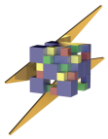
Rock Paper Scissors Classification

조교 정우근



THUNDER Research Group
Seoul National University
서울대학교 천둥 연구실

프로젝트 개요

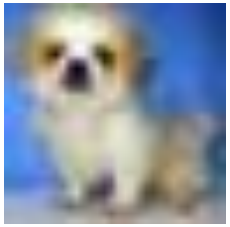
- 가위, 바위, 보를 구분(classify)하는 CNN 모델 학습
  - 간단한 hand gesture recognition project 코드를 수정하여 구현
    - https://github.com/SparshaSaha/Hand-Gesture-Recognition-Using-Background-Elllimination-and-Convolution-Neural-Network

- 총 세 가지 Task를 진행
  - Task 1 (50%): 공개된 데이터셋을 사용한 RPS classifier 구현
  - Task 2 (10%): SPDS-RPS 데이터셋 만들기
  - Task 3 (40%): SPDS-RPS 챌린지

뼈대 코드 설명

전통적인 인공 신경망

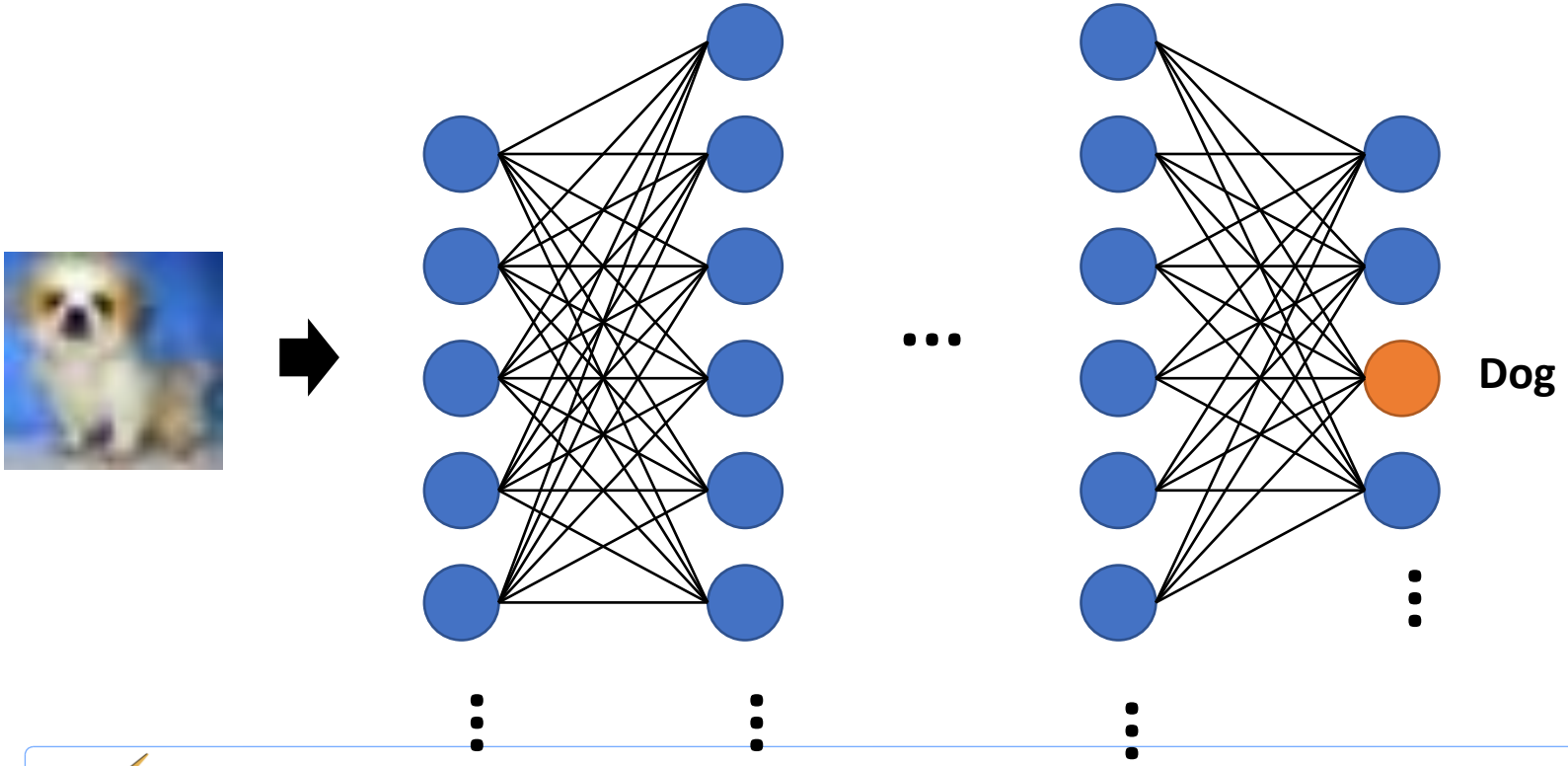
- 인공 뉴런으로 신경망을 구성
- 이미지 픽셀 값을 입력으로 넣어, 뉴런의 동작을 모사



Dog

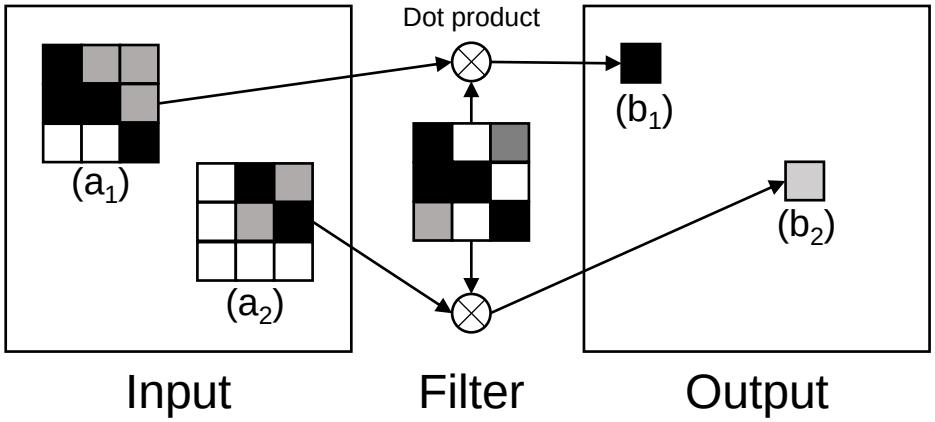
전통적인 인공 신경망

- 인공 뉴런으로 신경망을 구성
- 이미지 픽셀 값을 입력으로 넣어, 뉴런의 동작을 모사



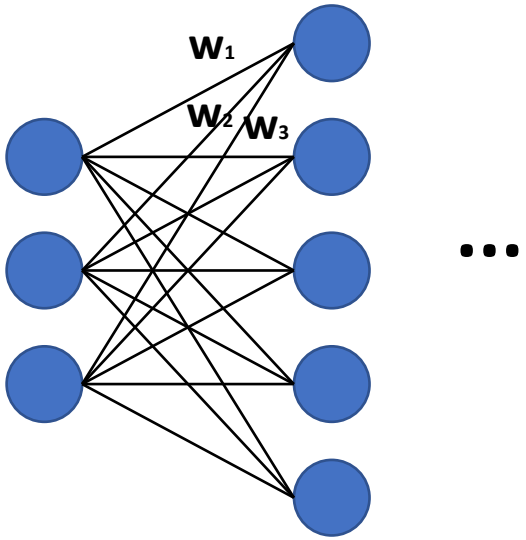
Convolution

- 이미지 내에서 필터와 유사한 패턴을 찾아내는 연산



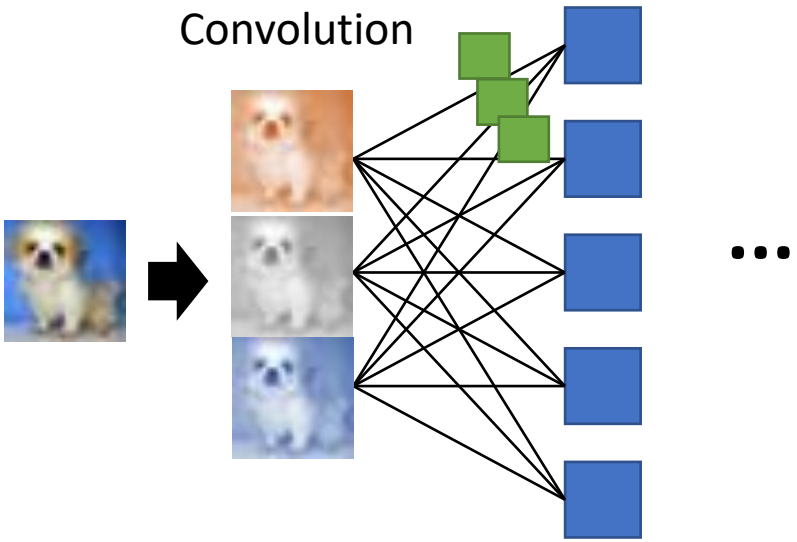
Convolutional Neural Network

- Convolution을 반복적으로 수행하는 인공 신경망



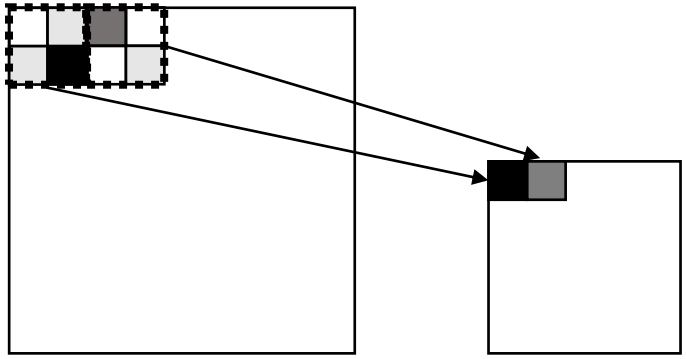
Convolutional Neural Network (CNN)

- Convolution을 반복적으로 수행하는 인공 신경망



Pooling

- 이미지 크기를 줄이는 연산
- 인접한 2x2 픽셀 중 가장 값이 큰 픽셀만 고름



2x2 max pooling

사용 네트워크 구조

- 뼈대 코드는 convolution, ReLU activation, pooling을 반복적으로 수행하는 CNN을 사용하여 이미지를 분류
  - 2x2 convolution, 2x2 max pooling을 반복적으로 실행

```
# Define Model
model = nn.Sequential(nn.Conv2d(1, 32, 2, padding=1),
                      nn.ReLU(),
                      nn.MaxPool2d(kernel_size=2),
                      nn.Conv2d(32, 64, 2, padding=1),
                      nn.ReLU(),
                      nn.MaxPool2d(kernel_size=2),
                      nn.Conv2d(64, 128, 2, padding=1),
                      nn.ReLU(),
                      nn.MaxPool2d(kernel_size=2),
                      nn.Conv2d(128, 256, 2, padding=1),
                      nn.ReLU(),
                      nn.MaxPool2d(kernel_size=2),
                      nn.Conv2d(256, 256, 2, padding=1),
                      nn.ReLU(),
                      nn.MaxPool2d(kernel_size=2),
                      nn.Conv2d(256, 128, 2, padding=1),
                      nn.ReLU(),
                      nn.MaxPool2d(kernel_size=2),
                      nn.Conv2d(128, 64, 2, padding=0),
                      nn.ReLU(),
                      nn.MaxPool2d(kernel_size=1),
                      torch.nn.Flatten(),
                      nn.Linear(64, 1000, bias = True),
                      nn.Dropout(0.75),
                      nn.Linear(1000, 3, bias = True),
                     )
```

뼈대 코드

- project.zip
- PyTorch 구현 제공
- 뼈대 코드는 palm, swing, fist를 구분하는 코드
  - palm, swing, fist로 label된 데이터셋을 사용하여 학습을 진행함
  - 뼈대 코드에 주어진 데이터셋이 아닌 다른 데이터셋을 사용하도록 코드를 수정, 학습을 진행하는 것이 프로젝트의 목표

뼈대 코드

- main.py
  - 실행 인자를 처리하는 코드
- dataset.py
  - 데이터셋을 읽는 코드
- train.py
  - 학습을 진행하며 진행도를 출력하는 코드

프로젝트 내용

딥 러닝 기술 활용(이상)

딥 러닝 기술 활용(현실)

?

Task 1: 공개된 데이터셋을 사용한 RPS classifier 구현

- TensorFlow rock-paper-scissors 데이터셋을 사용하여, 가위바위보 구분 학습을 진행
  - https://www.tensorflow.org/datasets/catalog/rock_paper_scissors
  - 데이터셋을 직접 받아서 사용해도 되고, 실습 클러스터의 /home/share/tf-rps 디렉토리에 저장된 데이터셋을 사용하여도 됨

- 학습 결과, Validation accuracy가 80% 이상이 되도록 구현하여야 함

Task 1 구현 시 유의사항

- **모델 구조는 뼈대 코드에서 주어진 그대로 사용**
  - 고칠 시 0점 처리
  - 고쳐도 되는 지 헷갈리는 부분이 있으면 조교에게 메일로 문의
- **Validation dataset을 training input으로 사용하면 안 됨**
  - 사용 시 0점 처리
- **실습 클러스터 기준, 실행 시간이 10분을 넘어가면 안 됨**
- **Pretrained model은 사용하면 안 됨**
- 그 외의 부분은 원하는 대로 수정 가능
  - 입력 이미지 처리 방법
  - Batch 크기
  - Learning rate
  - Dropout rate
  - ...

Task 2: SPDS-RPS 데이터셋 만들기

- 자신의 손 사진을 가위, 바위, 보 각각 10장씩 찍어서 제출
  - 학생들의 손 사진을 모아서 SPDS-RPS 데이터셋을 만들 예정
  - Task 3 evaluation을 위한 테스트셋으로 사용 예정

Task 2 제출 시 유의사항

- spdsXXX.zip 파일(xxx은 본인 계정 번호)을 ETL에 제출
  - 총 세 개의 디렉토리에 각각 jpg 파일이 10개씩 들어있는 압축 파일을 제출
  - 디렉토리명은 r, p, s
    - r, p, s는 각각 바위, 보, 가위
  - 사진 파일명은 spdsXXX_[r/p/s]_[번호].jpg
    - 번호는 0부터 9까지
    - 예: ID가 spds199이고 주먹 사진을 찍은 경우
      - spds199_r_0.jpg, spds199_r_1.jpg, ..., spds199_r_9.jpg
  - 예시 업로드 파일 참고

Task 3: SPDS-RPS 챌린지

- SPDS-RPS 데이터셋의 일부분(20%)을 공개 예정
  - 6/7 공개 예정
- 공개하지 않은 80%의 데이터셋으로 제출물 평가
  - 가장 높은 정확도를 달성한 수강생이 만점 획득
  - 타 학생들은 정확도에 따라 점수 차등 배분
  - 채점 예시
    - 1등 수강생 정확도의 70% 미만인 경우 40점 만점 중 5점 획득
    - 나머지 학생들은 정확도 순으로 정렬하여 5점~40점 사이의 점수를 획득

Task 3: SPDS-RPS 챌린지

- 제출 시 evaluate.py 파일을 같이 제출
- 사용법: python3 evaluate.py [DIR] [RESULT]
  - DIR directory 안에 있는 이미지 파일 (1.jpg, 2.jpg, ...) 각각에 대해 classify 후, RESULT 파일에 한 줄 당 하나씩 classification 결과를 출력
  - evaluate.py 예제는 SPDS-RPS dataset과 같이 공개 예정

- evaluate.py가 정상 동작하기 위해 미리 실행해야 하는 작업이 있으면 보고서에 이에 대해 명시할 것
  - 예: training, data preprocess 등

Task 3 구현 시 유의사항

- **학습 데이터셋은 Task 1과 마찬가지로 TensorFlow RPS 학습 데이터셋 사용**
  - TensorFlow RPS 학습 데이터셋을 기반으로 하되, 다양한 종류의 data augmentation을 허용함
  - 기본적으로 "손 모양 데이터"는 TensorFlow RPS 데이터셋의 것을 사용해야 하나, 그 외의 augmentation을 위해서는 다양한 기법 및 외부 데이터셋 사용 가능
  - 본인의 아이디어가 규칙 상 허용되는지 판단이 되지 않을 경우 조교에게 메일로 문의
- **SPDS-RPS 데이터셋은 validation용으로만 사용할 것**
- **실습 클러스터 기준, 실행 시간이 10분을 넘어가면 안 됨**
- **Pretrained model은 사용하면 안 됨**
- 그 외의 부분은 원하는 대로 수정 가능
  - **모델 구조**
  - 입력 이미지 처리 방법
  - Batch 크기
  - Learning rate
  - Dropout rate
  - ...

제출 기한 및 제출물

- SPDS-RPS 데이터셋 (Due: 6/4 23:59)
  - spdsXXX.zip 파일 제출
    - 디렉토리 구조, 확장자, 파일명 등의 제약사항에 유의할 것
    - 틀릴 시 감점될 수 있음

- 소스 코드 및 보고서 (Due: 6/18 23:59)
  - spdsXXX.zip 파일에 report.pdf, task1 디렉토리, task3 디렉토리를 담아 제출
  - Task1 디렉토리
    - 1번 task와 관련된 소스 코드 및 requirements.txt
    - 실습 클러스터 환경에서 실행 가능한 python 코드여야 함 (ipynb 파일 제출 시 0점 처리)
    - 특정 패키지 설치 등이 필요한 경우, 필요 설치 과정을 보고서에 작성
  - Task3 디렉토리
    - 3번 task와 관련된 소스 코드 및 requirements.txt
    - 반드시 evaluate.py 파일을 포함하여야 함

※ requirements.txt는 필요한 패키지를 한 번에 설치하기 위한 파일로, 본인의 코드 실행을 위한 패키지를 예시와 같이 제출해야 하며, pip install –r requirements.txt로 패키지가 정상적으로 설치되지 않거나, 본인 코드 실행에 필요한 패키지가 누락되어 있을 경우 감점될 수 있음

GPU 사용

GPU 사용

- 제출물 평가는 실습 클러스터에서 진행 예정
  - 프로젝트 진행은 각자의 개발 환경에서 자유롭게 진행하여도 상관 없으나, 제출 전에 반드시 실습 클러스터에서 정상 동작 여부를 확인해야 함

- Slurm을 통해 GPU를 사용하는 인터페이스를 제공

Slurm: Task Scheduler

- Slurm에 작업 제출하는 방법
  - 작업 제출은 login02 (147.47.200.188), login03 (147.47.200.192) 노드에서 가능 (포트: 22)
  - sbatch run.sh로 작업 제출하면 되며, run.sh 내용은 아래와 같음
  - srun.sh의 내용은 아래 두 줄 외에는 수정하면 안 됨
    - 가상환경을 사용하는 경우 conda activate spds 부분을 본인 환경에 맞게 설정
    - srun python main.py --epochs 30의 argument는 필요에 따라 적절히 수정

lurm-kyusu | 2 Slurm-kyusu

```
#!/bin/bash

#SBATCH --job-name=spdspjt       # Submit a job named "example"
#SBATCH --nodes=1                # Using 1 node
#SBATCH --gpus=1                 # Using 1 GPU
#SBATCH --time=0-00:10:00        # 10 minute timelimit
#SBATCH --mem=16000MB            # Using 16GB memory
#SBATCH --cpus-per-task=8        # Using 8 cpus per task (srun)
#SBATCH --output=log.txt         # Creating log file

echo ${USER}
eval "$(conda shell.bash hook)"
conda activate spds              # Activate your conda environment

srun python main.py --epochs 30
```

Slurm: Task Scheduler (Cont'd)

- Task schedule 여부 확인 방법
  - 'squeue –u 계정명' 으로 task가 schedule 되었는지 확인 가능

```
 master U:3311 ?:4  (c) base  ~/03_TA/SPDS-2021/final_pjt/Hand-Gesture-Recognition/answer_fps
❯ sbatch run.sh
Submitted batch job 1839
 master U:3311 ?:4  (c) base  ~/03_TA/SPDS-2021/final_pjt/Hand-Gesture-Recognition/answer_fps
❯ squeue -u kyusu
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
              1838 gsds_3090  spdspjt    kyusu  R       0:12      1 gpu13
              1839 gsds_3090  spdspjt    kyusu  R       0:05      1 gpu13
```