式服務。成績使用單位可在智慧型手機上下載右方之應用程式，並在網路連線下查閱本成績單之原始內容。

TOEIC成績屬於考生本人之隱私與個人資料，使用本查驗應用程式之用戶，請確認已取得考生同意或具有查驗該成績單之權利，否則請勿使用本應用程式，以免違反個人資料保護法之相關規定。

TOEIC成績保留兩年，可查驗期間為測驗日後兩年內。

TOEIC成績單查驗應用程式

Android版 iOS版

LISTENING

Your scaled score is between 400 and 495. Test takers who score around 400 typically have the following strengths:

- They can infer the central idea, purpose, and basic context of short spoken exchanges across a broad range of vocabulary, even when conversational responses are indirect or not easy to predict.
- They can infer the central idea, purpose, and basic context of extended spoken texts across a broad range of vocabulary. They can do this even when the information is not supported by repetition or paraphrase and when it is necessary to connect information across the text.
- They can understand details in short spoken exchanges, even when negative constructions are present, when the language is syntactically complex, or when difficult vocabulary is used.
- They can understand details in extended spoken texts, even when it is necessary to connect information across the text and when this information is not supported by repetition. They can understand details when the information is paraphrased or when negative constructions are present.

To see weaknesses typical of test takers who score around 400, see the *Proficiency Description Table.

READING

Your scaled score is close to 450. Test takers who score around 450 typically have the following strengths:

- They can infer the central idea and purpose of a written text, and they can make inferences about details.
- They can read for meaning. They can understand factual information, even when it is paraphrased.
- They can connect information across an entire text, and they can make connections between two related texts.
- They can understand a broad range of vocabulary, unusual meanings of common words, and idiomatic usage. They can also make distinctions between the meanings of closely related words.
- They can understand rule-based grammatical structures. They can also understand difficult, complex, and uncommon grammatical constructions.

To see weaknesses typical of test takers who score around 450, see the *Proficiency Description Table.

ABILITIES MEASURED

PERCENT CORRECT OF ABILITIES MEASURED

0%

Your Percentage

100%

Can infer gist, purpose and basic context based on information that is explicitly stated in short spoken texts

87

0%

100%

Can infer gist, purpose and basic context based on information that is explicitly stated in extended spoken texts

88

0%

100%

Can understand details in short spoken texts

100

0%

100%

Can understand details in extended spoken texts

94

0%

100%

Can understand a speaker's purpose or implied meaning in a phrase or sentence

80

0%

100%

ABILITIES MEASURED

PERCENT CORRECT OF ABILITIES MEASURED

0%

Your Percentage

100%

Can make inferences based on information in written texts

92

0%

100%

Can locate and understand specific information in written texts

100

0%

100%

Can connect information across multiple sentences in a single written text and across texts

83

0%

100%

Can understand vocabulary in written texts

88

0%

100%

Can understand grammar in written texts

90

0%

100%

※ HOW TO READ YOUR SCORE REPORT:

Percent Correct of Abilities Measured:
Percentage of items you answered correctly on this test form for each one of the Abilities Measured. Your performance on questions testing these abilities cannot be compared to the performance of test-takers who take other forms or to your own performance on other test forms.

Note: TOEIC scores more than two years old cannot be reported or validated.

Copyright © 2024 by ETS. All rights reserved.

ETS, the ETS logos, and TOEIC are registered trademarks of ETS in the United States and other countries, used under license in Taiwan.

專業課程成績

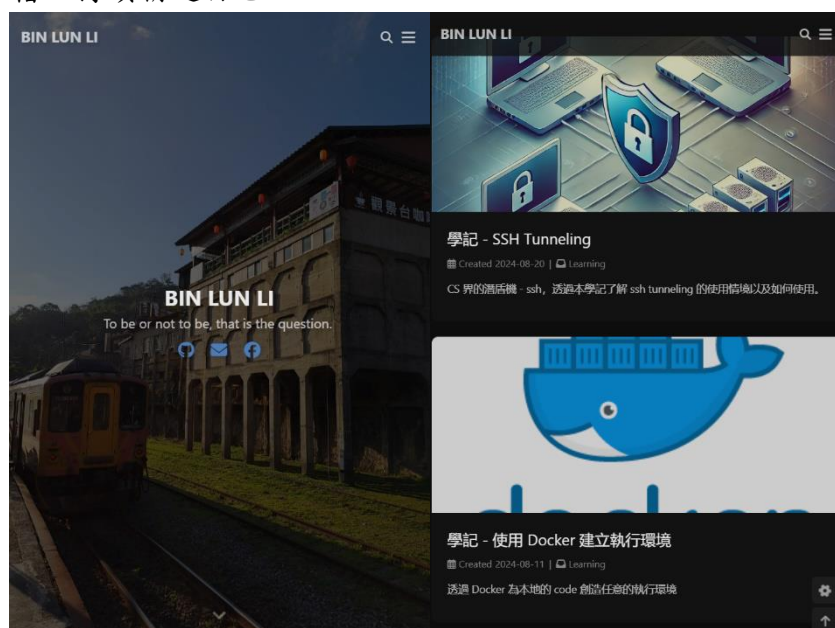
修習課程名稱	授課老師	成績 (GPA / T 分數)
計算機結構	王廷基教授	A+ / 57.97
計算方法設計	盧錦隆教授	A- / 57.91
作業系統	周志遠教授	A+ / 60.78
軟體設計與實驗	吳尚鴻教授	A+ / 65.14
硬體設計與實驗	黃稚存教授	A+ / 64.83
資料庫系統概論	吳尚鴻教授	A+ / 61.26
計算機系統管理	張君天教授	A+ / 55.65
機器學習概論	郭柏志教授	A+ / 60.27
編譯器設計	李政崑教授	A+ / 59.72
機率	許秋婷教授	A / 55.90
邏輯設計	劉怡君教授	A+ / 60.92
Python 語言程式入門	周百祥教授	A+ / 59.51

課外活動

- 國立清華大學資訊工程學系系學會會長
統整疫情後重創的系學會，任內辦理諸多活動、聯絡系上感情，培養領導能力。



- 個人網頁
Link: <https://mike911209.github.io/>
紀錄我學習、經歷過的事情，在分享的過程中我也對該主題變得更加熟悉，藉以持續精進自己。



- 國立清華大學通識教育中心通識優秀學生作品獎

Link: <https://cge.site.nthu.edu.tw/p/406-1573-276357,r10609.php>

不僅僅限於主科，對於通識課程我也是全力以赴，榮獲 112 年上學期的通識優秀學生作品獎。



附錄 1

Scalable Serving System for LLM with Kubernetes and Nivida MIG

Scalable Serving System for Large Language Models with Kubernetes and NVIDIA MIG

BIN-LUN LI

Derek (Kai-Jun) Lin

Li-Shang Lin

Abstract

As the computing power of GPUs continues to grow, many workloads are unable to fully utilize an entire GPU, leading to internal resource fragmentation. To address this, technologies enabling GPU sharing have become crucial. NVIDIA’s A100 GPU introduces **Multi-Instance GPU (MIG)**, a feature that physically partitions a single GPU into multiple slices, allowing more efficient resource allocation. With MIG, the A100 can achieve significantly higher efficiency.

Meanwhile, **serverless platforms** have gained popularity with the rise of cloud computing. Due to their resource management-free nature, auto-scaling capabilities, and cost-efficiency, serverless platforms have become a preferred choice for users, making them a key focus in cloud computing development.

This project aims to combine these two technologies by developing a Scalable Serving System for Large Language Models (LLMs) using Kubernetes and NVIDIA MIG. The system dynamically scales based on LLM inference throughput while maintaining high GPU utilization through MIG partitioning.

1 Introduction

This project introduces a Kubernetes-based serving system for language models that leverages NVIDIA Multi-Instance GPU (MIG) technology to partition GPU resources into multiple isolated instances. By enabling more flexible and efficient use of computational resources, the system can allocate different GPU slices based on the performance needs of LLM inference, optimizing resource utilization and improving overall throughput.

The system integrates Hugging Face’s **Text Generation Inference (TGI)** framework to manage and serve language model inferences. A key feature is its real-time monitoring of each TGI instance’s workload, allowing the system to automatically adjust GPU resource allocation for inference tasks. This adaptive resource management ensures an optimal balance between efficiency and throughput, catering to varying workloads and enhancing the scalability and flexibility of LLM serving.

2 Background

2.1 Kubernetes

Kubernetes is an open-source platform that automates the deployment, scaling, and management of containerized applications. It allows developers to deploy applications packaged in containers (e.g. Docker) and manage them across a cluster of machines.

Kubernetes is an open-source platform that automates the deployment, scaling, and management of containerized applications. It allows developers to deploy applications packaged in containers (e.g., Docker) and manage them across a cluster of machines. Kubernetes handles the automatic scheduling and orchestration of containers, ensuring efficient resource utilization. It plays a critical role in cloud environments and serverless platforms

by enabling dynamic resource allocation and scaling without manual intervention. Key features include load balancing, dynamic scaling based on resource demands, and ensuring high availability, reliability, and scalability for modern cloud-native and serverless applications.

In our system, containers run inside Kubernetes cluster, which allows us to interact with the containers using Kubernetes’ predefined API.

2.2 Multi-Instance GPU

NVIDIA Multi-Instance GPU (MIG) is a technology that enables a single GPU to be physically partitioned into multiple, fully isolated instances, each with dedicated compute, memory, and cache resources. A MIG-enabled GPU can only be partitioned into fixed types of slices. Table 1 shows all the MIG profiles available on an A100 GPU, and Figure 1 provides a visualization of how independent MIG-enabled GPU instances operate.

MIG allows multiple workloads or users to run independently on the same GPU, optimizing resource utilization and efficiency. MIG is particularly useful in cloud computing, AI, and high-performance computing environments, where smaller tasks or multiple users can share GPU resources without performance interference.

We leverage MIG to address internal fragmentation issues (where a single task may not fully utilize an entire GPU) by partitioning the GPU into multiple smaller instances, each running an LLM inference server. This not only improves resource utilization but also enables fault tolerance through physical GPU partitioning.

Table 1: Complete list of MIG profile on an A100 GPU.

Slice	SM	Memory	Cache	Max Count
7g.40gb	7 GPC	40 GB	Full	1
4g.20gb	4 GPC	20 GB	4/8	1
3g.20gb	3 GPC	20 GB	4/8	2
2g.10gb	2 GPC	10 GB	2/8	3
1g.5gb	1 GPC	5 GB	1/8	7

2.3 Text Generation Inference

Hugging Face Text Generation Inference (TGI) is a highly optimized, production-ready solution for deploying large language models. It supports a wide range of models available in the Hugging Face ecosystem, ensuring broad compatibility.

TGI is easy to deploy using its Docker image, simplifying the integration process. Additionally, it supports request batching, which enhances throughput and improves resource efficiency, making it an ideal choice for scalable, high-performance inference in real-time applications.

TGI is used as the inference server in our system. Through the acceleration provided by TGI, the inference process is optimized.

2.4 Prometheus

Prometheus is an open-source monitoring and alerting toolkit that supports monitoring a wide variety of systems, including containers and microservices. It stores metrics using a powerful data model that allows for flexible querying and is designed for reliability and scalability, particularly in cloud-native environments.

The TGI itself can automatically expose multiple metrics that can be collected by the Prometheus monitoring server. By introducing Prometheus, we can seamlessly collect statistics related to our server operations, monitor workloads, and adjust resources as needed.

3 Motivation

We are designing a language model serving system that can handle requests **utilizing different models**. This approach is driven by several key factors:

- Some tasks may require domain-specific knowledge or specialization that is better suited to certain models.
- While larger models are powerful, it can be resource-intensive, and for simpler tasks, smaller and more efficient models are preferable.

However, one of the significant challenges in LLM inference is managing memory usage, which includes not only the model weights but also the growth of the KV cache. Additionally, when a large volume of inference requests arrives in a short period, the workload can increase dramatically. If the system cannot automatically scale computational resources to meet this demand, it may result in resource shortages, creating bottlenecks. Furthermore, when different jobs share a single GPU without MIG, resource contention may occur, preventing proper distribution of memory and computational resources as different inference jobs compete for the same resources, resulting in performance degradation.

To address these challenges, we aim to utilize a serverless Kubernetes system. The serverless system offers dynamic scaling and resource provisioning without manual intervention, making it ideal for fluctuating workloads. Kubernetes’s ability to manage containerized applications and efficiently allocate resources allows us to deploy language models in a scalable, modular fashion. By combining serverless platforms with Kubernetes, we can ensure auto-scaling of compute resources based on demand, leading to better cost efficiency, resource optimization, and the ability to handle spikes in traffic while maintaining high system availability.

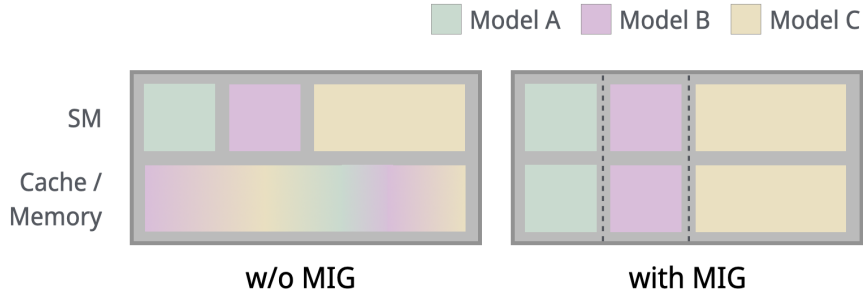


Figure 1: Inference Resource Allocation: Without vs. With MIG

Figure 1 illustrates a scenario where three different inference requests are served using distinct models on a MIG-capable GPU. The left-hand side of the figure shows the case without MIG, where, as the workload of each model fluctuates, there might be interference between models. This can lead to **resource contention**, where multiple inference jobs compete for the same resources, potentially resulting in performance degradation. However, by leveraging MIG (as shown on the right-hand side of Figure 1), we can allocate resources more precisely to each inference job. This allows for finer control over GPU resources assigned to each model, ensuring they scale appropriately with the workload, avoiding resource competition, and improving overall efficiency. Additionally, it takes advantage of the physical GPU partitioning provided by MIG.

To address the issue of resource competition when sharing GPU, we have decided to leverage NVIDIA’s MIG technology. MIG enables us to physically partition a GPU, such as the A100, into smaller, isolated GPU instances, each with its own dedicated memory, cache, and compute cores. This isolation ensures that different language models can run on separate instances without interfering with one another.

Our system integrates MIG to dynamically allocate resources based on the specific needs of each model, ensuring stable performance even under heavy workloads. The MIG technology also enhances performance by eliminating the problem of resource contention through physically partitioning GPU. By continuously monitoring key metrics (which will be elaborated in Section 4.4), the system intelligently adjusts the GPU capacity allocated to each model without affecting other inference processes. When necessary, the system can automatically reconfigure the MIG partitions, minimizing **resource fragmentation** and ensuring efficient utilization of the available hardware.

This approach allows us to optimize inference performance across multiple models, ensuring that tasks are handled with the appropriate portion of computational resources while preventing resource contention.

4 System Design

4.1 Overview

Our system is composed of four main components:

- **TGI manager**
- **TGI** (inference server)
- **Autoscaler**
- **Prometheus** (metrics scraper)

When a user submits a request to the TGI manager, the request contains the model to be used and the input prompt. Upon receiving the request, the TGI manager dispatches it to the corresponding TGI worker handling the specified model.

During inference, TGI exposes metrics which serves as a key factor for scaling. Prometheus scrapes the metrics exposed by TGI at regular intervals, storing the data in its database, allowing the autoscaler to query the metrics. If the data stored in the Prometheus database indicates that the model is overloaded, the autoscaler initiates the scaling routine to ensure that latency remains under a defined threshold.

Next, we will elaborate on each component in detail.

4.2 TGI Manager

TGI manager oversees multiple TGI instances running on MIG slices. When a request comes in, it contains two key pieces of information:

- **Model specification:** The specific model that needs to be used for inference.
- **Input prompt:** The data or query that the model will process.

TGI manager uses these information to route the request to the appropriate TGI instance running on a partitioned MIG slice.

Figure 2 provides an overview of TGI manager’s workflow, which initiates the inference server for the selected model and allocates smallest size of MIG slice that can fit the model. For each model, we run a dedicated TGI instance, which is served on a specific GPU slice. These GPU slices are allocated dynamically using MIG, allowing multiple instances to run on the same GPU hardware with isolated resources.

4.3 Autoscaler

The autoscaler monitors metric for each TGI instance and the associated GPU slice, the **TGI request mean time per token duration**: a key metric that reflects the workload of a specific inference request. This metric is exposed by TGI and can be collected through the Prometheus endpoint. It measures the mean time spent generating a token. We primarily use this metric to determine whether to adjust the resource allocation for a single TGI instance.

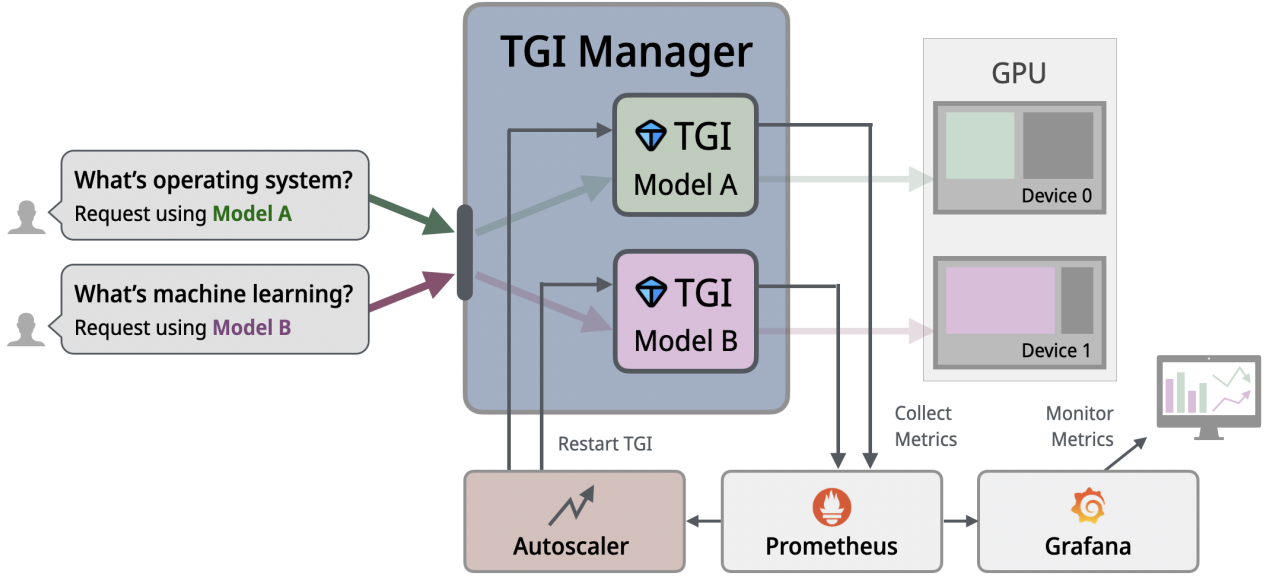


Figure 2: System Overview

Figure 2 provides an overview of the autoscaler. The TGI instance exposes metrics, which Prometheus stores in its database. Based on these workload metrics, the autoscaler dynamically adjusts the system by restarting the TGI instance with an appropriately sized GPU slice to match the current demand.

4.4 Scaling Policies

The autoscaler monitors the TGI request duration metric to determine whether to scale up or down the resources allocated to a specific TGI. This allows the system to dynamically adjust the resource slice given to a TGI based on the workload.

To enhance system flexibility, we propose that when a user initializes a new TGI request with a model, they also specify an **intended SLO**. This SLO defines the acceptable mean time per token for the TGI request. The system will then use this value as the threshold for scaling decisions. If the monitored mean time per token exceeds the user-specified SLO, a scale-up action will be triggered automatically, ensuring optimal performance according to the user’s requirements.

The autoscaling policy operates as follows:

- When requests use a new model and the corresponding TGI is not yet running, the system allocates the smallest resource slice that can fit the model.
- If the duration exceeds or falls below the threshold, a scaling action is triggered. A new TGI instance is then started with a resized resource slice.
- Once the new TGI instance is fully initialized, the system redirects incoming requests to it and terminates the old TGI instance.

By following this approach, the system can dynamically reassign resources to TGIs based on real-time workload demands. Figure 3 illustrates this process, where the line represents the request mean time per token duration. The system uses this value to decide whether to scale up or down.

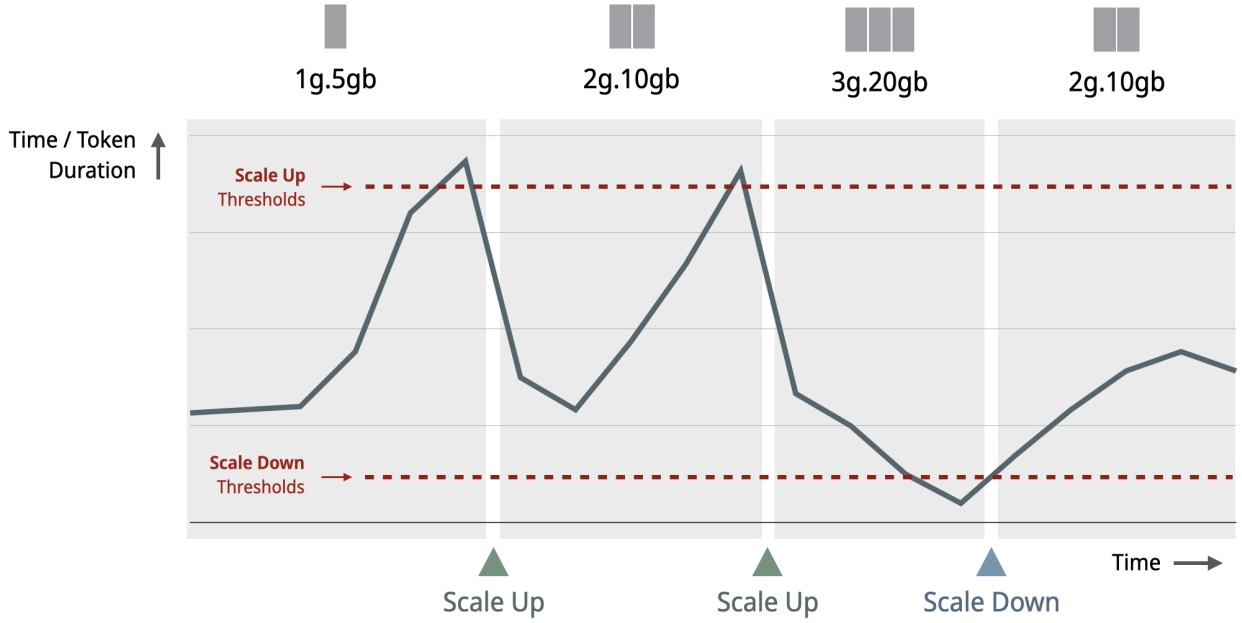


Figure 3: Scaling Policies Overview

5 Conclusion

The Scalable Serving System for LLMs with Kubernetes and NVIDIA MIG dynamically monitors inference server’s workloads and adjusts the resource allocation to each TGI. With MIG technology, the system not only solve the internal resources fragmentation but also distribute resources much more efficiently, ensuring inference requests for each model are handled appropriately.

However, there are some limitations with TGI and the PyTorch framework, particularly the lack of robust support for MIG. This prevents us from splitting models into multiple shards and allocating them to different slices. By enabling this capability, we could manage heavier workloads more effectively, distributing portions of the model across different slices while using a single TGI instance, which would increase inference throughput.

Currently, our scaling policy primarily relies on monitoring request handling duration metrics. However, incorporating more diverse metrics and using a more comprehensive set of data to define what “**workload**” truly represents could further optimize the system. This might even involve leveraging machine learning techniques or allowing users to specify more custom SLOs based on their needs, which would enhance the system’s ability to meet diverse requirements.

Additionally, leveraging Kubernetes’s serverless capabilities allows the system to auto-scale efficiently in response to fluctuating workloads. The combination of Kubernetes and serverless infrastructure has been critical in enabling rapid deployment, seamless scaling, and resource optimization. By using this approach, the system can automatically allocate the necessary compute resources without manual intervention, ensuring that spikes in traffic are handled smoothly and cost-effectively.

Furthermore, if in the future we are able to utilize **live migration** for GPU, we could address the issue of downtime associated with scaling the TGI. Currently, scaling often requires spinning up additional instances of TGI to avoid disruption, which consumes extra computational resources. With live migration, the system could seamlessly transfer workloads between GPUs without interrupting active inference requests and model states. This would eliminate the need for redundant TGI instances handling the same model, reducing computational overhead and enhancing resource efficiency during scaling events.

This system demonstrates how MIG, Kubernetes, and serverless architecture can work together to ensure higher inference throughput. We aim to explore alternative inference frameworks, or even create a custom solution, to better fit our needs. By doing so, we can better utilize available resources, allowing the full power of language models to be applied more efficiently and effectively.

References

- [1] Xinning Hui, Yuanchao Xu, Zhishan Guo, and Xipeng Shen. Esg: Pipeline-conscious efficient scheduling of dnn workflows on serverless platforms with shareable gpus. In *Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing*, pages 42–55, 2024.
- [2] Baolin Li, Viay Gadepally, Siddharth Samsi, and Devesh Tiwari. Characterizing multi-instance gpu for machine learning workloads. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 724–731. IEEE, 2022.
- [3] Baolin Li, Tirthak Patel, Siddharth Samsi, Vijay Gadepally, and Devesh Tiwari. Miso: exploiting multi-instance gpu capability on multi-tenant gpu clusters. In *Proceedings of the 13th Symposium on Cloud Computing*, pages 173–189, 2022.
- [4] Cheng Tan, Zhichao Li, Jian Zhang, Yu Cao, Sikai Qi, Zherui Liu, Yibo Zhu, and Chuanxiong Guo. Serving dnn models with multi-instance gpus: A case of the reconfigurable machine scheduling problem. *arXiv preprint arXiv:2109.11067*, 2021.
- [5] Bingyang Wu, Zili Zhang, Zhihao Bai, Xuanzhe Liu, and Xin Jin. Transparent {GPU} sharing in container clouds for deep learning workloads. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 69–85, 2023.
- [6] Yanan Yang, Laiping Zhao, Yiming Li, Huanyu Zhang, Jie Li, Mingyang Zhao, Xingzhen Chen, and Keqiu Li. Infless: a native serverless system for low-latency, high-throughput inference. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 768–781, 2022.

附錄 2

Mass detection in Mammogram images

Mass Detection in Mammogram Images

Pinshun Wang

Fenyu Hsieh

Fangyu Hsu

Yahui Chang

BingLun Li

Iting Hsueh

Abstract—Mass detection in mammogram images is a critical aspect of early breast cancer diagnosis. This project aims to develop an efficient model for accurately identifying and marking masses in mammograms. The proposed approach involves preprocessing DICOM (Digital Imaging and Communications in Medicine) images and utilizing YOLO (You Only Look Once) for initial mass region identification. Subsequently, a CNN (Convolutional Neural Network) is employed to refine and evaluate the likelihood of each identified region being a mass. Additionally, a user-friendly web interface is designed for medical personnel to upload DICOM images directly.

I. INTRODUCTION

Breast cancer is one of the most prevalent cancers among women globally. Early detection is crucial for treatment and patient survival. Identifying abnormalities in breast X-ray imaging and determining the presence of masses are essential components of this process. Therefore, the goal of this project is to develop a model capable of accurately identifying and marking breast masses in mammography. Our dataset is sourced from EMBED (Emory Breast Imaging Dataset), comprising a large number of DICOM images that combine breast cancer data from different regions and diagnostic images. We selected approximately 6000 DICOM images meeting specific requirements for model training and evaluation. However, DICOM is a medical imaging format used for storing and transmitting medical images. Therefore, we converted it to PNG format to better meet the subsequent needs of model analysis and we also apply a series of preprocessing to the images before using it in our model. When conceptualizing our model architecture, we found two main approaches from existing research: one utilizing specific algorithms for mass separation followed by CNN analysis, and the other relying solely on YOLO for detection and classification. Ultimately, we decided to combine these two approaches. YOLO is renowned for its fast real-time object detection, quickly identifying regions in the image that may contain masses. Subsequently, CNN is used to further analyze and enhance the accuracy of detection in identified potential regions. The training process involves using 5000 images for YOLO training and 618 for validation, utilizing the YOLOv7 model. CNN training involves 3328 images of masses and backgrounds, using CovNeXt and ResNeXt models. We merged YOLO and CNN for final evaluation, striking a balance between YOLO detection threshold and CNN performance.

II. METHODS

A. Overview

The concept of our framework is shown in the Fig. 1. In our training phase, DICOM files undergo an initial preprocessing

step to extract features with enhanced clarity, resulting in PNG-formatted images. Subsequently, these images are separately input into both Convolutional Neural Network (CNN) and You Only Look Once (YOLO) architectures for model training. YOLO takes ROI coordinates from metadata as ground truth to train a model capable of marking the areas that might be masses. CNN is trained to discern whether specific features in the images indicate the presence of masses.

In summary, our approach involves preprocessing the dataset, followed by YOLO identifying potential mass features based on ROI coordinates from metadata. These identified features are then input to CNN, which makes the final determination on whether they correspond to masses.

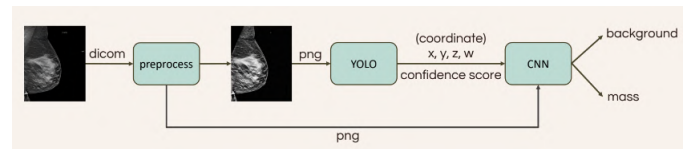


Fig. 1. Overall Design Framework.

B. Data

1) *Datasets*: We chose EMBED as our dataset for its extensive collection of mammographic exams, which includes detailed Regions of Interest annotations for masses.

The EMBED (Emory Breast Imaging Dataset) utilized in this study is a comprehensive collection of mammographic exams, encompassing a total of 364,000 screenings and diagnostic tests from 110,000 patients across four hospitals over an eight-year period. This dataset predominantly includes 2D and C-view mammographic images. This dataset is instrumental in facilitating advanced studies in breast imaging and cancer diagnosis. [1]

2) *DICOM Preprocessing*: The purpose of preprocessing DICOM is to convert the file format into a format that the model can read and to enhance the efficiency of subsequent preprocessing steps. The following will sequentially introduce how to convert DICOM to PNG, improve the readability of PNG, and synchronize the metadata table within the dataset.

• Step 1: Transfer DICOM to PNG.

Below is the information we accessed in the DICOM files

- **To ensure uniform breast positioning:** The attributes Laterality, ViewPosition, and PatientOrientation in DICOM are utilized to determine if the image needs flipping.

- **To obtain the pixel array mapping from DICOM to an 8-bit PNG:** Access to the VOI LUT Function within the DICOM file is necessary.
- **To comprehend pixel value interpretation:** The Photometric Interpretation attribute in DICOM provides relevant information.
- **For windowing in the resulting PNG from DICOM:** It is essential to read the DICOM attributes for window center and window width.

- **Step 2: Windowing**

By utilizing window width (WW) and window center (WC), it is possible to map the pixel array of DICOM images to an 8-bit PNG pixel array, highlighting the crucial color regions of the image. This facilitates subsequent preprocessing and model operations for improved analysis and interpretation.

- **Step 3: Update The Metadata Table**

Standardizing the format of the 'ROI coord' field in the unified metadata table to enhance the smooth retrieval and utilization of data by subsequent models and preprocessing procedures.

3) *PNG Preprocessing:* Preprocessing of images is an essential step in readying data for both training and application in the model. This process ensures uniformity in the input to the neural network, enhances image clarity. We used the following methods to preprocess raw mammogram X-ray images from the EMBED database. [2]

- **Step 1: Normalization of Breast Orientation**

Aligning all PNG files to a uniform orientation for enhanced overall consistency, facilitating subsequent model training and preprocessing.

- **Step 2: Handle Images with Paddle**

When capturing breast images, different types of paddles can be utilized to focus on specific tissues for observation. Among them, spot compression and magnification are the paddle types we aim to address. Spot compression involves compressing a specific tissue area to enhance its visibility, allowing for targeted observation of that region; magnification, on the other hand, involves enlarging a specific tissue area, making it the focal point in the imaging. Since the paddles used in breast imaging are made of metal, the pixels in the paddle can be particularly intense, leading to potential confusion in the results of subsequent models and preprocessing. Therefore, it is necessary to employ techniques to remove the paddles and extract the underlying tissue information from within the paddles.

- **Step 3: Remove Noise on Mammogram Images**

Raw mammogram X-ray image has noise and small holes in the image. Using bilateral filtering is able to reduce the noise and fill in the holes in the image. We import bilateral filter from OpenCV. Subsequent to the application of the bilateral filter algorithm, an attempt the switching bilateral filter was made. However, this endeavor was unsuccessful. It requires processing each

pixel individually, however our images include extensive pixel count resulting in a prohibitively time-consuming computation.

- **Step 4: Remove Artifacts by Selecting Breast Area**

In a raw mammogram X-ray image, labels manually added by individuals may affect the accuracy of the model in detecting masses. Therefore, it is necessary to remove these labels and retain only the breast area. Since labels are typically smaller compared to the breast area, we can use Morphological Transformations to eliminate them. For this purpose, we import the "opening" function from OpenCV to effectively remove the labels. The results of the mammogram image after breast tissue enhancement is illustrated in Fig. 2.

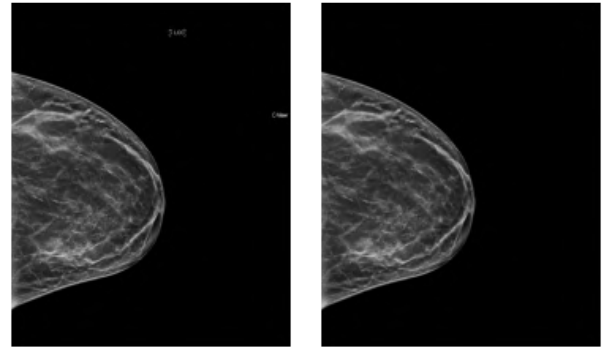


Fig. 2. Example of a mammogram image result of removing artifacts by selecting breast area: the original image and the image that the artifacts removed. (from left to right).

- **Step 5: Enhancing the Contrast of Breast Tissue**

This step involves the amplification of breast tissue contrast. We achieved this through using CLAHE algorithm, designed to intensify the contrast in grayscale images. CLAHE operates by dividing the image into small blocks, known as tiles, and then applying histogram equalization to each of these independently. The results of the mammogram image after breast tissue enhancement is illustrated in Fig. 3.

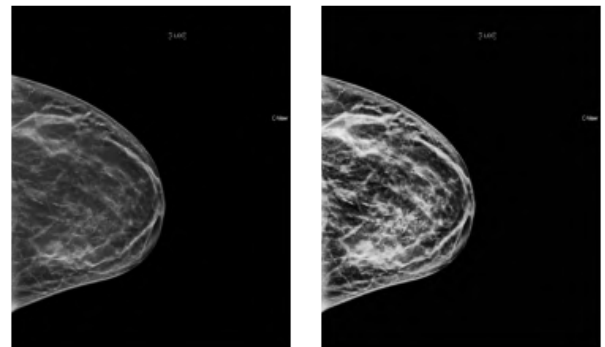


Fig. 3. Example of a mammogram image result of the breast-tissue using CLAHE algorithm: the original image and the image with contrast enhancement (from left to right).

• Step 6: Removing Pectoral Muscle

The step incorporates a series of algorithms to eliminate the pectoral muscle representation in mammogram images. Removing pectoral muscle is a crucial step in the image processing sequence, since it displays a similar intensity to anomalies. The result of this step is shown in Fig. 4 [3].

The algorithm applied for the removal of the pectoral muscle uses the Hough transformation in the following steps:

Step 1: Find the region of interest. Since our images are uniformly oriented, with the pectoral muscle consistently positioned in the upper left corner, we specifically target this area for analysis.

Step 2: Apply the median blur filter. Since the images may contain some noise, therefore we choose to use the median blur filter to reduce noise. We also tried out some other filters but found out the median blur filter best fit our expectation.

Step 3: Apply the Canny filter for contour detection, the objective of which is to identify the edges and contours. The Canny filter achieves this by looking for places where the brightness changes sharply to find edges. Then, it will isolate strong and relevant edges.

Step 4: Line detection using the Hough Transform. In this step, it utilizes the Hough Transform to detect lines in the image that has been processed by the Canny filter. It identifies potential lines, then calculates their positions and orientations.

Step 5: Find the shortlist relevant lines. This step involves filtering the lines detected in the previous step to identify the most relevant ones for our analysis. The function begins by setting angle and distance threshold. It selects those whose distance and angle align with our predefined threshold. This would narrow down lines to those that meet our specific geometric criteria.

Step 6: Apply a Hough mask to the region of interest. First, it takes the list from the previous step and sorting these lines based on their distance from the image origin. Then, it selects the line that is closest, identifying it as the pectoral line and used it to mask and remove the pectoral muscle from the image.

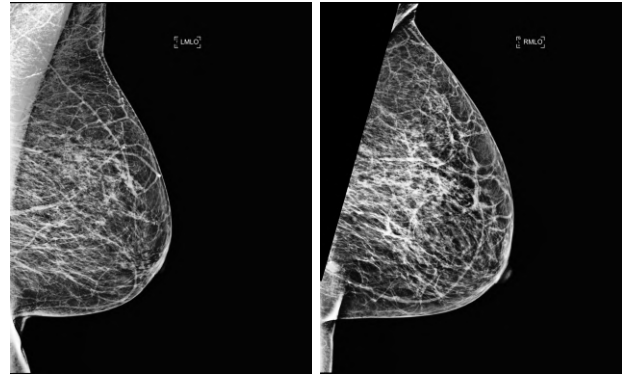


Fig. 4. the image after png preprocessing step 1 to 5 and the image removing pectoral muscle (from left to right).

As depicted in the Fig. 5, we input preprocessed images into YOLO to identify objects resembling a mass, along with associated confidence scores and coordinates. Subsequently, this information (coordinate, confidence scores) is transmitted to the CNN, which ultimately produces the final prediction based on the provided data.

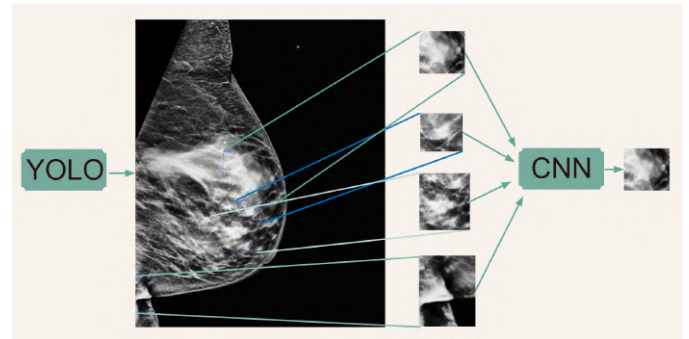


Fig. 5. the model structure

C. Model

1) *Innovative Model Structure*: Inspired by this paper [4], which utilized morphology method to propose candidate regions of masses, we introduced an innovative two-stage model structure, leveraging YOLOv7 as a stronger preliminary mass detector and employing a CNN model to validate the output generated by YOLO. This strategic combination aims to achieve superior performance compared to relying solely on YOLO. Moreover, we anticipate that leveraging the high accuracy in object detection achieved by YOLO will contribute to further improving the overall results.

2) YOLOv7:

• Purpose of YOLO

As mentioned earlier, we consider YOLO as a preliminary mass detector in this paper. Consequently, our focus is not on achieving high accuracy but rather on maximizing recall. As a result, we plan to adjust the confidence threshold after training to produce more objects resembling a mass in the output. This intentional adjustment is expected to lead to lower precision but higher recall.

• Training Preparation

Prior to initiating our training process, several preparatory steps are imperative for refining YOLO's labels and filtering out undesirable data. Initially, we will eliminate duplicate Regions of Interest (ROIs) by establishing a specific threshold for the Intersection Over Union (IOU) in the datasets, set up of the threshold will be discussed later in CNN's section. Subsequently, any ROIs that are entirely black will be excluded. Lastly, ROIs with an area exceeding 1000*1000 pixels will undergo removal as a third step. Implementing these three measures not only

enhances the performance of YOLO but also speed up the training process.

- **Train Validation Data Split**

After removal of undesirable data in the original datasets, we obtain 5618 Mammogram images in the end. We split 5000 images as training data and 618 images as validation data in YOLO.

- **Training Process**

The effectiveness of transfer learning in deep neural network training is demonstrated. We utilized a pre-trained weight provided by YOLOv7 to train our YOLO, employing 3 epochs as a warm-up phase and continuing training for a total of over 100 epochs. Despite the training duration, precision and recall stop improving around the 100th epoch, prompting us to early stop the training process.

To expedite the training process and reduce memory usage (that is, having larger batch size), we compressed training images to sizes of 640*640 and 1280*1280 pixels. However, not only the worse performance in precision and recall at the larger size, the training time for 1280*1280 images was also prolonged. Consequently, we decided to terminate the training process with the 1280*1280 size.

- **Preliminary Mass Detector**

YOLO will provide a confidence score for each identified object, such scores indicate that the model's confidence in its prediction that an object is present within that bounding box. Consequently, we can set up a certain threshold to filter out some wrongly predicted mass. A lower threshold will increase the number of identified objects output by YOLO, thereby improving recall but decreasing precision. Conversely, a higher threshold will yield fewer identified objects, leading to higher precision but lower recall. As a result, we conduct an experiment in order to fine-tune the confidence threshold and achieve better overall performance.

3) CNN:

- **Purpose of CNN**

As previously indicated, following the preliminary detection of masses by YOLO, a secondary-stage classification is performed using a Convolutional Neural Network (CNN). The objective is to enhance the precision of YOLO while concurrently preserving its recall, thereby improving the overall performance of the final system.

- **Data labeling**

The data generated by YOLO includes the coordinates and confidence scores for each image, representing the location of masses in the mammograms. In this section, we discuss the labeling process for each Region of Interest (ROI) image as either "mass" or "background." Initially, we compare the Intersection over Union (IOU) between the ground truth provided by the EMBED dataset and the ROI coordinates provided by YOLO in each mammogram. An observation reveals that the ROI areas

provided by EMBED are consistently larger than those identified by YOLO. Consequently, an appropriate IOU threshold is crucial to prevent mislabeling.

To determine the optimal threshold, we systematically analyze the ROI images within various IOU intervals. Fig. 6 illustrates successfully detected masses when the IOU is between 0.1 and 0.15, while Fig. 7 demonstrates instances where masses are not correctly cropped when the IOU is between 0.05 and 0.1. The upper images in the figures depict the ground truth, while the lower images represent the output generated by YOLO. Consequently, by observation from every intervals of IOU, we decide on an IOU threshold of 0.1. If the IOU between the ground truth and the data generated by YOLO exceeds 0.1, the data is labeled as "mass"; otherwise, it is labeled as "background."

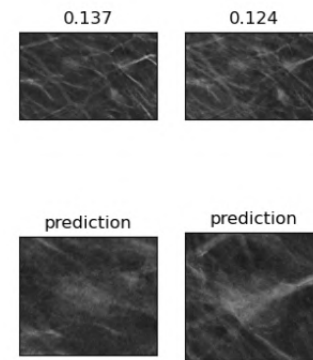


Fig. 6. IOU between 0.1-0.15 ROI images

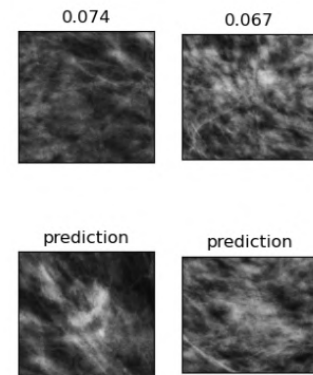


Fig. 7. IOU between 0.05-0.1 ROI images

- **Data splitting**

618 mammogram images from YOLO's validation data are utilized as testing data for the final evaluation of our overall performance. Moreover, 5000 mammogram images from YOLO's training data are partitioned, with 80 percent allocated for training and 20 percent for validation in the context of the CNN model. Finally, we identify 5326 detected ROI images in the training dataset,

comprising 2663 mass images and 2663 background images. Additionally, the validation dataset consists of 1332 detected ROI images, encompassing 666 mass images and 666 background images.

- **Five different methods applied in training CNN**

To improve the performance of the CNN model, we investigated five distinct approaches for classifying masses, outlined as follows. Additionally, the experiments were conducted using PyTorch version 2.1 and CUDA version 12.1 within the Anaconda environment with Jupyter Notebook. The models were trained on Nvidia RTX3060ti GPU.

- **Transfer learning on a single CNN model**

Initially, we employed a CNN model to classify masses detected by YOLO, specifically opting for the ConvNeXt-tiny and ResNeXt50_32x4d models from PyTorch. These models were initialized with pretrained weights "IMAGENET1K_V1." The selection of these models was driven by the advantageous utilization of transformers in mass detection, enabling the capture of long-range dependencies within medical images. Unlike traditional convolutional neural networks, transformers leverage self-attention mechanisms, facilitating the efficient consideration of global context information. This capability proves particularly beneficial in mass detection tasks, where the spatial relationships between masses and their surroundings play a crucial role. However, despite the anticipated advantages, our testing results did not meet the expectations, yielding only a 65 percent accuracy for both ConvNeXt-tiny and ResNeXt50_32x4d models.

- **Ensemble method with soft-voting**

During our investigation of mass pathology classification tasks in the literature, we observed that some researchers employed ensemble methods for mass classification. Given that our existing model is a two-stage architecture, involving substantial training and inference time, we opted for a simpler approach. Instead of training new neural networks for ensemble purposes, as demonstrated in [5], we opted to leverage the probabilities generated by two pre-trained models, ConvNeXt-tiny and ResNeXt50_32x4d, as discussed in previous approach. Our aim was to enhance accuracy through a straightforward averaging of probabilities. However, the achieved improvement in accuracy was merely 0.01-0.02 percent, falling short of our expectations.

- **Ensemble method with hard-voting**

As both previously mentioned models yielded suboptimal performance, even when employed in an ensemble approach, our focus shifted towards extracting more information from ROI images by leveraging the confidence scores provided by YOLO. To enhance performance, we refined the soft-voting strategy used earlier, incorporating the YOLO confidence scores.

Upon analyzing the training data, we observed that mass images received an average confidence score of 0.163, while background images only received 0.11 on average. Consequently, we opted to classify ROI images based on their YOLO confidence scores. Specifically, we designated ROI images with confidence scores greater than 0.15 as mass and those with scores equal to or below 0.15 as background. This information was then combined with the pre-trained ConvNeXt-tiny and ResNeXt50_32x4d models using a hard-voting approach. Contrary to our expectations, the results deteriorated, suggesting that this approach may not be effective for combining YOLO confidence scores in this context.

- **Fusion network for CNN and YOLO**

Since the YOLO threshold determined in the previous hard-voting approach was evidently suboptimal, while determining a fair and effective threshold proved challenging. Consequently, we pursued an alternative strategy by training a fusion network to learn the classification of masses using both the YOLO confidence score and the probabilities output by the pre-trained ConvNeXt-tiny and ResNeXt50_32x4d models. The rationale was that a machine learning model might better discern these features than manually selecting a YOLO threshold.

The fusion network comprised a fully connected layer with dimensions 5x32 and a ReLU activation function, followed by a 32x2 fully connected layer with a sigmoid activation function. The Adam optimizer with a learning rate of 0.001 was employed, and the loss function was set to crossentropy loss. During the training process, we observed that the fusion network achieved a maximum accuracy of only 67 percent. This outcome suggested that the YOLO confidence score did not provide significantly more information about mass images for CNN models. It implies that certain masses are inherently challenging to classify for both CNN models and YOLO, making the combination of YOLO confidence scores an ineffective method.

- **Concatenate CNN model with SVM**

Recognizing the limitations of the YOLO confidence score, we explored an alternative approach inspired by traditional R-CNN methods [6]. This involved utilizing a CNN model as a feature extractor and employing SVM for classification. Recognizing that relying solely on CNN-extracted features might be insufficient, we extended our methodology to include a radiomics approach. This involved extracting handcrafted features from ROI images, which were then combined with the CNN-derived features to enhance overall performance. For the handcrafted features, 83 features are extracted through a radiomics approach as Table I shows [7] [8]. Subsequently, feature selection, involving 4 steps in total. Step 1, features are scaled using Z-score for uniformity in magnitude. Step 2, is conducted via

ANOVA F-test, seeking the most significant features for the models. Step 3, a Pearson Redundancy-Based Filter (PRBF) is applied to mitigate multicollinearity by identifying and removing features that are highly correlated with one another beyond a certain threshold. Step 3, Backward Elimination using Ordinary Least Squares (OLS) regression for feature selection. It starts with all candidate variables and systematically removes the least significant variables until all variables in the model are statistically significant. Step 4, using Recursive Feature Elimination (RFE) with XGBoost as the base estimator to select the ten most important features from a set that has already been pruned by backward elimination. Subsequent to the identification of the ten most significant features, we visualized them using a plot diagram. As illustrated in Fig. 8, it was observed that the distributions of 'mass' and 'background' classes for four features are indicative of the similar distribution patterns found across all selected features. To further our analysis, we proceeded to plot the remaining features. However, this extended examination revealed that none of these additional features demonstrated noticeably distinct distribution patterns. The observed similarity in feature distribution for 'mass' and 'background' categories in our study, as opposed to the distinct distributions reported in other works, may be attributed to the differences in methodological application. The referenced papers utilized these methods for discerning benign from malignant masses, a task for which the features are presumably more discriminating. In contrast, our study aimed to differentiate between 'mass' and 'background' as determined by YOLO's outputs, which may inherently bear closer feature similarities.

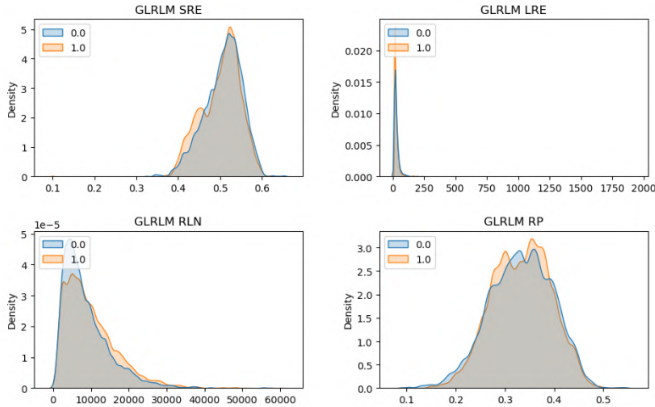


Fig. 8. distribution of four key features for 'Mass' (Class 1 in orange) and 'Background' (Class 0 in blue).

As for the feature extracted from pretrained ConvNeXtiny and ResNeXt50_32x4d models, concerning the features extracted from these two models, when combined, the total number of features amounted

TABLE I
STATISTICAL MEASURES OF RADIOMICS FEATURES

Feature categories	Feature names (optional)	Dimensions
First-order statistics	Energy, Entropy, ...	18
GLCM	Autocorrelation, Cluster Shade, ...	24
GLSZM	Gray Level Variance, ...	16
GLRLM	Gray Level Run Emphasis, ...	16
NGTDM	Coarseness, ...	5
GLDM	Dependence Entropy, ...	14

to 2816, we employed the feature selection method called VarianceThreshold from scikit-learn to eliminate features with low variance. This process resulted in a reduction from 2816 features to 526 features. However, despite this feature reduction, the accuracy achieved by SVM with linear kernel was only 64 percent, which was lower than using only a single CNN model to classify.

III. RESULTS

A. YOLO

In this section, we will showcase the outcomes of YOLO's predictions under various confidence score thresholds. Given that YOLO serves as our preliminary mass detector, it becomes imperative to fine-tune the threshold in order to optimize recall while maintaining an acceptable level of precision.

In Fig. 9, we note that under YOLO's default confidence threshold of 0.25, the model can detect some evident masses. However, non-round masses are occasionally overlooked. Therefore, we opted to further reduce the confidence threshold to 0.15. In Fig. 9, YOLO's predictions under the 0.15 threshold are depicted, revealing distinctions compared to the predictions under the 0.25 threshold. Unlike the 0.25 threshold, the 0.15 threshold results in the detection of more objects, thereby improving recall, albeit at the risk of misclassifying some tissues as masses. Subsequently, to capture more objects resembling masses, we also conducted tests with thresholds of 0.10 and 0.05, as illustrated in Fig. 10. Furthermore, Table II presents the precision, recall, and f1-score for 618 YOLO's validation images, evaluated by IOU at each confidence threshold.

TABLE II
PERFORMANCE OF DIFFERENT CONFIDENCE THRESHOLD IN PRELIMINARY MASS DETECTOR (YOLO)

Threshold	Precision	Recall	F1
0.25	0.615	0.194	0.295
0.15	0.442	0.381	0.410
0.10	0.333	0.512	0.403
0.05	0.204	0.710	0.316

B. CNN

To assess the accuracy of mass predictions, we compute the Intersection over Union (IOU) between Region of Interest

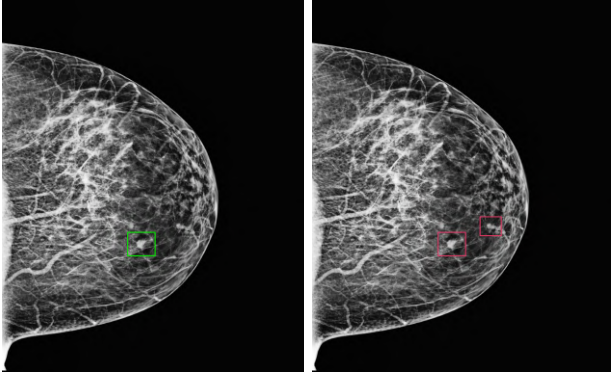


Fig. 9. Left: Threshold 0.25 image; Right: threshold 0.15

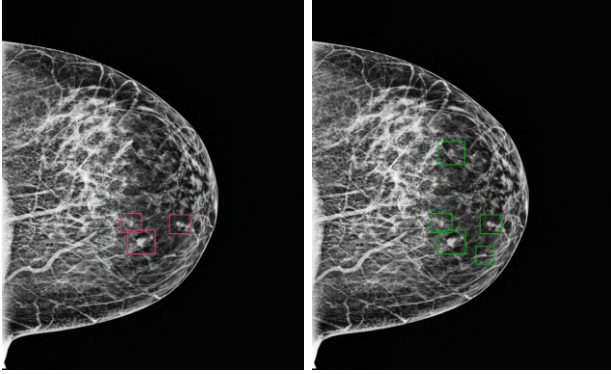


Fig. 10. Left: threshold 0.1; Right: threshold 0.05

(ROI) images and ground truth, as elaborated in Method of CNN during the data labeling process. Table III comprehensively presents the accuracy, precision, recall, and F1-score for the five distinct methods under the testing data. The fusion network is not included in the table, as the training accuracy remains at 67 percent, and we did not perform additional evaluation. The top two methods in the table indicate the use of the single CNN model, while the other methods remain consistent with the descriptions in Method of CNN.

TABLE III
PERFORMANCE OF DIFFERENT METHODS IN MASS CLASSIFICATION

Method	Accuracy	Precision	Recall	F1
ConvNeXt-tiny	0.659	0.666	0.643	0.654
ResNeXt50	0.652	0.651	0.655	0.653
Soft-voting	0.667	0.672	0.653	0.663
Hard-voting	0.654	0.694	0.551	0.614
CNN + SVM	0.648	0.648	0.649	0.648

C. YOLO+CNN

For the evaluation of our overall model performance, we introduced the ratio R as (1).

$$R = \frac{\text{increase in precision}}{\text{decrease in recall}} \quad (1)$$

This ratio is utilized to compare the original YOLO performance with the performance after concatenating it with

CNN models. We anticipate R value larger than one, indicating that the CNN model removes more background images than mass images. In Table IV below, different YOLO confidence scores are employed for the preliminary detection of masses, followed by classification with the pretrained ConvNeXt-tiny model. Since different methods shown in Table III have the similar results when concatenating with YOLO, so we only show the result of using convNeXt-tiny. The ratio R in the table is calculated by comparing the performance between Table II and Table IV. The observed trend reveals that R increases as the YOLO confidence score threshold becomes higher. This suggests that the CNN model performs better on images with higher YOLO confidence scores, indicating that the features learned by YOLO and the CNN model may be similar. Consequently, the YOLO confidence score is proportional to the CNN output logits. Additionally, the results from the fusion network described in Method of CNN also support this conclusion since combining the confidence score in the fusion network does not improve the model performance.

TABLE IV
PERFORMANCE FOR YOLO+CNN UNDER VARIOUS YOLO CONFIDENCE SCORE THRESHOLDS

Threshold	Precision	Recall	F1	R
0.05	0.334	0.498	0.399	0.614
0.1	0.480	0.376	0.422	1.084
0.15	0.596	0.304	0.403	1.986
0.2	0.698	0.225	0.340	3.614

IV. APPLICATION

A. Front end

we design an user interface via a webpage. In order to allow user upload the image without any transformation. Our website enable users to upload DICOM images directly. As the Fig. 11 show, the model will convert these image into PNG format and display them on the left side of the website after preprocessing. And then, our model will identify and mark the masses, displaying the results on the right side.

B. Back end

After receiving PNG images from the frontend, backend preprocessing is initiated to optimize the images for analysis. The processed results are first transmitted back to the frontend for visual representation. Subsequently, the preprocessed images undergo YOLO analysis using previously trained parameters to identify potential mass locations. The features identified as potential masses are then passed to a Convolutional Neural Network (CNN) for a final determination of their classification as actual masses.

In the concluding phase, the images with annotated features, along with information regarding the classification of each feature as a mass or non-mass, are communicated back to the frontend for visualization.

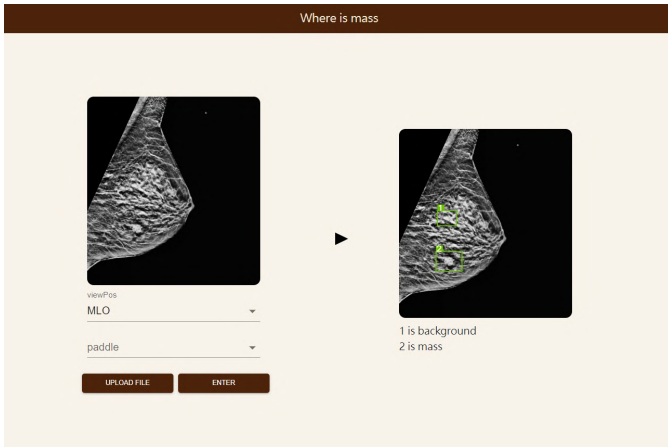


Fig. 11. the application structure

V. CONCLUSION

Firstly, YOLO's identification of the background, resembling the actual context, poses a challenge for our CNN model. In our system, we initially employ YOLO for object detection. When the background shares visual features with the objects, YOLO might mistakenly identify background regions as objects or extract features insufficient for accurate differentiation. Specifically, when YOLO identifies background regions resembling actual breast cancer lesions, such as certain tissue structures, shadows, or other visual features, it may incorrectly classify these areas as potential objects containing masses. This similarity can lead to inaccuracies in the generated labels or regions, subsequently impacting further analysis conducted by the CNN. For our CNN model, the presence of these similar regions makes it challenging to distinguish which features correspond to genuine breast cancer lesions, thereby reducing the model's ability to accurately detect masses. This predicament adversely affects the overall performance of the model, making it challenging to achieve satisfactory accuracy and recall. Secondly, in our attempt to explore various datasets, we encountered challenges as many datasets were not sufficiently comprehensive. Some did not provide ROI coordinates, while others had insufficient data volume. After finally identifying a dataset that better suited our requirements, we encountered issues during the practical implementation. We observed mislabeling errors in the Regions of Interest (ROIs), with many ROIs outlining completely black areas. These inaccuracies introduced confusion, adversely affecting the training process and yielding suboptimal model results. In future work, addressing the mentioned issues and optimizing our methodology could offer a pathway for improving the performance of the breast cancer mass detection model.

VI. DATA AND CODE AVAILABILITY

You can get more code information about our project with the following link.

<https://github.com/smilingweixiao/MLTeam28>

AUTHOR CONTRIBUTION STATEMENTS

- **Pinshun Wang:** Mass detection and classification related paper research, CNN model training and evaluation, Concatenate YOLO with CNN and evaluate the performance, Write CNN and result part in the report
- **Fenyu Hsieh:** Data preprocess method researching, Remove noise on images, Enhance the contrast of breast tissue, Remove pectoral muscle, Find other datasets and do some handle, Handcraft feature extraction and selection, Help to train CNN, Managed front-end and back-end integration of the website, Write dataset, preprocess introduction, step 3, 5, 6, and feature extraction, selection in report
- **Fangyu Hsu:** Data preprocess method researching, remove the artificial labels on the raw mammography. Find other datasets and do some handle. Design the UI of the website. Write the PNG preprocess step 4, Application Front end, model structure figure and do final check in report. Prepare and conclude the information for presentation. Presentation speaker. Prepare meeting minute.
- **Yahui Chang:** Classify the dataset into with mass and without mass. DICOM preprocessing. Fix the error in the metadata table. Extract the tissue in the paddle. Enhance ROI image. Maintain the API for preprocess Maintain website backend. Maintain README in repo. Write overview, DICOM preprocessing, PNG preprocessing step 1 and 2, backend in the report.
- **BingLun Li:** Mass detection and classification related paper research, YOLO model training and evaluation, prepare meeting agenda, write YOLO and result part in the report.
- **Iting Hsueh:** Mass detection and classification related paper research, YOLO model training, write abstract, introduction and conclusion in the report.

REFERENCES

- [1] A. T. T. R. R. M. G.-I. G. M. C. M. I. J. H. JiwoongJ.Jeong, BriannaL.Vey, "The emory breast imaging dataset (embed): A racially diverse, granular dataset of 3.4 million screening and diagnostic mammographic images," *Radiology: Artificial Intelligence*, 2023.
- [2] K. A. D.-V. N. K. M. G. M. A. N. Ruchaya, V. I. Kobera, "Segmentation of breast masses in digital mammography based on u-net deep convolutional neural networks," *Journal of Communications Technology and Electronics*, 2022.
- [3] T. J. O. V. D. T. K. Pascal Vagssa, Nafissatou Mallam Doudou, "Pectoral muscle deletion on a mammogram to aid in the early diagnosis of breast cancer," *International Journal of Engineering, Science and Technology*, 2020.
- [4] W. J. W. S. Z. Y. X. Y. Sun L, Sun H, "Breast mass detection in mammography based on image template matching and cnn," *Sensors*, 2021.

- [5] A. S. E. Asma Baccouche, Begonya Garcia-Zapirain, "An integrated framework for breast mass classification and diagnosis using stacked ensemble of residual neural networks," *Scientific reports*, 2022.
- [6] T. D. J. M. Ross Girshick, Jeff Donahue, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014.
- [7] Y. Q. B. Z. Meredith A. Jones, Rowzat Faiz, "Improving mammography lesion classification by optimal fusion of handcrafted and deep transfer learning features," *Physics in Medicine Biology*, 2022.
- [8] G. Z. J. H. Y. J. Y. C. Min Li, Liyu Zhu, "Predicting the pathological status of mammographic microcalcifications through a radiomics approach," *Intelligent Medicine*, 2021.

附錄3

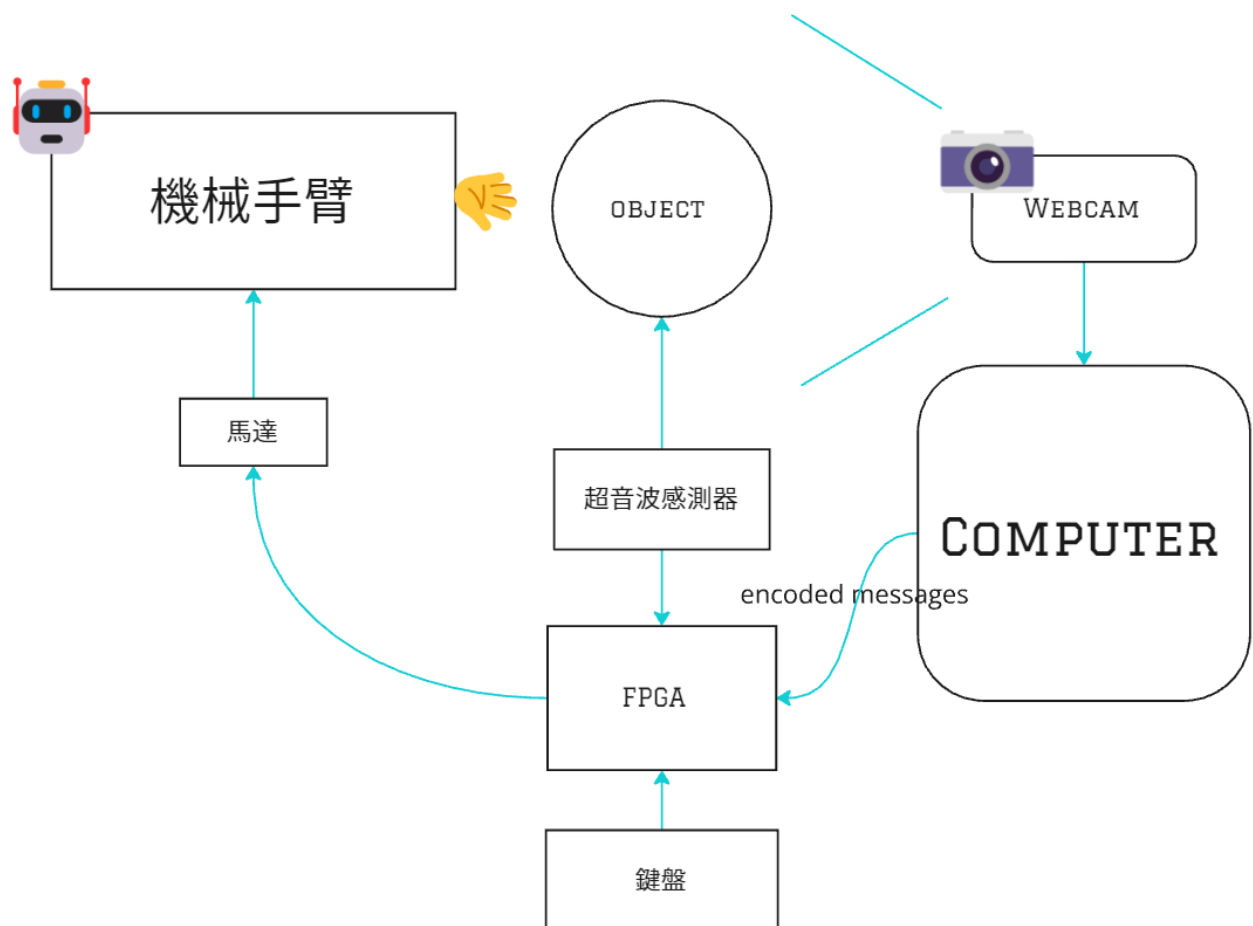
全自動液壓手臂顏色分類

Team No: 1	Team Name: 資電館的垃圾		
Project Title: 資電館的垃圾分類者			
Name: 李秉綸		ID: 110062240	
Name: 董柏宏		ID: 110062304	

一、設計概念與架構細節：

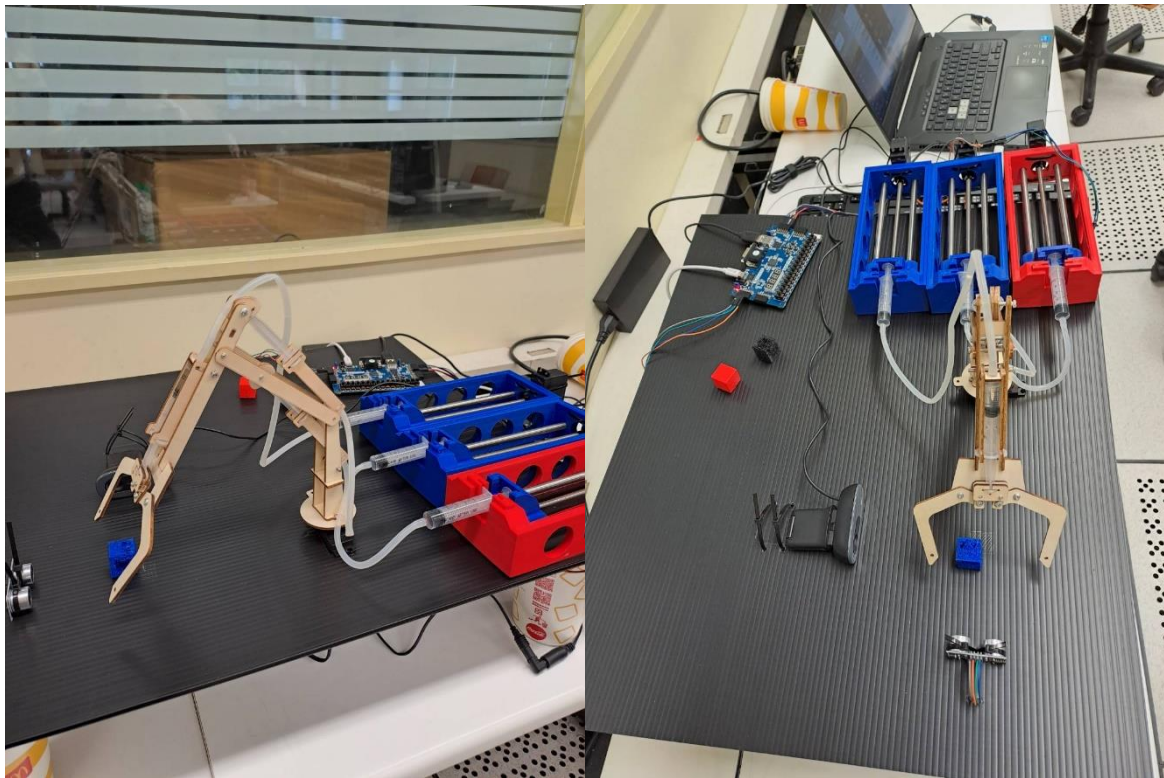
在此 project 中，我們期望可以透過電腦與 Basys 3 的溝通，讓 Basys 3 控制液壓機械手臂，將特定顏色方塊放到指定地點。

系統架構：



在我們的實作中可以分為硬體機構、軟體設計與 Verilog 相關設計，其實作細節如下：

硬體機構：

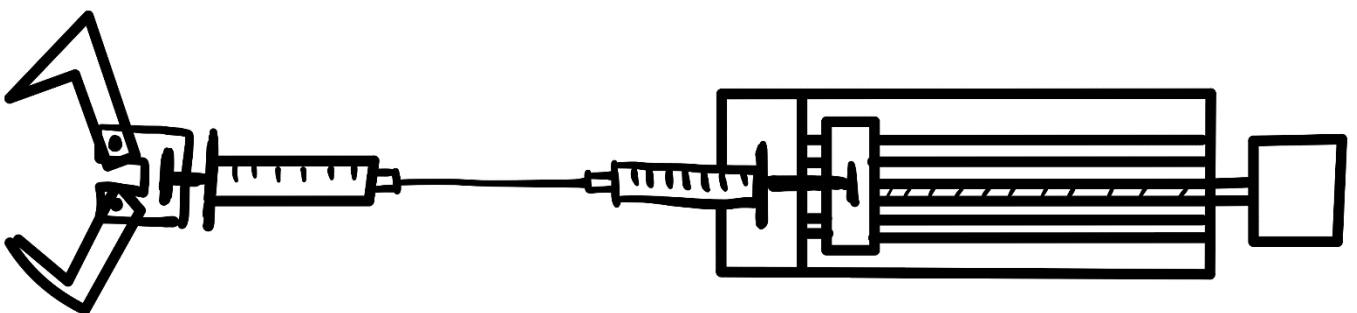


有關硬體機構設計，我們是自行設計與發想，將不同的部分組合而成我們的 design。

我們從網路上購買了液壓機械手臂，並且使用注射幫浦控制液壓，使得我們可以透過馬達控制機械手臂。並且因為是透過液壓，讓我們有度量衡（毫升）可以精準控制馬達的轉動，同時也避免馬達直接在關節處承受扭力以及硬體機構可能不夠堅固，無法承受馬達重量等種種問題。注射幫浦我們參考了此[連結](#)，有做些修改以符合我們的要求。

我們使用相機與超音波模組交錯來確認手臂前方塊的顏色，透過相機擷取圖片，超音波則確認手臂前方是否有物體，若顏色正確且超音波有感測，則將物體夾起放至指定地點。若沒有超音波，則可能相機判斷顏色正確手臂就開始動作，而前方卻沒有物體。

液壓控制關節概念圖如下：

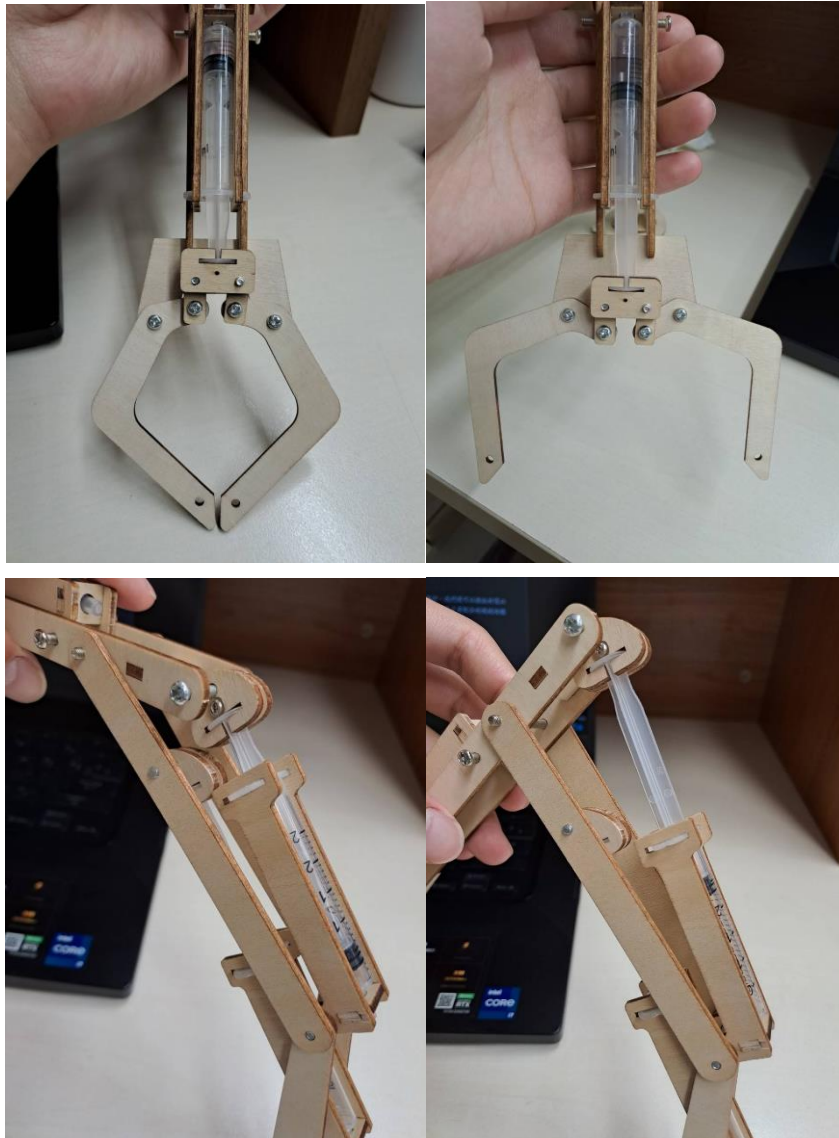


我們透過上圖中的右邊的注射幫浦推拉針筒，液體的流動使得在關節處的針筒伸縮，透過我們

的硬體結構設計，讓針筒可以帶動爪子跟手臂。

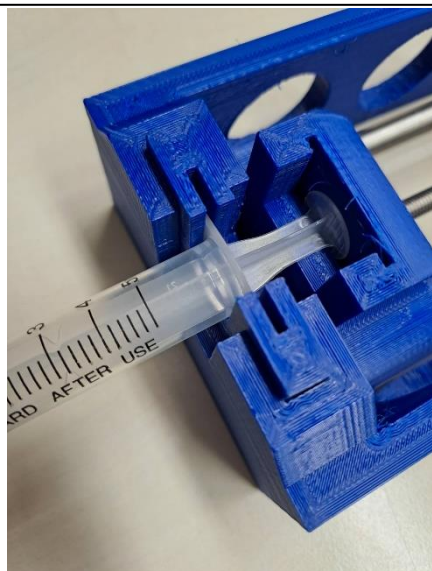
關節結構如下：

透過關節的機械架構，讓我們可以透過推拉針筒控制機械手臂移動。



注射幫浦結構如下：





我們在針筒 holder 中塞入一顆螺母，透過轉動馬達帶動連動的螺絲杆，螺絲杆的轉動將帶動螺母前進後退，我們便可以推拉針筒以控制液壓。

在注射幫浦中我們還有使用了 2 顆培林以及 2 根線性軸承，此 2 者配合的機械結構可以讓我們的針筒推拉的更穩定，讓我們可以更精準的控制機械手臂。

軟體機構：

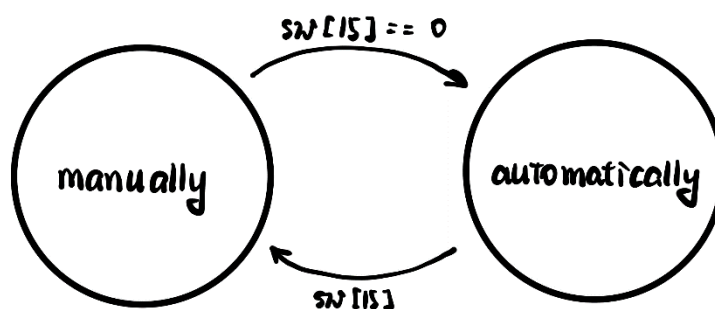
關於軟體機構，我們使用 python 中的 serial 模組，開啟 serial port 單向傳輸訊息給 Basys 3，我們將判斷的顏色 encode 成一個 byte，encode 對照如下：

Red	8'h41
Black	8'h42
Blue	8'h43
Other colors	8'h00

我們透過 python 的 cv2 模組開啟 webcam，取得 webcam 的照片後分成每一幀送進顏色判斷的 machine learning 模型中。

顏色判斷我們是在電腦中 run 一個 KNN model，判斷當前照片中最主要的顏色並回傳。此部分我們是參考 [Github repo](#) 並做些許修改以符合我們的需求。原先的 model 是每一幀都判斷，可能導致顏色與光線還來不及穩定而判斷失準，在我們的修改過後會等待光線、顏色穩定後大約 3~4 秒再判斷顏色並回傳。

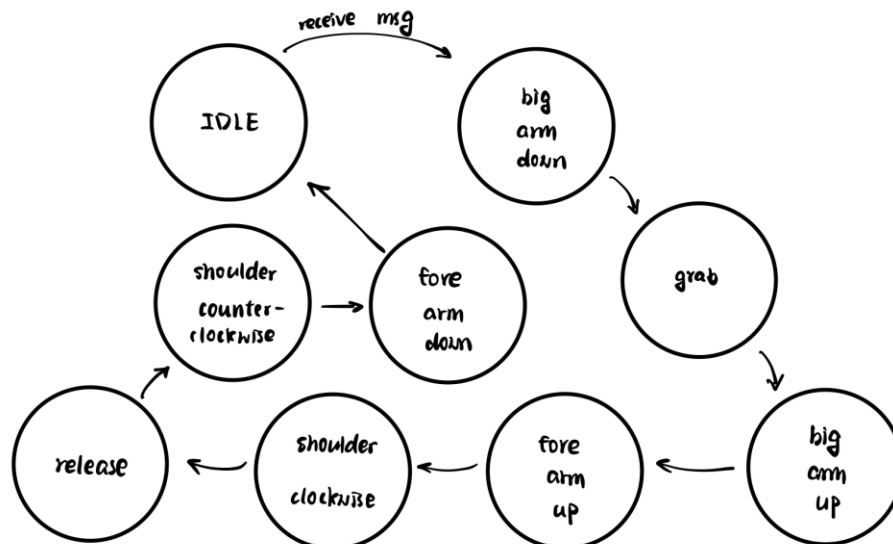
Verilog 相關設計：



在我們的 design 中機械手臂的控制有分 2 個 state，一個是手動控制，另一個則是自動控制。當 sw[15] 為 off 時，切換到自動控制，而 sw[15] 為 on 時則切換到手動控制。

手動控制透過判斷指定按鍵是否按下控制馬達轉動，我們將 1, 2 按鍵 map 到肩膀、q, w 按鍵 map 到爪子、a, s 按鍵 map 到小臂、z, x 按鍵 map 到大臂，且在我們的設計中可以一次轉動多個馬達。

自動控制邏輯如下：



我們依據調好的 counter 參數來決定馬達要動多久，如下圖：

```

RED: begin
  if (counter_count < 1700) begin
    next_mode3 = 2'b10;
  end else if (counter_count < 15'd4800) begin
    next_mode1 = 2'b10;
  end else if (counter_count < 15'd6500) begin
    next_mode3 = 2'b01;
  end else if (counter_count < 15'd10000) begin
    next_mode2 = 2'b10;
  end else if (counter_count < 15'd10100) begin
    next_mode4 = 2'b01;
  end else if (counter_count < 15'd13200) begin
    next_mode1 = 2'b01;
  end else if (counter_count < 15'd13290) begin
    next_mode4 = 2'b10;
  end else if (counter_count < 15'd16790) begin
    next_mode2 = 2'b01;
  end else begin
    // reset counter
    next_counter1 = 0;
    next_counter_count = 0;
    // reset state
    next_state = NONE;
  end
end
  
```

在我們的設計中機械手臂在放置方塊至指定位置後會回到初始位置。

伺服馬達控制機制：

我們使用的是 360 連續旋轉 Mg996r 伺服馬達，透過傳送 refresh rate 為 50Hz、duty cycle 1%~2% 的 PWM 訊號我們可以控制馬達的轉速及方向，然而在我們的 design 中馬達轉速皆固定，為的是讓機械手臂速度可以維持在最快。

Uart 傳遞機制：

我們所使用的是非同步單線傳輸，當沒有資料要傳輸時，TX（傳遞訊息的腳位）維持 high，

當有資料要傳輸時 TX 改為 low，讓接收端知道準備要接受訊息了，接下來固定傳送 8 個 bit 的 data，8 個 bit 傳遞結束後，將 TX 拉回 high，使接收端知道訊息傳遞完畢，這部分有參考[網路上的資料](#)，並做修改來符合我們的需求。

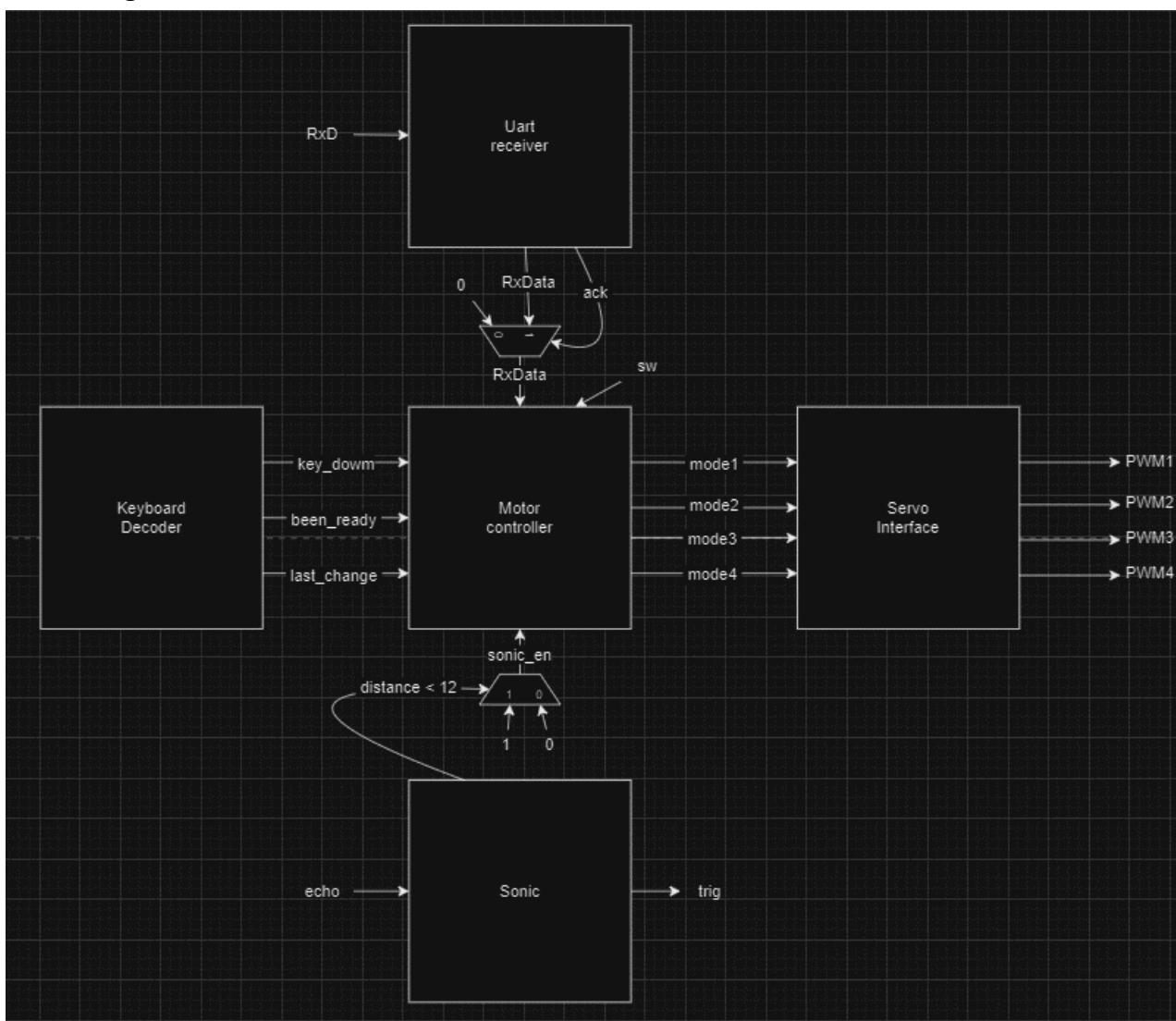
超音波控制：

我們使用 Lab8 中提供的 module，但我們有透過移除除法器將此 module 優化如下圖，讓超音波模組可以使用更少合成資源。

```
assign distance_count = distance_register >> 6;
```

雖然距離變成不是以公分為單位，但我們可以透過調較參數來選擇我們要讓超音波感測的距離，因此做到使用更少資源擁有相同的感測效果。

二、Block Diagram：



在上圖中可以看到我們將所有訊號線送至 Motor controller 模組內，包含代表顏色的 RxData、鍵盤的輸入、超音波是否感測到物體等等的訊號線，利用這些訊號決定馬達的運作方式，output mode1, mode2, mode3, mode4 至 Servo Interface 內（mode 代表的是不動或順時針或逆時針），透過 mode 決定 PWM 訊號。

三、難易度說明：

在我們的 design 中，我們覺得最困難的是硬體層面，初期在控制不熟悉的伺服馬達時遇到非常多的問題，供電、coding 等等，甚至可能是馬達壞掉，不知道問題出在哪的情況非常難 debug。

再來是我們硬體架構組裝困難，我們一開始就決定要使用較強的硬體結構以防萬一，所以我們選擇使用 3D 列印，在學習 3D 列印的過程中也遇到很多不同的問題，花了很多時間才成功列印 3 個注射幫浦。

取得注射幫浦後，我們開始機械零件購買，我們跑了很多間五金行才找到有賣我們需要的機械材料，好不容易等五金行調貨送達之後，我們切割材料（線性軸承、螺絲杆），結果發現螺絲杆沒辦法固定在伺服馬達上，這樣就代表注射幫浦無法成功，因為馬達無法帶動幫浦。我們花了好幾天的時間在想解決辦法，最後是透過一個非常緊的塑膠套，一端套馬達，一端套螺絲杆，讓馬達可以帶動螺絲杆。經過了這麼多困難我們才組裝完注射幫浦。

在最後硬體的組合中還有遇到許許多多的小問題，舉例而言，針筒中的氣泡可能導致手臂動作不準確，我們也費了好一番功夫才將氣泡去除，還有許多其他問題，此處就不再贅述。

因此，綜上，我們覺得我們的 design 難度很高，然而因為有許多貴人的幫助，所以難度有下降許多，我們才得以完成這樣的 Final project。

四、分工：

董柏宏：硬體組裝、3D 列印、零件採買。

李秉綸：硬體組裝、coding。

五、測試完整度：

我們覺得非常好，跟我們預期完全相同，手臂可以自動精準夾起顏色方塊並將方塊放到指定位置，手動控制機械手臂也非常精準。

六、困難與解決方法：

馬達無法順時針轉動：

在剛開始的時候馬達只能逆時針，那時以為是馬達的問題，因為我的 code 已經檢查了非常多次，不應該有問題，然而到實體店確認過後發現 3、4 個馬達我的 code 都還是無法順時針轉動，思考良久以及上網搜尋過後，我們得出的結論是可能是供電的問題，而問題就在我們使用外接電源的當下解決了。

硬體機構組裝：

舉例而言，像是馬達無法固定在注射幫浦上，我們遇到很多這類型的硬體組裝問題，這種問題通常需要客製化的解法，因此需要一點天馬行空的想像力。馬達這個問題我們的解法是在注射幫浦上鑽洞，用 2 條束帶將馬達鎖上。