Portfolio

Machine Learning Engineer

Edge AI Computing Engineer

42dot 전문연구요원 신규편입 지원자

김 동 희

Personal Information

인적 사항

Dong-Hee Kim

M.S. student in Hanyang University

Embedded System on Chip Laboratory

Research Interest

- Reliable AI
- On-Device AI
- Reinforcement Learning

I am ASCII

AI/ML Generalist

Self-motivated

Collaborative

d**I**scussion-fr**I**endly

I AM

AI/ML Generalist

다음 단어를 **모두** 설명 할 수 있음

- SVM, PCA
- Word2vec, BERT
- SimCLR, EfficientNet
- Policy-based, Value-based, PPO

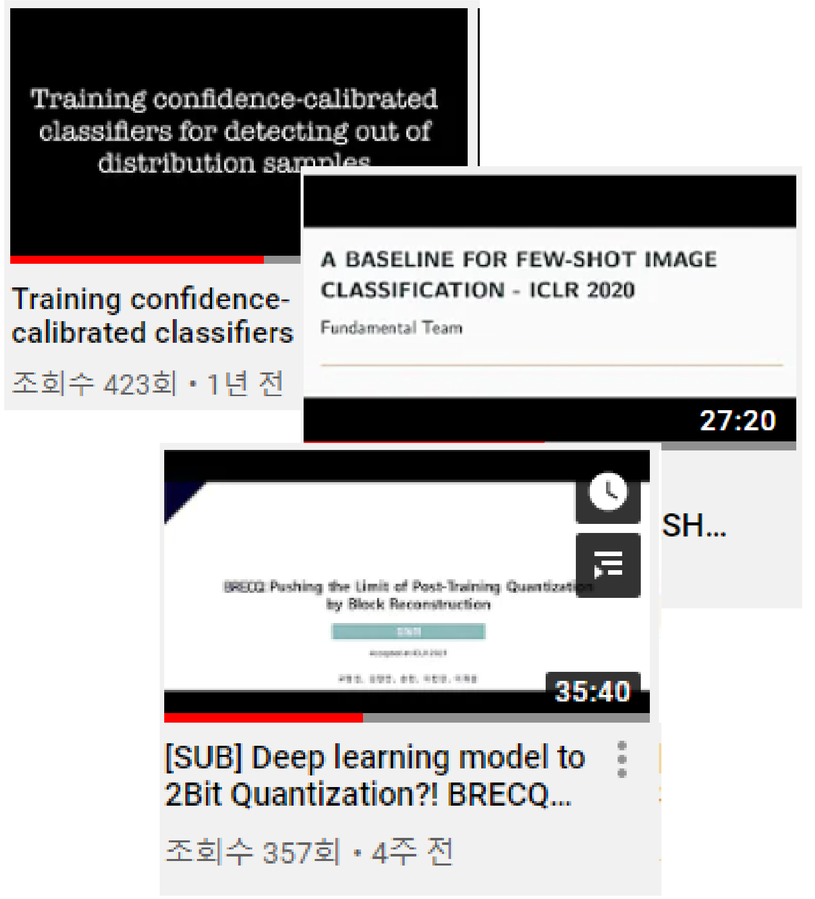
딥러닝 논문 스터디

- 연구 분야 외 다양한 분야의 논문 리뷰 (Quantization, Few-Shot Learning, Knowledge Distillation etc.)

→ **도메인을 넘나드는 딥러닝 지식**은 학계 & 산업계 모두 도움이 됨

진행하는 **프로젝트 & 논문**에선 **Specialist!**

- 2편의 논문 작성과 2 개의 프로젝트를 진행



<논문 리뷰 영상 예시>

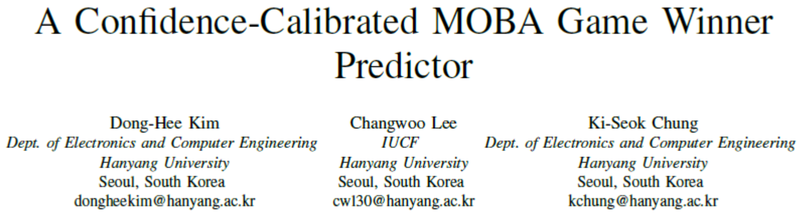
I AM

Self-Motivated

AI 관련 논문 **국제학회 1저자 논문 2편** 작성

- 1편은 게임 승률 예측, 1편은 OOD detection 관련
- 두 논문 모두 **연구실에서 누구도 연구하지 않았던** 주제

→ 논문, 프로젝트를 완료했을 때의 성취감으로 연구, 개발을 계속 함



A Confidence-Calibrated MOBA Game Winner Predictor

Dong-Hee Kim
Dept. of Electronics and Computer Engineering
Hanyang University
Seoul, South Korea
dongheekim@hanyang.ac.kr

Changwoo Lee
IUCF
Hanyang University
Seoul, South Korea
cwl30@hanyang.ac.kr

Ki-Seok Chung
Dept. of Electronics and Computer Engineering
Hanyang University
Seoul, South Korea
kchung@hanyang.ac.kr

AI 관련 **블로그 운영**

- 작성한 논문과 진행한 프로젝트에 대해 설명하는 블로그 운영
- 한 달 방문자수 약 200 명

→ 사고의 흐름을 정리하는 데 큰 도움을 줌

뿌요뿌요 강화학습으로 학습시키기 - 1

23 Aug 2019

Reading time ~3 minutes

소스 코드

뿌요뿌요의 추억

어렸을 때 공부하라고 보낸 학원에서 게임했다. 물론 게임만 했던건 아니고 당연히 쉬는시간 중간중간 마다 했으며 내가 했었던 게임 중에는 컴파일의 뿌요뿌요2가 있었고 바둑이나 룬키런을 강화학습으로 정복했듯 뿌요뿌요를 강화학습으로 훈련시킨다면 재미있는 프로젝트가 될 거라 생각했다. 필자는 처음 강화학습을 접했을 때 이 글을 보고 조금이나마 쉽게 텐서플로우와 강화학습 코드를 이해할 수 있도록 노력했다.(정말 문자 그대로 아~무것도 모른다면 블로그의 이전 글 중 강화학습 부분을 읽어 기본 지식은 습득한 상태라고 가정한다)

또한 즉시 실행으로 구글에서 그렇게 자랑하는 tensorflow 2.0으로 개발을 진행해보고 싶었고(tensorflow1의 sess.run이 싫기도 했고) 그에 따라 tf2 beta버전을 주로 사용하였다.

학습시킨 결과는 다음과 같다.

하드웨어 환경은 ryzen 1700X와 Nvidia의 Titan V를 사용했고, cuda 10.0을 Ubuntu 16.04LTS와 python 3.5를 virtualenv 함께 사용했으나 강화학습은 NN의 깊이가 깊지 않기 때문에 CPU로 학습하는 것과 GPU로 학습하는 것의 차이가 없다고 봐도 무방하다. 필자가 직접 로컬 환경에서 구동하진 않았으나 무겁지 않은 코드니 중분히 개인이나 Colab에서도 구동시킬 만하다고 생각한다.

Collaborative

I AM

한양대학교 커뮤니티 위한(weehan.com) 운영진

- Web Programmer (HTML, javascript, jquery)
- **디자이너, 기획자**와의 **협업**

→ 프로젝트의 리더가 되어 기획자, 디자이너 등 **개발 지식이 없는 사람**과 **소통** 방법을 배움

제 1회 현대자동차 해커톤

- 음주운전 예방을 위한 시동잠금장치 및 리워드 서비스
- Developer (Arduino, PHP)

→ 시간이 빠듯할 때 **팀을 위하여** 스스로를 **한계까지 밀어붙여 본 경험**

공동 1저자 논문 작성

- Notion, Git을 통한 **코드 협업**

→ 공동연구를 진행하면서 나오는 실험 설계, 서술 **이견** 등을 원만히 **조율**

What Matters for Out-of-Distribution Detectors using Pre-trained CNN?

Dong-Hee Kim*
dongheekim@hanyang.ac.kr
Jaeyoon Lee*
dlwodbs1421@gmail.com
Ki-Seok Chung
kchung@hanyang.ac.kr

Department of Electronic Engineering
Hanyang University
Seoul, Republic of Korea

* Equal contribution

I AM

Discussion-Friendly

질문에서 시작하는 Discussion이 많음

- 질문 예시) “High-Performance Large Scale Image-Recognition Without Normalization” 논문 발표 중
- ‘논문에서 Batch Normalization의 단점으로 Network가 깊어질수록 Variance가 커지는 것을 지적하고 있는데 제안한 Adaptive Gradient Clipping은 Variance를 조정한다는 내용은 발표 중에 없었다. AGC가 Variance를 조정한다는 내용은 혹시 논문에 언급되어 있는가?’

→ 질문을 통해 **확실하게 내용을 이해**할 수 있고 **새로운 인사이트**를 얻을 수도 있음

언제든지 틀릴 수 있다고 생각함

- 첫 번째 논문 작성 시 처음에는 “이건 논문이 아니라 일기다.”라는 의견을 듣기도 했으나 **연구 방향을 수정**하여 국제학회에 게재.
- 공동 1저자 논문 작성 시 연구실 후배와의 **토론을 통해** 서술방향, 실험방향 설정

→ **실수**를 확실하게 **인지**하고 **재발 방지**를 위해 노력함

Build-Up

Paper #1

League of Leagend 대회 중계를 보던 도중...

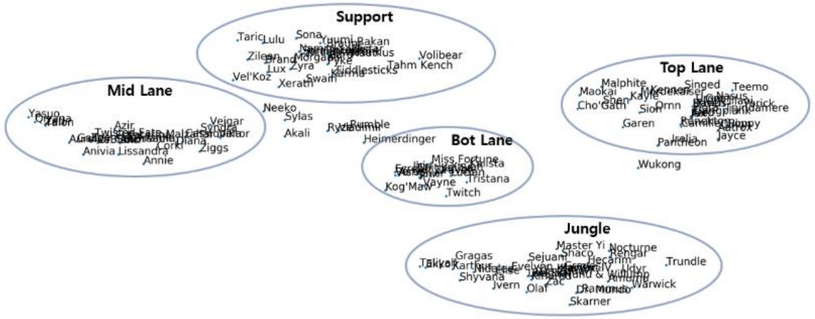
- 우측사진과 같이 마지막 픽으로 팀을 구성하려 할 때 **캐릭터 구성으로 승률을 예측할 수 있는 예측기**에 LoL에 존재하는 모든 캐릭터를 다 대입한 후 승률이 가장 좋은 캐릭터를 선택한다면 **가장 좋은 캐릭터를 고를 수** 있지 않을까?

→ 조합만으로 승패를 예측할 수 있는 예측기를 **딥러닝으로 설계**하자!

LoL 조합 승률 예측기 설계

- **Pandas**를 **사용**하여 수집한 데이터를 **전처리**
- Word2vec을 LoL 데이터셋에 적용하여 Champ2vec을 통해 **벡터화 시도**
- CNN, LSTM 등 기존 딥러닝 방법 적용

→ 벡터화 후 캐릭터 벡터들은 **캐릭터의 특징**을 잘 살리고 있음



Build-Up

Paper #1

실망스러운 결과

- 여러가지 방법을 통한 승리 팀 예측 결과는 **뚜렷한 차이를 나타내지 못함**
- 51%로 A 팀의 승리를 예측하는 것과 99% 확률로 A 팀의 승리를 예측했을 때 모두 Accuracy로 성능을 측정한다면 **Accuracy의 변화가 없음**

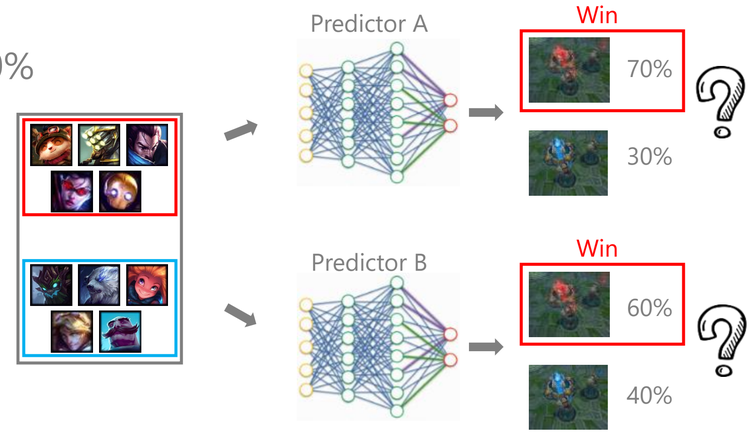
TABLE II
INGAME DATA ACCURACY

	Model		
	Multi-hot	CNN	RNN
Baseline	74.81%	73.39%	73.8%

새로운 가정

- A라는 predictor는 70%확률로 승리를 예측하고 B라는 predictor는 60% 확률로 승리를 예측한다면 더 좋은 predictor는 무엇일까?

→ '어느 팀이 유리한가'라는 사실보다 **'어느 정도 유리한가?'**가 **더 중요**



Motivation

A Confidence-Calibrated MOBA Game Winner Predictor
(IEEE Conference on Games 2020, **1st author**)

Paper #1

Tech Stack:
Python, Pandas, Pytorch, Riot API

Question

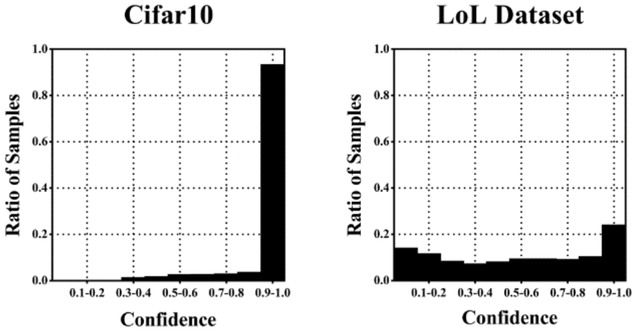
- 실시간으로 LoL의 경기 결과를 예측하는 Predictor가 70% 확률로 A 팀이 이긴다고 할 때 진짜 A 팀의 승리 확률이 70%일까?

→ 그 상황에서 진짜 승리 확률이 70%인지 평가할 수 있는 **Confidence-Calibration 성능** 또한 Predictor의 **평가 지표로 사용**되어야 함

Problem

- 이미지 데이터셋과 LoL 데이터셋의 Confidence 분포가 다름
- 이미지에서 적용되는 Calibration 방법은 게임 데이터에 적합하지 않음

→ **Data Uncertainty를 고려**한 **Confidence-Calibration 방법**이 필요



* 발표 영상: https://youtu.be/jRyFk-WHOo0

A Confidence-Calibrated MOBA Game Winner Predictor
(IEEE Conference on Games 2020, **1st author**)

Paper #1

Tech Stack:

Python, Pandas, Pytorch, Riot API

Solution

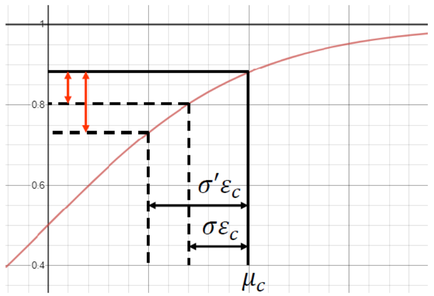
Solution

- LoL의 승리 확률을 예측하기 위한 Data Uncertainty Loss Function 제안
- $\mathbf{u}^{(k)} \sim \boldsymbol{\mu}(\mathrm{x}) + \mathrm{diag}(\sigma(\mathrm{x}))\epsilon, \epsilon \sim \mathcal{N}(0, I)$ $\quad \mathbb{E}[\mathbf{p}] \approx \frac{1}{K}\sum_{k=1}^{K} \mathrm{Softmax}(\mathbf{u}^{(k)})$(μ: mean vector, σ: data uncertainty)

Analysis & Result

- 다음과 같이 불확실성의 크기에 따라 calibration 효과가 달라짐
- 이외 올바른 Calibration을 위한 효과 세 가지 분석

→ **승률 예측기의 Confidence-Calibration 성능 증가**



A Confidence-Calibrated MOBA Game Winner Predictor
(IEEE Conference on Games 2020, **1st author**)

Paper #1

Tech Stack:

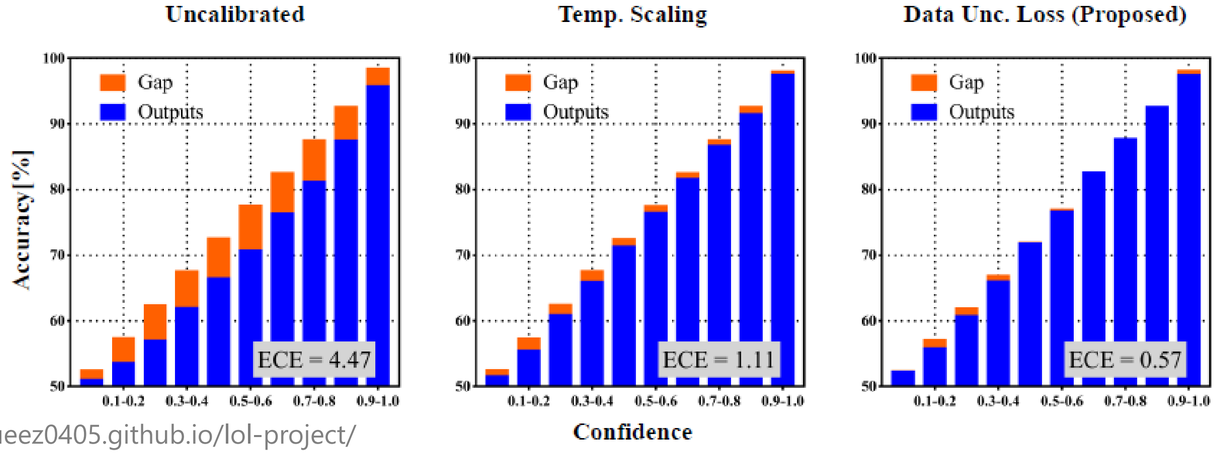
Python, Pandas, Pytorch, Riot API

Results

Results

- Data Uncertainty Loss Function 적용 시 대표적인 Metric인 ECE를 포함한 calibration 성능 향상

Method	Accuracy[%]	ECE[%]	MCE[%]	NLL
No calibration	72.95	4.47	6.76	0.515
Temp. Scaling	72.95	1.11	1.87	**0.505**
Vector Scaling	72.71	4.63	6.66	0.536
Matrix Scaling	73.05	4.43	6.90	0.540
DU Loss (Proposed)	**73.81**	**0.57**	**1.26**	0.515



* 블로그 참조: https://queez0405.github.io/lol-project/

Motivation

What Matters for Out-of-Distribution Detectors using pre-trained CNN?
(Tentative, Under Writing, **co-1st author**)

Paper #2

Tech Stack:
Python, Pytorch, Git

Problem

- Pre-trained 된 CNN으로 OOD sample을 Detection하는 방법론은 많은 장점이 있음
- 최근 제안된 OOD Detection 방법들은 뛰어난 성능을 보여주고 있음

→ **그러나 기존 논문은 implementation detail이나 evaluation을 사소하게 치부하는 경우가 존재함**

기존 논문에서 충분히 검증하지 못한 예시

- Mahalanobis는 2018년 제안된 pre-trained 기반 방법론
- G-ODIN은 2020년 제안된 full-training 방법론

OOD	TNR at 95% TPR	AUROC
	Mahalanobis / G-ODIN	
SVHN	89.29 / **93.18**	98.02 / **98.69**
LSUN	**95.40** / 88.12	**98.99** / 97.50
TinyImageNet	**93.05** / 80.81	**98.66** / 96.05

→ Mahalanobis를 사용하기 위해서는 **Tuning이 필요**한데 기존 논문에 언급되어 있지 않고 본 논문에서 제안한 **적절한 Tuning**을 통한다면 **최근 방법**보다 OOD Detection **성능이 비슷하거나 좋을 수 있음**

* OOD detection task: 학습에 사용하지 않은 데이터 분포가 입력으로 들어왔을 때 이를 알아채서 탐지하는 것

Solution & Dataset

What Matters for Out-of-Distribution Detectors using pre-trained CNN?
(Tentative, Under Writing, **co-1st author**)

Paper #2

Tech Stack:
Python, Pytorch, Git

Solution

- Pre-trained OOD detector의 성능에 영향을 줄 수 있는 **7가지 요인**을 분류
- 이미지의 의미가 보존되는 OOD samples, CNN을 훈련시키기 위한 데이터 수, Contrastive Learning의 사용 유무 등
- 4가지 서로 다른 pre-trained OOD detector에서 **각 요인의 영향 분석**

→ 각 요인의 **영향 분석 후** 일반적으로 가장 좋은 성능을 낼 수 있는 **조건**을 **'Recommendation'**

Dataset & Metrics

- In-Distribution: CIFAR-10, CIFAR-100, SVHN
- Out-of-Distribution: CIFAR-10, CIFAR-100, SVHN, LSUN-resize, LSUN-FIX, TinyImageNet-resize, TinyImageNet-FIX
- Metric: TNR@95%TPR, AUROC, Detection Accuracy

What Matters for Out-of-Distribution Detectors using pre-trained CNN?
(Tentative, Under Writing, **co-1st author**)

Paper #2

Tech Stack:

Python, Pytorch, Git

Result

ID	OOD	TNR at 95% TPR	AUROC	Detection Accuracy
		Baseline / ODIN / Mahalanobis / Gram		
CIFAR-10	SVHN	56.76 / 69.08 / **99.46** / 97.39	93.36 / 93.82 / **99.64** / 99.43	88.43 / 88.28 / **98.05** / 96.62
	LSUN	58.42 / 81.98 / **98.61** / 95.27	93.64 / 95.35 / 99.63 / **99.85**	88.25 / 89.99 / 97.53 / **98.55**
	TinyImageNet	49.74 / 70.99 / 97.03 / **98.63**	91.02 / 91.84 / 99.35 / **99.70**	85.46 / 86.03 / 96.36 / **97.59**
	LSUN FIX	46.23 / **58.84** / 40.93 / 32.81	**90.24** / 90.20 / 87.91 / 84.64	**84.78** / 84.43 / 80.89 / 77.24
	TinyImageNet FIX	46.77 / **56.99** / 48.91 / 39.36	**89.97** / 88.99 / 89.13 / 85.12	**85.46** / 83.47 / 81.90 / 77.26
	CIFAR-100	41.77 / **49.48** / 47.67 / 33.06	88.00 / 86.93 / **88.27** / 80.95	**82.68** / 81.50 / 80.93 / 73.52
CIFAR-100	SVHN	27.10 / 63.74 / **93.43** / 80.01	83.43 / 94.79 / **98.12** / 96.01	76.14 / 89.70 / **95.24** / 89.48
	LSUN	28.48 / 58.79 / 95.60 / **97.92**	82.25 / 92.15 / 98.63 / **99.47**	74.81 / 84.60 / 95.38 / **96.92**
	TinyImageNet	30.15 / 61.17 / 90.21 / **96.13**	82.73 / 92.81 / 98.01 / **99.10**	74.94 / 85.35 / 92.98 / **95.77**
	LSUN FIX	13.07 / **15.57** / 13.48 / 10.74	73.14 / **73.97** / 67.65 / 64.62	68.45 / **69.07** / 63.37 / 60.95
	TinyImageNet FIX	22.63 / **24.97** / 12.69 / 20.54	78.58 / **78.73** / 70.61 / 74.19	72.28 / **72.69** / 65.93 / 68.34
	CIFAR-100	20.60 / **20.77** / 8.17 / 12.80	78.78 / **79.45** / 62.63 / 68.79	72.16 / **72.93** / 59.74 / 59.39

Semantic Vague

Semantic Preserved

Experiment Example #1 (semantic preserved OOD sample)

- 기존 benchmark로 사용되는 LSUN, TinyImageNet은 오른쪽 위 그림과 같이 의미를 찾기 어려움
- FIX 데이터는 이를 보완하여 아래와 같이 낮은 해상도에서도 의미가 살아있음

→ **semantic preserved sample에 대해서는 Baseline이 가장 나은 성능을 보이기도 함**

What Matters for Out-of-Distribution Detectors using pre-trained CNN?
(Tentative, Under Writing, **co-1st author**)

Paper #2

Tech Stack:

Python, Pytorch, Git

Result

ID	OOD	TNR at 95% TPR	AUROC	Detection Accuracy
		Cross Entropy (base) / Supcon		
CIFAR-10	SVHN	56.76 / **85.70**	93.36 / **97.69**	88.43 / **92.40**
	LSUN	58.42 / **72.29**	93.64 / **96.04**	88.25 / **90.63**
	TinyImageNet	49.74 / **62.50**	91.02 / **94.51**	85.46 / **88.71**
	LSUN FIX	46.23 / **62.73**	90.24 / **94.56**	84.78 / **88.62**
	TinyImageNet FIX	46.77 / **60.78**	89.97 / **94.12**	84.57 / **88.00**
	CIFAR-100	41.77 / **52.49**	88.00 / **91.95**	82.68 / **85.32**
CIFAR-100	SVHN	29.46 / **33.82**	84.39 / **86.49**	76.81 / **79.26**
	LSUN	**32.40** / 26.79	**84.00** / 83.15	75.88 / **76.33**
	TinyImageNet	**30.51** / 20.71	**82.98** / 80.13	**74.92** / 73.71
	LSUN FIX	12.99 / **15.48**	72.99 / **73.90**	68.28 / **68.81**
	TinyImageNet FIX	22.67 / **24.70**	78.36 / **79.16**	71.95 / **72.75**
	CIFAR-10	**20.73** / 18.20	**78.69** / 76.01	**72.02** / 70.27

Table 18: Supcon with Baseline.

Experiment Example #2 (Supervised Contrastive Learning)

- Cross-Entropy로 학습했을 때보다 Supervised Contrastive Loss를 사용했을 때 성능이 좋음
- 특히 **Semantic Preserved 이미지**에서 큰 **성능 향상**이 관찰
- 특정 Detector에서는 성능이 떨어지기도 함

→ **Semantic Preserved Image를 Detection 하기 위해서는 Contrastive Learning도 고려해 볼만한 Option이다**

Project

뿌요뿌요 AI 개발, 개인 프로젝트

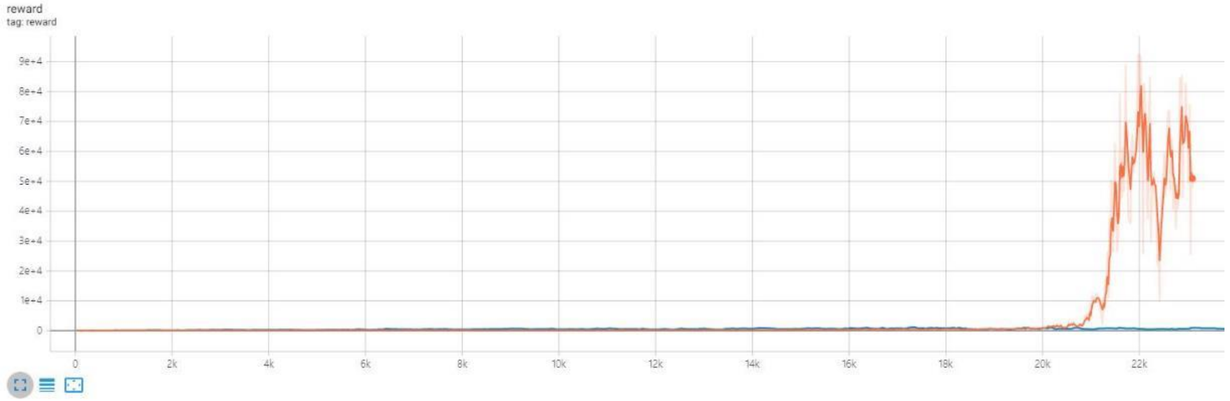
Project #1

Tech Stack:

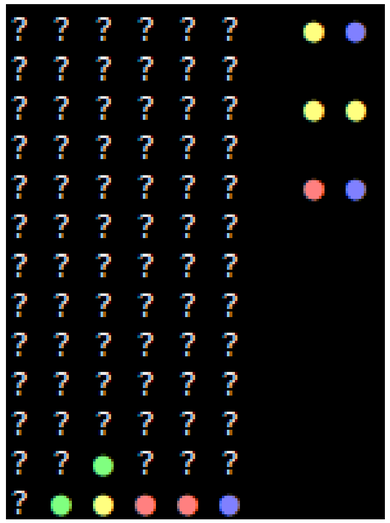
Python, Tensorflow, OpenAI gym

뿌요뿌요를 플레이하여 점수가 올라가도록 하는 AI

- 뿌요뿌요 학습을 위한 Environment 구현
- 강화학습 알고리즘 중 **Actor-Critic**과 **A2C**를 **한 줄 한 줄** 구현
- 60시간 이상의 학습 진행 후 10000점 이상 기록.
- Github: https://github.com/queez0405/puyopuyoRL



<Episode가 지남에 따른 Return 그래프>



<Environment 예>

* 블로그 참조: https://queez0405.github.io/puyoRL-1/

Project

임베디드 환경 기반 제어 지능화 기술 플랫폼 개발, LG 전자, 19.12 ~ 20.11

Project #2

Tech Stack:

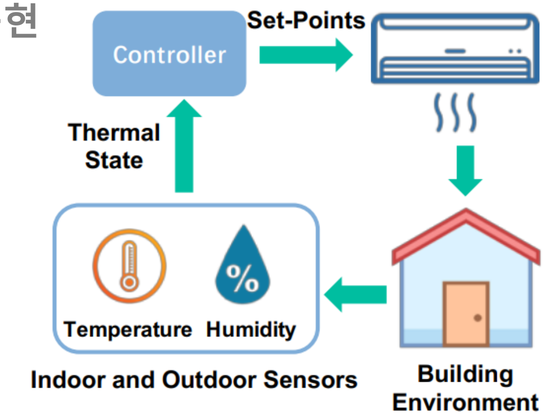
Python, OpenAI gym, C, Darknet

경량 하드웨어에 최적화된 딥러닝 프레임워크 ALWAYSNet 개발

- **ALWAYSNet**: **A**dvanced **L**ayer **W**ise **A**na**ly**sis and Optimization for Deep Neural **Net**work
- ARM Cortex-A9가 장착된 임베디드 보드에서 동작하는 강화학습 Agent 개발
- 담당 업무: **C 언어**의 Darknet 프레임워크를 통한 **REINFORCE 알고리즘 구현**
- 결과: 10 episode reward의 평균이 높아지는 것을 확인

Problem

- 실내 공조기의 성능을 강화학습을 통해 optimize 시도
- 고성능 컴퓨터를 설치할 수 없는 임베디드 환경
- 최적화를 위해 C 언어 사용



〈공조시스템을 위한 강화학습 기반 제어기의 예〉

Project

임베디드 환경 기반 제어 지능화 기술 플랫폼 개발, LG 전자, 19.12 ~ 20.11

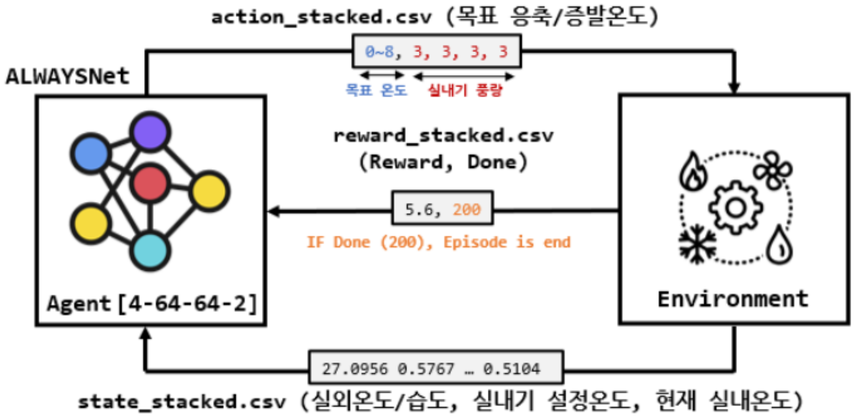
Project #2

Tech Stack:

Python, OpenAI gym, C, Darknet

Environment

- Pascal을 기반으로 하는 Dysim 공조 시뮬레이터
- .csv 파일에 state, reward 기록
- Reward는 목표온도와 사용 전력을 응용한 지표



< ALWAYSNet-DySim 상호작용 인터페이스 >

Agent

- C 언어를 사용하는 Darknet 프레임워크를 활용
- Action space가 작지 않기 때문에 Policy gradient 방법론 중 REINFORCE 사용
- .csv 파일에 action을 기록

Thank You