

# 성균관대학교 - 소이넷

- 딥러닝 모델의 SoyNet 포팅 -

2023. 8. 3

성균관대학교 소프트웨어융합대학

발표자: 백승렬 (소프트웨어학과, 2학년)

# Contents

- 팀 소개
- 과제 목표
- 과제 수행 과정
- 과제 결과물
- 2학기 개발 계획
- 과제 성과 및 의미
- 하계집중근무 수행 소감
- Appendix
  - 2023년 하계집중근무 과제 요약

# 팀 소개



김호재



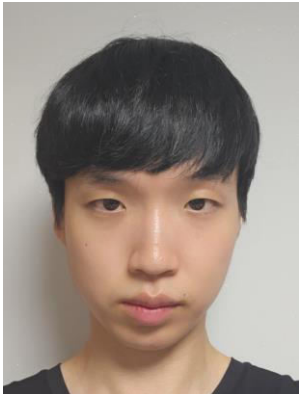
백승렬



이미향 교수님



유경석 소이넷 팀장님



박세훈



박제현



이해성



김용호 소이넷 대표님

# 과제 목표 (선정 배경)

## □ 선정 배경

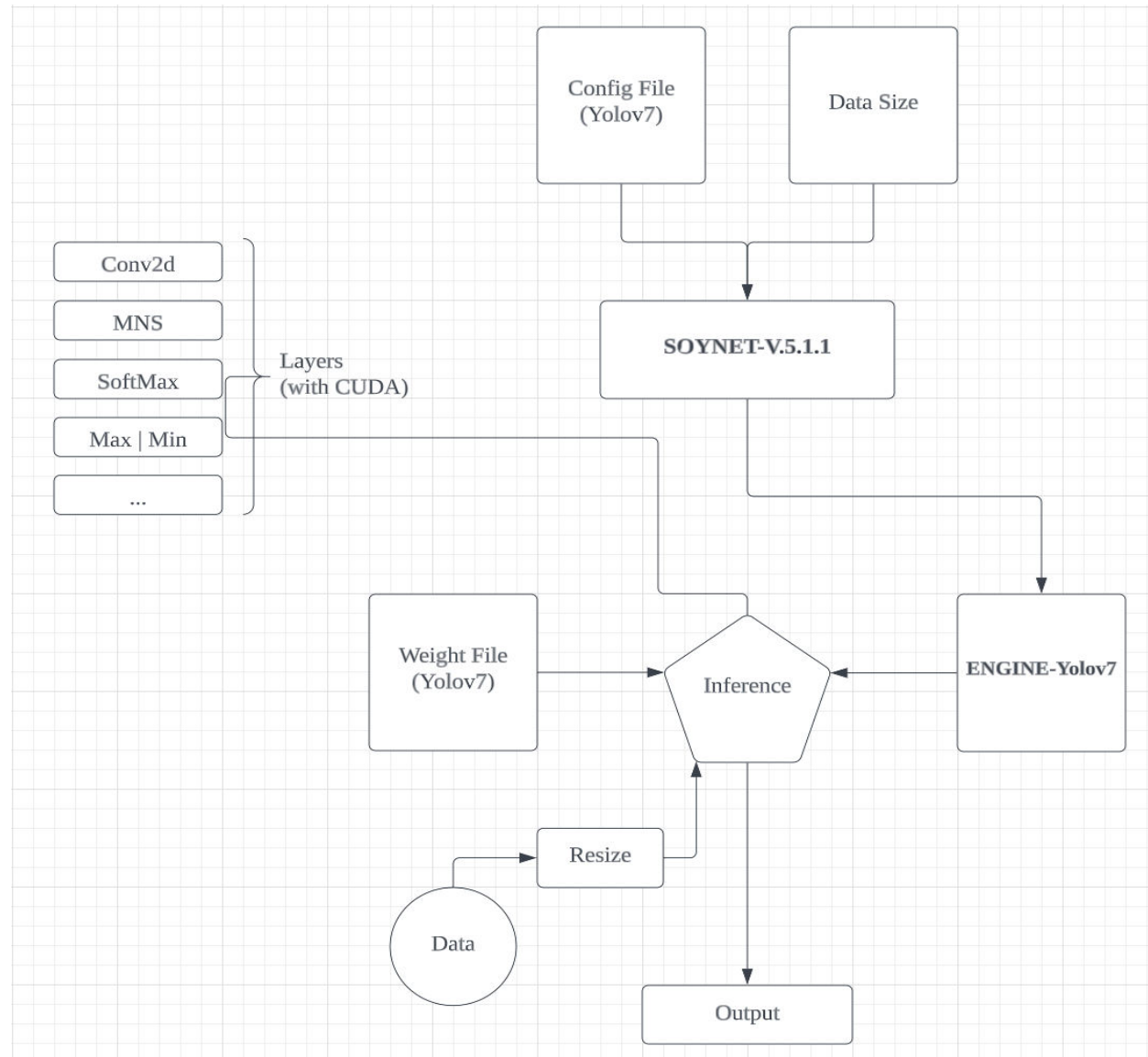
- 소이넷이 보유하고 있는 솔루션은 config파일을 읽고 그 문법에 따라 .weights파일에서 순서대로 가중치 값을 불러와 자동으로 모델의 추론작업을 구현한 엔진을 만들어 줄 수 있다.
- 이러한 솔루션을 활용해 비교적 쉽게 cpp과 cuda를 이용한 효율적인 추론 프로그램을 만들어 줄 수 있다.
- 따라서 이번 과제에서는 이 소이넷 솔루션을 활용하여 여러 딥러닝 모델에 대하여 원본 딥러닝 모델과 동일한 작업을 하면서도 더 빠르고 효율적인 프로그램을 구성하고자 한다.

# 과제 목표 (목표)

## □ 목표

- 여러 딥러닝 모델들 분석(소이넷 솔루션이 효과적인지 아닌지 판단)
- 소이넷 솔루션을 활용하여 **빠르고 효율적인 딥러닝 추론 프로그램 구성**
- **성능비교(결과물 vs. 원본 추론 모델)**
- 전처리와 후처리 부분의 업데이트를 통해 실제 고객사의 요구 충족

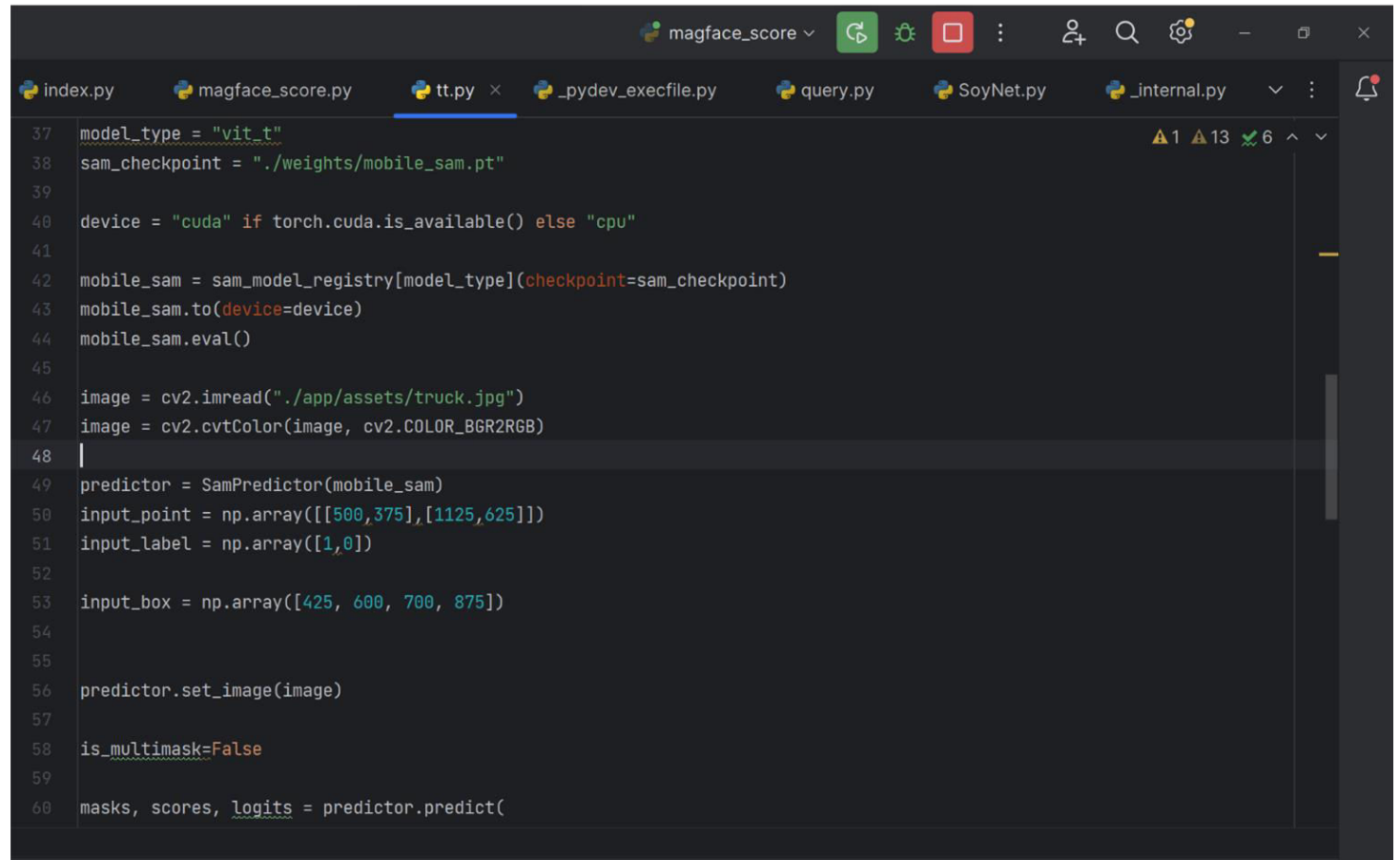
# 과제 수행 과정(구현 방법)



# 과제 수행 과정(구현 방법)

## □ 실행 파일 만들기

- 원본 모델의 추론과정만을 실행하는 파이썬 프로그램.
- 원본 모델의 github에 있으면 사용 없으면 새로 생성.
- 이 과정에서 모델 내부 코드에 오류가 있다면 디버깅함.



```
magface_score v magface_score.py tt.py x _pydev_execfile.py query.py SoyNet.py _internal.py
37 model_type = "vit_t"
38 sam_checkpoint = "./weights/mobile_sam.pt"
39
40 device = "cuda" if torch.cuda.is_available() else "cpu"
41
42 mobile_sam = sam_model_registry[model_type](checkpoint=sam_checkpoint)
43 mobile_sam.to(device=device)
44 mobile_sam.eval()
45
46 image = cv2.imread("./app/assets/truck.jpg")
47 image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
48
49 predictor = SamPredictor(mobile_sam)
50 input_point = np.array([[500,375],[1125,625]])
51 input_label = np.array([1,0])
52
53 input_box = np.array([425, 600, 700, 875])
54
55
56 predictor.set_image(image)
57
58 is_multimask=False
59
60 masks, scores, logits = predictor.predict(
```

# 과제 수행 과정(구현 방법)

## □ Config 만들기

- 오른쪽의 예시와 같이 소이넷 솔루션 프로그램이 만들 엔진의 레이어와 그에 대한 인자의 크기와 참조에 관한 정보를 적는 파일
- 제공된 솔루션에 있는 레이어만 파싱하여 읽을 수 있으며 추가한 레이어는 layer\_dict파일에 적어야 파싱할 수 있음

```
[reshape] shape=$BATCH_SIZE,19,7,19,7,128
[trans] order=0,1,3,2,4,5
[reshape] shape=$BATCH_SIZE*361,49,128
[norm] mode=layer axis=2 weight_order=rb eps=1e-5

[dense] hidden=384 weight_order=wa refname=QKV_0
[reshape] shape=$BATCH_SIZE*361,49,4,96
[chunk] axis=3 count=3 refname=Q_IN,K_IN,V_IN
[trans] input=Q_IN order=0,2,1,3 refname=Q
[trans] input=K_IN order=0,2,1,3 refname=K
[trans] input=V_IN order=0,2,1,3 refname=V

#[trans] input=K order=0,1,3,2 refname=K_T
[matmul] input=Q,K trans_b=1 refname=QK_IN
[scale] input=QK_IN scale=0.176777 refname=QK
[eltwise] mode=mobile-sam input=QK
[softmax] refname=ATTN_0
[matmul] input=ATTN_0,V refname=ATTN_0V
[trans] input=ATTN_0V order=0,2,1,3 refname=ATTN_0T

[reshape] input=ATTN_0T shape=$BATCH_SIZE*361,49,128
[dense] hidden=128 weight_order=wa
[reshape] shape=$BATCH_SIZE,19,19,7,7,128
[trans] order=0,1,3,2,4,5
[reshape] shape=$BATCH_SIZE,133,133,128
[slice] start=0,0,0,0 shape=$BATCH_SIZE,128,128,128
[reshape] shape=$BATCH_SIZE,-1,128 refname=X_0_1
[eltwise] input=BLK2_IN,X_0_1 mode=add
```



# 과제 수행 과정(구현 방법)

## □ Layer\_dict

- 소이넷 솔루션이 파싱하는 config파일의 문법을 정리한 곳
- cpp, cuda로 직접 구현한 레이어를 이곳에 작성하여 파싱할 수 있도록 함.

output	refname	vs			1						
base	version	s			0						
base	act	s	identity	linear,iden	0	??????	??????	??????, otherwise	custom layer??	????????	??
base	act_alpha	f32			0						
base	act_beta	f32			0						
base	prec_mod	s	f32	i8,u8,f16,f:	0	model??	??????	????	default??	????.	
base	input	vs	*		0						
base	name	s			0	??????	??????	??????, otherwise	custom layer??	????????	??
base	refname	vs			0						
bbox_regr	mode	s		glip,retina	0						
bbox_regr	anchor	vf32			0						
bbox_regr	stride	i32			0						
activation	backward	i32	0		0	backpropagation	????				
einreshape	mode	s	palm	generic,pa	0						
softmax	mode	s	generic	generic,alc	0						
softmax	axis	i32	-1		0						
attn_mask	mode	s		glip	0						
attn_mask	window	i32			0						
attn_mask	shift	i32			0						
einmask	mode	s	palm	generic,pa	0						
rotemb	mode	s	palm	generic,pa	0						
nms_merge	mode	s		yolov8,yol	1						
nms_merge	result_cou	i32	50		0						
nms_merge	org_size	vi32			0						
embed	mode	s		vit_b32	0						
embed	table_shape	vi32			0	??????	??????	????, fastspeech????	reps??	ENC??	??????
posemb	mode	s		fastspeech	1						

# 과제 수행 과정(구현 방법)

## □ Weights 파일 만들기

- 소이넷 솔루션이 config파일을 읽으며 지정된 weight 파일에서 자동으로 필요한 가중치의 크기만큼 데이터를 읽어옴.
- 모델의 가중치를 읽어와 특정 순서대로 weights 파일에 작성하는 파이썬 프로그램을 따로 작성하여 생성시킴.

> 내 PC > DEV (E:) > DEV-5.1.0 > mgmt > weights				
이름	수정한 날짜	유형	크기	
deep-lab-v3.weights	2023-06-27 오전 10:09	WEIGHTS 파일	164,302KB	
FastSAM.weights	2023-06-29 오후 3:13	WEIGHTS 파일	0KB	
mobile_sam.weights	2023-07-18 오후 2:33	WEIGHTS 파일	42,432KB	
mobile_sam_backward_point.weights	2023-07-18 오후 3:58	WEIGHTS 파일	3KB	
mobile_sam_box.weights	2023-07-19 오전 9:59	WEIGHTS 파일	4KB	
mobile_sam_decoder.weights	2023-07-19 오전 11:24	WEIGHTS 파일	19,949KB	
mobile_sam_forward_point.weights	2023-07-18 오후 3:00	WEIGHTS 파일	3KB	
mobile_sam_img_encoder.weights	2023-07-18 오후 2:35	WEIGHTS 파일	22,476KB	
mobile_sam_mask.weights	2023-07-19 오후 2:42	WEIGHTS 파일	19KB	
mobile_sam_no_box.weights	2023-07-19 오전 10:28	WEIGHTS 파일	2KB	
mobile_sam_no_mask_dense.weights	2023-07-19 오전 9:06	WEIGHTS 파일	2KB	
yolov5m6r62.weights	2023-06-26 오전 9:40	WEIGHTS 파일	0KB	

# 과제 수행 과정(구현 방법)

## □ cpp프로그램 만들기

- 지금까지 만든 것들을 가지고 실제로 엔진을 만들고 추론 작업을 수행하도록 메인 프로그램을 cpp로 구성
- 여기서 디버깅을 하면서 지나가는 데이터를 분석하여 프로그램이 의도적으로 돌고 있는지 확인할 수 있다.
- 작업이 끝나고 이 프로그램과 같은 역할을 하는 프로그램을 파이썬으로 만든다.

```
256 feedData(img_encoder_handle, 0, source.data());
257 inference(img_encoder_handle);
258 getOutput(img_encoder_handle, 0, image_embed.data());
259
260 if (forward_point_encode_max != 0) {
261     feedData(forward_point_prompt_handle, 0, point_coord1.data());
262     inference(forward_point_prompt_handle);
263     getOutput(forward_point_prompt_handle, 0, forward_point_embed.data());
264 }
265
266 if (backward_point_encode_max != 0) {
267     feedData(backward_point_prompt_handle, 0, point_coord2.data());
268     inference(backward_point_prompt_handle);
269     getOutput(backward_point_prompt_handle, 0, backward_point_embed.data());
270 }
271
272 feedData(box_prompt_handle, 0, box_pt.data());
273 inference(box_prompt_handle);
274 getOutput(box_prompt_handle, 0, box_embed.data());
275
276
277 std::ifstream ifs("../mgmt/weights/mobile_sam_no_box.weights", std::ios::binary);
278 ifs.seekg(10 * sizeof(float), std::ios::beg);
279 ifs.read((char*)no_box_embed.data(), no_box_embed.size() * sizeof(float));
280
281
282 if (is_mask) {
283     feedData(mask_encoder_handle, 0, logits_single.data());
284     inference(mask_encoder_handle);
285     getOutput(mask_encoder_handle, 0, dense_embed.data());
286 }
287 else {
288     feedData(no_mask_dense_handle, 0, nothing.data());
289     inference(no_mask_dense_handle);
290     getOutput(no_mask_dense_handle, 0, dense_embed.data());
291 }
292
293 for (int i = 0; i < batch_size; i++) {
294     memcpy(sparse_embed.data() + i * sparse_embed_num * 256, forward_point_embed.c
```

# 과제 수행 과정(구현 방법)

## □ 확인 작업

- 원본 모델과 소이넷 솔루션을  
이용해 만든 모델의 오차값을 확인

데이터 값의 오차가 작다면 괜찮고  
아니면 다시 디버깅 하면서 수정  
해야 함.

```
(tt) E:\DEV-5.1.0\TEMP>python tt1.py
0 R0= -0.0654622 R= -0.0654620 diff=0.0000001
2 R0= -0.0457002 R= -0.0456995 diff=0.0000006
15 R0= -0.0132941 R= -0.0132948 diff=0.0000007
62 R0= -0.1333544 R= -0.1333551 diff=0.0000007
94 R0= -0.1315866 R= -0.1315859 diff=0.0000008
200 R0= -0.0781807 R= -0.0781815 diff=0.0000008
293 R0= -0.0812233 R= -0.0812245 diff=0.0000012
12833 R0= -0.0209088 R= -0.0209100 diff=0.0000013
13424 R0= -0.1146743 R= -0.1146757 diff=0.0000014
14145 R0= -0.1654208 R= -0.1654224 diff=0.0000016
77880 R0= 0.0010930 R= 0.0010947 diff=0.0000017
128750 R0= 0.0788076 R= 0.0788059 diff=0.0000017
128751 R0= 0.0874563 R= 0.0874541 diff=0.0000022
729263 R0= -0.1257302 R= -0.1257279 diff=0.0000023
729499 R0= -0.1379474 R= -0.1379445 diff=0.0000030
731035 R0= -0.1502266 R= -0.1502296 diff=0.0000030
731151 R0= -0.0241387 R= -0.0241350 diff=0.0000037
cnt_total=4194304 cnt_diff=4130780 max_idx=731151
```



# 과제 수행 과정(구현 방법)

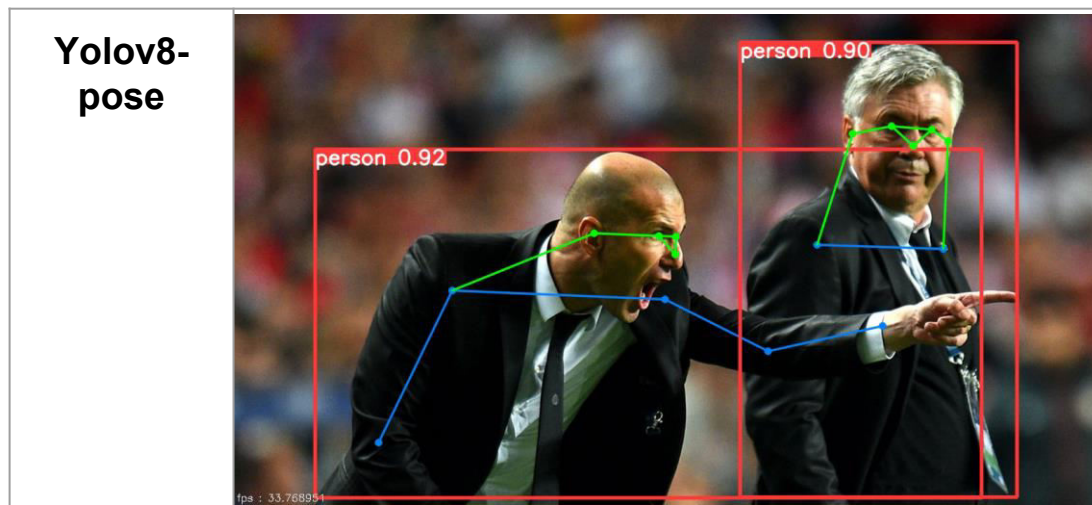
## □ Benchmark 작성

- 완성된 작업물에 대해 작업물의 성과를 기록한 Benchmark를 작성

9	pretrained	yes				
10	porting 시작	#####				
11	porting 종료	#####				
12	porting 작업자	백승렬				
13	train-code name					
14	inference-code name	tt.py(batch_size=1), tt1.py(batch_size=2,4)				
15	weight-down-code name	ww.py				
16	work 코드					
17	Benchmark 기기	RTX 3080 12GB				
18	data	truck.jpg				
19	batch	1	2	4		
20	precision	f32	f32	f32		
21	data size or length	batch_size x input_height x input_length x 3				
22	SoyNet fps(입력 전부 사용)	34.49	27.46	17.08		
23	SoyNet GPU memory(입력 전부 사용)	865	1109	1357		
24	pytorch fps(입력 전부 사용)	34.55	35.08	35.13		
25	pytorch memory(입력 전부 사용)	1697	1761	1765		
26	Benchmark brief	입력 이미지는 전부 resize되어 들어감. 각 배치별로 다				
27	Model Consistency	pytorch	SoyNet	Diff		
28	(정합성 비교)	각 좌표마다	각 좌표마다	0		

# 과제 결과물

- Yolov8-pose (이해성 - RTX2080\_Ti\_10GB - 1X1280X720X3)
  - Fps
    - 0.29 => 25.26
  - Memory Consume (MB)
    - 1552 => 619



# 과제 결과물

- Yolov8-face (이해성 - RTX2080\_Ti\_10GB - 1X1280X720X3)

- Fps

- 21.84 => 72.17

- Memory Consume (MB)

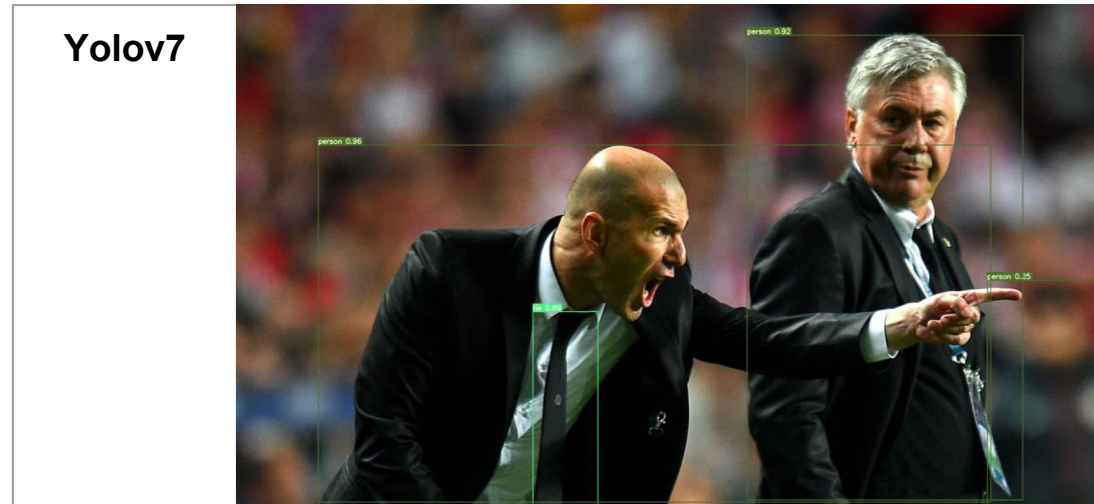
- 1254 => 279

Yolov8-  
face



# 과제 결과물

- Yolov7 (박세훈 - RTX3080\_Ti\_10GB - 1X1280X720X3)
  - Fps
    - 21.84 => 78.31
  - Memory Consume (MB)
    - 1626 => 592





# 과제 결과물

- Yolov6-face-S (박세훈 - RTX3080\_Ti\_10GB - 1X1280X720X3)
  - Fps
    - 29.7 => 96.58
  - Memory Consume (MB)
    - 1440 => 406



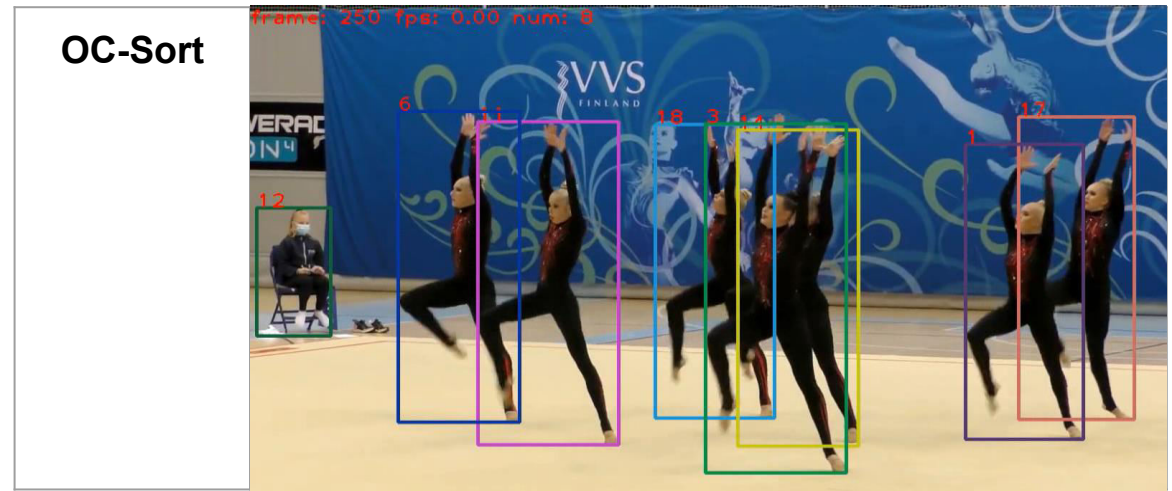
# 과제 결과물

- ByteTrack (박세훈 - RTX3080\_Ti\_10GB - NX1280X720X3)
  - Detection(Yolox) + Tracking Algorithm (C++ 구현)
- Fps
  - 7.82 => 21.39
- Memory Consume (MB)
  - 2458 => 1136



# 과제 결과물

- OC-Sort (박세훈 - RTX3080\_Ti\_10GB - 1X1280X720X3)
  - Detection(Yolox) + Tracking Algorithm (C++ 구현)
- Fps
  - 7.62 => 21.59
- Memory Consume (MB)
  - 2088 => 1140



# 과제 결과물

- Yolov8-track (김호재 - RTX3080\_Ti\_10GB - 1X1280X720X3)

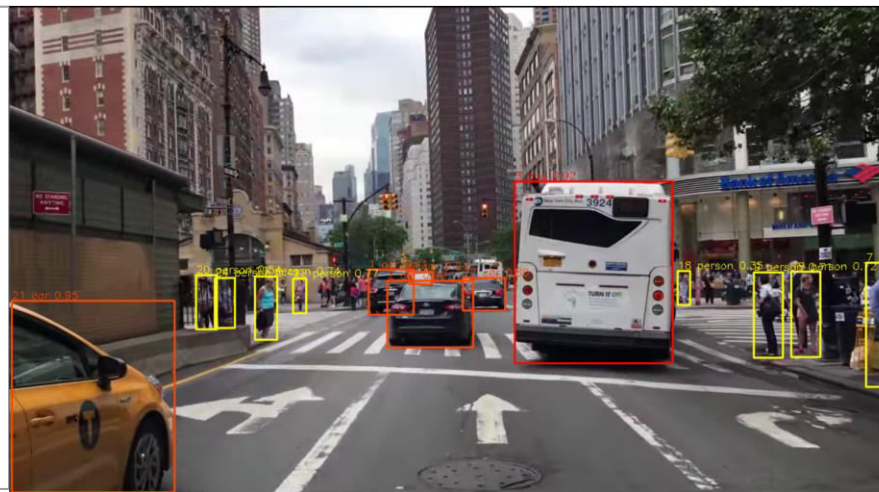
- Fps

- 10.0 => 69.32

- Memory Consume (MB)

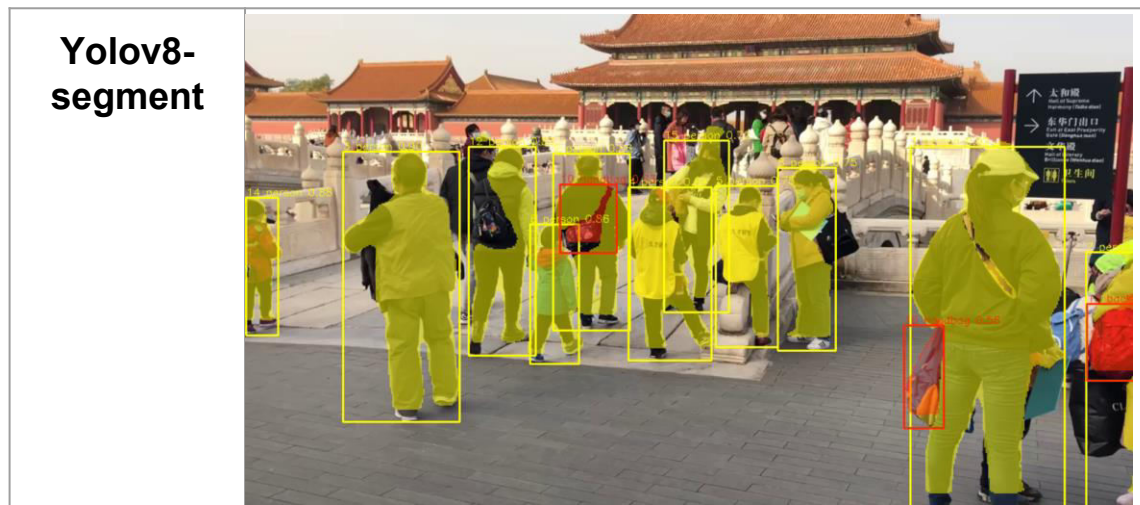
- 8061 => 3853

Yolov8-track



# 과제 결과물

- Yolov8-track & segment (김호재 - RTX3080\_Ti\_10GB - 1X1280X720X3)
  - Fps
    - 10.0 => 15.04
  - Memory Consume (MB)
    - 8061=> 3853



# 과제 결과물




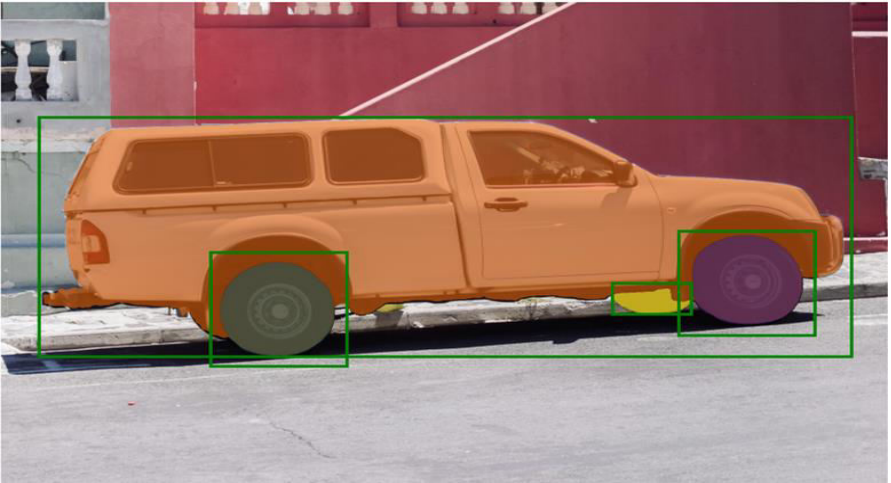
- **Mobile-SAM(백승렬 - RTX3080\_Ti\_10GB - 입력크기 상관없이 처리)**  
(총 5개의 Encoder모델과 Decoder모델로 구성된 복합 모델)  
(Image Encoder: Tiny ViT)  
(Prompt Encoder: Point Encoder(Forward, Backward)  
Box Encoder, No Box Encoder(박스 입력 없을 시 사용)  
Mask Encoder(Transformer 구조 기반))  
(Decoder: Two-way Transformer)
  - **Fps**
    - 34.55 => 34.49
  - **Memory Consume (MB)**
    - 1765 => 865

# 과제 결과물

- **Mobile-SAM(백승렬 - RTX3080\_Ti\_10GB - 입력크기 상관없이 처리)**  
(총 5개의 Encoder와 Decoder로 구성된 모델)  
(Image Encoder: Tiny ViT)  
(Prompt Encoder: Point Encoder(Forward, Backward)  
Box Encoder, No Box Encoder(박스 입력 없을 시 사용)  
Mask Encoder(Transformer 구조 기반))  
(Decoder: Two-way Transformer)
  - **Fps**
    - 34.55 => 34.49
  - **Memory Consume (MB)**
    - 1765 => 865



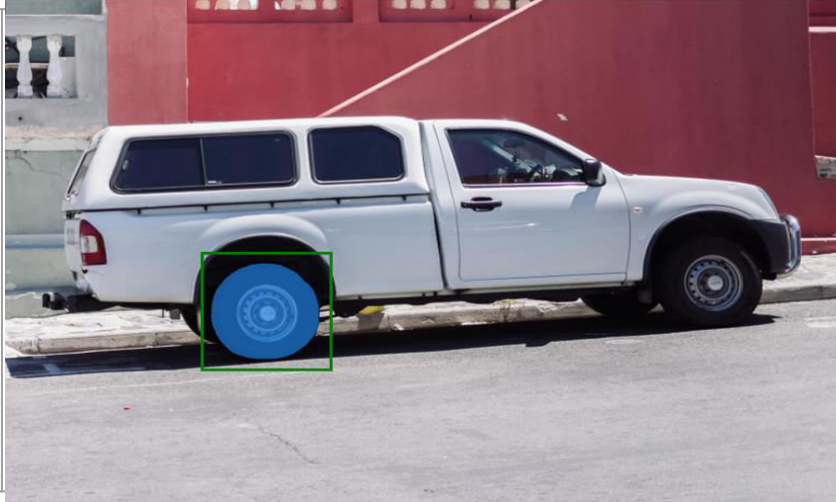
# 과제 결과물

<p>mobile-SAM Forward-point Embed</p>		<p>Mobile_SAM BOX&amp;Point Embed</p>	
<p>mobile-SAM with Backward-point Embed</p>		<p>Mobile-SAM Multi Object Segmentation</p>	



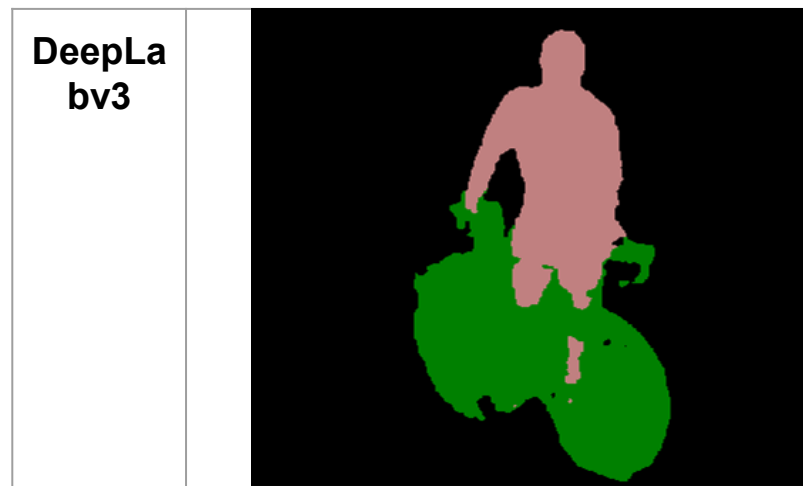
# 과제 결과물

mobile-  
SAM  
Box-  
Embed



# 과제 결과물

- DeepLabv3 (백승렬 - RTX3080\_Ti\_10GB - 입력크기 상관없이 처리)
  - Fps
    - 8.99 => 17.73
  - Memory Consume (MB)
    - 2105 => 837



# 과제 결과물

- REID-KIST-extraction (박제현-RTX4070\_Ti\_12GB-1X1280X720X3)

- Fps
  - 2.42 => 7.07
- Memory Consume (MB)
  - 1947 => 639

REID-KIST  
extraction

.json  
&  
.pt or npv  
feature

```
E> DEV-5.1.0 -past > data > reid-kist > testing > gallery > {} gallery_metadata_soyne.json > ...
1  {
2  },
3  { "img_path": "../../../data/reid-kist/testing/gallery/img/0000121_0000_person_0.85986328125.jpg"
4  },
5  { "img_path": "../../../data/reid-kist/testing/gallery/img/0000127_0000_person_0.86083984375.jpg"
6  },
7  },
8  { "img_path": "../../../data/reid-kist/testing/gallery/img/0000133_0000_person_0.88818359375.jpg"
9  },
10 },
11 { "img_path": "../../../data/reid-kist/testing/gallery/img/0000139_0000_person_0.88525390625.jpg"
12 },
13 },
14 { "img_path": "../../../data/reid-kist/testing/gallery/img/0000145_0000_person_0.85595703125.jpg"
15 },
16 },
17 { "img_path": "../../../data/reid-kist/testing/gallery/img/0000151_0000_person_0.8564453125.jpg"
18 },
19 },
20 { "img_path": "../../../data/reid-kist/testing/gallery/img/0000157_0000_person_0.83837890625.jpg"
21 },
22 },
23 { "img_path": "../../../data/reid-kist/testing/gallery/img/0000163_0000_person_0.84228515625.jpg"
24 },
25 },
26 { "img_path": "../../../data/reid-kist/testing/gallery/img/0000169_0000_person_0.8583984375.jpg"
27 },
28 },
29 { "img_path": "../../../data/reid-kist/testing/gallery/img/0000175_0000_person_0.87353515625.jpg"
30 }
```

# 과제 결과물

- REID-KIST-identification (박제현-RTX4070\_Ti\_12GB-X1280X720X3)

- Fps
  - 56.085 => 205.64
- Memory Consume (MB)
  - 2657 => 709



## 2학기 개발 계획

- **이론적인 부분에 대한 보충 (작업물의 활용 계획 및 작업 모델에 대한 이해)을 위한 스터디**
- 현재 작업한 모델들에 관한 논문 스터디
- 논문을 통해 이론적 배경에 대해 이해를 높일 예정
- 해당 논문들을 분석하여 작업한 모델들의 구체적인 활용에 대해 정리
- 포팅한 모델의 쓰임과 특화 부분에 대해 정리하여 기업에 제출할 계획

# 과제 성과

- REID-KIST-extraction, REID-KIST-identification, DeepLabv3, YOLOv7은 현재 활용 중(국민안전과제 SGI소프트)
- CCTV에 적용
- CCTV에 찍히는 영상을 프레임 별 입력으로 사용  
(최적화 작업을 통해 이러한 real-time 에서의 작업이 가능하게 됨.)
- 딥러닝 모델들을 활용해 지나가는 사람을 확인
- 이를 바탕으로 실종자를 찾을 수 있음

# 하계집중근무 수행 소감

## □ 이해성

- SW 제품 개발 과정을 전반적으로 알 수 있었던 좋은 기회였다. 실무에서 쓰이는 기술 스택을 익힐 수 있었고 SW를 설계 및 디버깅하는 실력이 향상되었다고 생각한다. 또한 인공지능 기술 발전의 흐름을 파악할 수 있었고 여러 모델들의 내부 구조에 대한 분석력을 기를 수 있었다. 여러모로 역량 강화에 도움이 많이되는 기회였다. 개인적으로는 학습(Training) 과제에서 기대한만큼의 성과를 얻지 못해 아쉬웠지만, 이 또한 개발자로 성장하는데에 자양분이 될 것이라 믿는다.

## □ 박세훈

- C++로 Tracking 알고리즘을 구현하면서 C++의 다양한 요소들과 조심해야 할 사항들을 많이 익힐 수 있었다. 예를 들어 cpp를 dll로 만드는 방법이나, 파일을 분리할 때 네임 충돌 등의 문제점에 대한 해결 방안들 등 많은 것을 배웠다. 무엇보다 회사에서 전반적인 AI, cpp, cuda, 네트워크 등의 지식을 강의해줘서 정말 값진 시간이 되었다. 전공적인 측면도 좋았지만 회사가 어떤 식으로 돌아가고 근무 분위기는 어떤지 파악할 수 있는 기회가 되어서 더욱 귀중한 시간이었던 것 같다.

# 하계 집중근무 수행 소감

## □ 김호재

- 그저 이론과 간단하게 사용해보기까지만 했던 인공지능 모델들을 직접 분석해보고 포팅을 진행하는 과정에서 많은 것을 배울 수 있었다. 직접 교육을 받는 현장에서 많은 시간 집중하고 인공지능이라는 분야만을 위해 공부하고 시간을 쓰는 과정에서 앞으로의 진로에 대해 생각의 가닥을 잡을 수 있었다. 또한 학과 공부 중에는 접할 기회가 많지 않았던 디버깅에 대해서 능숙하게 다룰 수 있게 되었으며, 코드를 보는 눈을 어느정도 뜨게 되었다. 많은 것을 배우고 많은 것을 얻어 갈 수 있었던 여름 방학이었다.

## □ 박제현

- C++를 통해 python으로 이루어진 코드를 리빌딩 하면서, 공간복잡도 및 시간복잡도를 다루고 최적화하는 과정에 대해서 직접 배울 수 있었다. 또한, 인공지능의 기본적인 구조를 직접다뤄보면서 앞으로 지속적으로 나올 AI들의 뼈대에 대한한 파악을 할 수 있어 추후 AI를 다룸에 있어 엄청난 도움이 될 것 같다. 모델에 대해 작업 하며 입력 데이터의 구조화 및 레퍼런스를 다루는 기술을 통해 직접 작업을 할 수 있었어서 앞으로 고객 데이터를 다루는 작업을 함에 있어서 어떤 방식으로 이루어져야 하는지 스케치를 파악할 수 있었다.



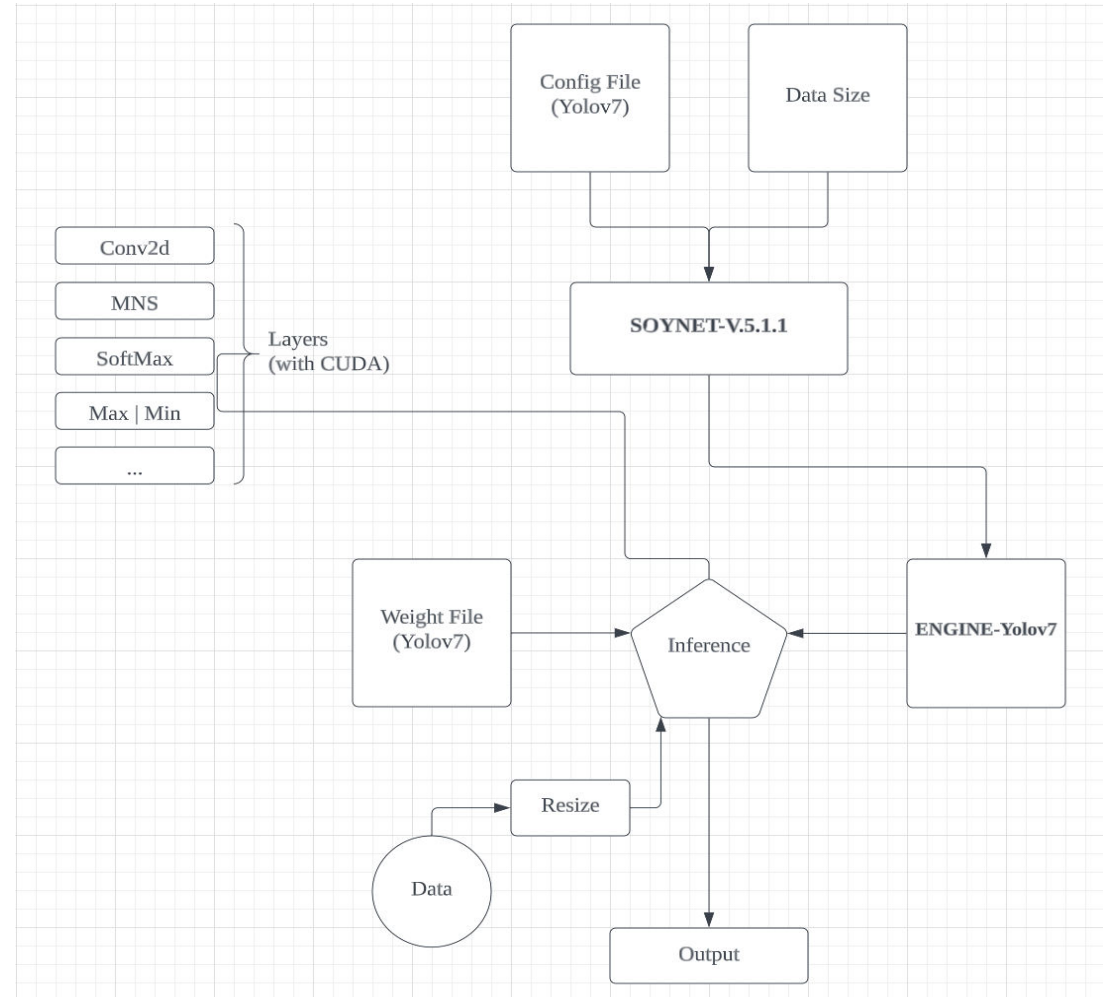
# 하계 집중근무 수행 소감

## □ 백승렬

- 인공지능에 대해 전반적인 이해를 높일 수 있었다고 생각한다. 먼저, 작업할 모델에 대한 이해를 하기 위해 모델에 관한 논문을 분석하여 발표하고 그에 대한 피드백을 받는 과정에서 인공지능에 대한 연구에서 어떠한 기여들이 이루어지고 있으며 어떠한 방향으로 논문에 접근해야 하는지에 대한 이해에 도움이 되었다. 또한 인공지능 모델의 코드를 한줄씩 분석해 나가며 config파일을 만들고 weight파일을 구성하며 모델내의 각각의 모듈 및 구조와 그에 맞는 학습된 가중치 값을 맞추어 나가며 딥러닝 모델 내의 추론 구조를 명확하게 이해할 수 있었으며 코드 분석 능력이 늘 수 있었다. 또한 상당히 거대한 딥러닝 추론 프로그램을 스스로 만드는 경험을 통해 프로그래밍 역량이 늘 수 있었고 소이넷의 솔루션에 아직 없는 딥러닝 모델 내의 일부 연산레이어를 직접 cpp과 cuda로 구현하는 경험을 통해 cpp과 cuda에 대한 이해가 늘 수 있었다. 또한 만들고 있는 프로그램과 원본 모델의 결과를 부분별로 분석하여 계속 비교해 나가고 프로그램을 여러 번 돌리며 실행결과를 분석해 디버깅 하는 과정에서 데이터 자체에 대한 이해를 필요로 하였으며 이 과정에서 데이터를 다룰 수 있는 역량도 늘었다고 본다. 결론적으로 쉽지는 않았지만 매우 도움이 되었다고 생각한다.

# Appendix

## □ 2023년 하계집중근무 과제 요약



# 2023 하계 집중근무 과제 요약

과제명	딥러닝 모델의 SoyNet 포팅	기업명 / 기업 멘토	소이넷 / 유경석 팀장님 /
과제개요	-배정받은 모델별로 - 모델의 역할과 구조파악을 위한 논문 읽기 - 소이넷 솔루션을 쓰기 위한 config파일 작성 - 추론에 쓰기 위해 가중치를 순서대로 불러올 w e i g h t 파일 작성 - 실제 추론을 시행하는 cpp과 파이썬 프로그램 제작 - 소이넷 솔루션을 이용해 만든 결과물과 원본 모델의 성능을 측정하여 benchmark작성	참여 학생	백승렬 (소프트웨어학과, 2학년), 팀장
			김호재 (소프트웨어학과, 2학년)
			박세훈 (소프트웨어학과, 2학년)
			박제현 (소프트웨어학과, 2학년)
			이해성 (소프트웨어학과, 2학년)
과제 기간	2023년 4월 **일 ~ 2023년 12월 31일	지도교수/ 멘토	이미향 교수
			xxx 멘토 (멘토가 있는 경우)
과제 최종 결과물	소이넷 솔루션을 이용한 원본과 동일한 작업을 하되 더 빠르고 메모리 사용량이 적은 딥러닝 추론 모델		