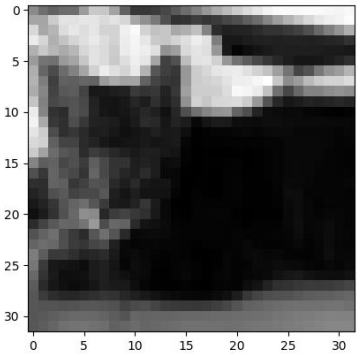
인공지능 팀프로젝트
딥러닝을 이용한 영상 분류

Artificial Intelligence Lab
Kyonggi University

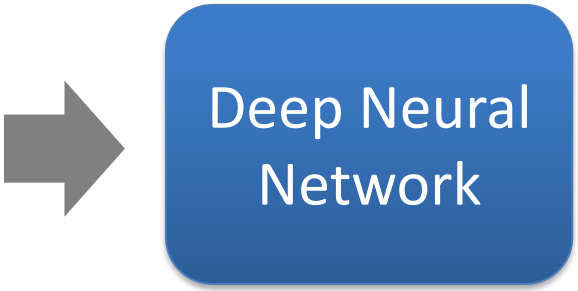
프로젝트 내용

- 프로젝트 목표
  - 1. 문제 해결을 위한 인공신경망 설계
  - 2. 주어진 데이터셋으로 모델을 학습하고 평가

- 주제
  - 강아지와 고양이 영상 분류하기



영상 데이터



분류 결과

프로젝트 결과물

- 제출 파일
  - 1. 구현 코드 (main.py)
  - 2. 팀별 보고서 (팀명.hwp or 팀명.docx)
    - 2-0. 팀소개 [팀명, 팀원(학번/이름)]
    - 2-1. 모델 특징 설명
    - 2-2. 모델 학습 및 실험 결과
    - 2-3. 결과에 대한 평가글
    - 2-4. 핵심 코드 (*get_hypterparameter(), get_model()*)
  - 팀명.zip 으로 압축해서 업로드

- 예시 파일과 같은 양식으로 제출

프로젝트 안내

- 팀 구성
  - 팀별 3~4명
- 제출기한 (2주)
  - 월요일반 : 11월 26(일) 자정까지
  - 화요일반 : 11월 27(월) 자정까지
  - 수요일반 : 11월 28(화) 자정까지
- 관련 문의
  - 조교 메일 : sdh9446@gmail.com

평가 사항

- 1. 모델 설계 이유
  - 왜 이런 구조로 설계하였는 지
- 2. 모델의 성능 비교 분석
  - 모델 구조 및 학습 파라미터 별 성능 분석
  - 최대한 다양한 방면에서 모델 비교 분석
  - ex) 유닛 수에 따른 성능 비교, FC 레이어 수에 따른 성능 비교 등
- 3. 모델의 성능 분석 평가
  - 결과에 대한 주관적인 평가
  - 개선할 수 있는 방안

과제 내용 설명

데이터셋

- 데이터 분할 및 크기
  - Train data# : 8,000개
  - Validation data# : 2,000개
  - Test data# : 2,000개

- 입력 데이터
  - 흑백 영상
  - shape : 32x32x1
- 출력 데이터
  - 영상 라벨 {0: 고양이, 1: 강아지}
  - shape : 1

1. 모델 구현, 학습, 평가

- 기본 코드 제공 (main.py, util.py)
  - 데이터셋, 기본 모델, 학습, 평가 코드 제공
  - main.py를 실행하여 학습, 평가 수행
  - 환경 세팅 필요 (다음 슬라이드에서 설명)

- 구현(수정)해야 할 사항
  - **모델 구조**
    - get_model() 함수 구현
    - *BatchNormalization* 사용 금지
  - **모델 학습 파라미터**
    - get_hyperparameter() 함수 수정
    - batch size, epochs(최대 20), optimizer, learning late 값 수정

기본 코드 설명

- main.py를 실행하여 모델 학습 및 평가 가능

- ModelMgr() : 모델 학습/평가 클래스
  - train() : 모델 학습
  - test() : 모델 평가
  - get_hyperparameter() : 학습에 사용될 하이퍼파라미터 정의
  - get_model() : 구현할 모델 정의
    - get_model_sample_1() : 샘플 모델1 (MLP base)
    - get_model_sample_2() : 샘플 모델2 (CNN base)
  - draw_history() : 모델 학습, 평가 결과 그래프 생성
    - result.png 파일에서 확인 가능
  - save/load model() : 학습된 모델 저장 및 불러오기
- **get_hyperparameter(), get_model() 외는 수정 금지**

구현 및 수정해야하는 코드 부분 - 1

- 모델 부분 구현
  - get_model() 구현
  - 샘플 코드 2개 제공[각 성능: MLP:60%, CNN:65%]

```
def get_model(self):
    model = Sequential()
    ################
    '''
     모델 코드를 완성해주세요.
     '''
    ################
    ##
    '''
     주의사항
     1. 모델의 입력 데이터 크기는 (batch_size, 32, 32, 1) # 고양이 or 강아지 흑백 사진
        출력 데이터 크기는 (batch_size, 2) # 고양이일 확률, 강아지일 확률
     2. 최초 Dense() 사용 시, Flatten()을 먼저 사용해야함
     3. out of memory 오류 시,
         메모리 부족에 의한 오류임.
         batch_size를 줄이거나, 모델 구조의 파라미터(ex. 유닛수)를 줄여야함

     기타 문의 : sdh9446@gmail.com (수업조교)
     '''
    ##
    return model
```

구현 및 수정해야하는 코드 부분 - 2

- 하이퍼파라미터 수정
  - get_hyperparameter() 수정

```
def get_hyperparameter(self):
    hyper = dict()
    ################################
    '''
     파라미터들을 수정해주세요
    '''
    hyper['batch_size'] = 16  # 배치 사이즈
    hyper['epochs'] = 10  # epochs은 최대 20 설정 !!
    hyper['learning_rate'] = 0.1  # 학습률
    # 최적화 알고리즘 선택 [sgd, rmsprop, adagrad, adam 등]
    hyper['optimizer'] = optimizers.sgd(lr=hyper['learning_rate'])
    ################################
    return hyper
```

수정 대상

2. 보고서

- 1. 모델 특징 설명
- 2. 모델 학습 결과
  - **최종 모델 성능, 학습 결과 그래프**
- 3. 모델 성능 실험
  - **비교 실험 3개** 이상 수행
- 4. 결과 평가
  - 실험 및 결과에 대한 **주관적인 평가**

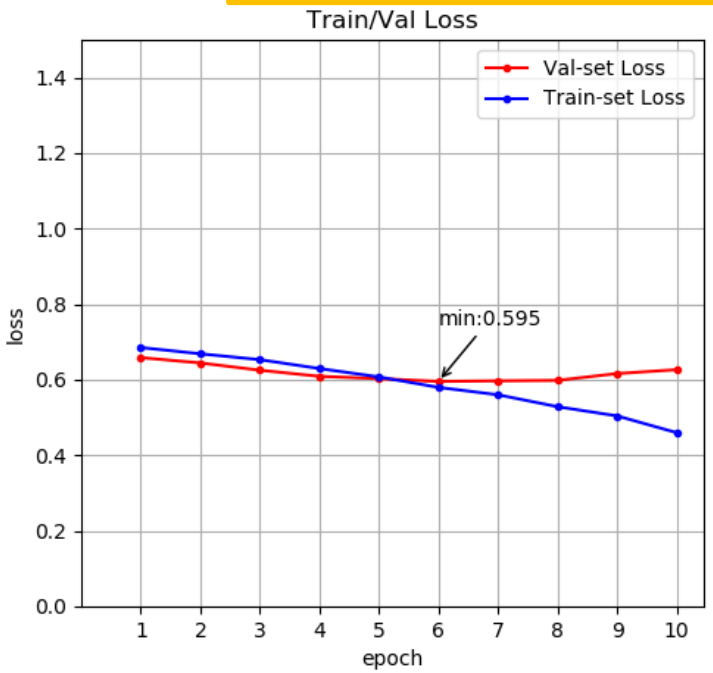
학습 결과 그래프

- 학습, 평가 종료 시 자동으로 result.png 파일 생성

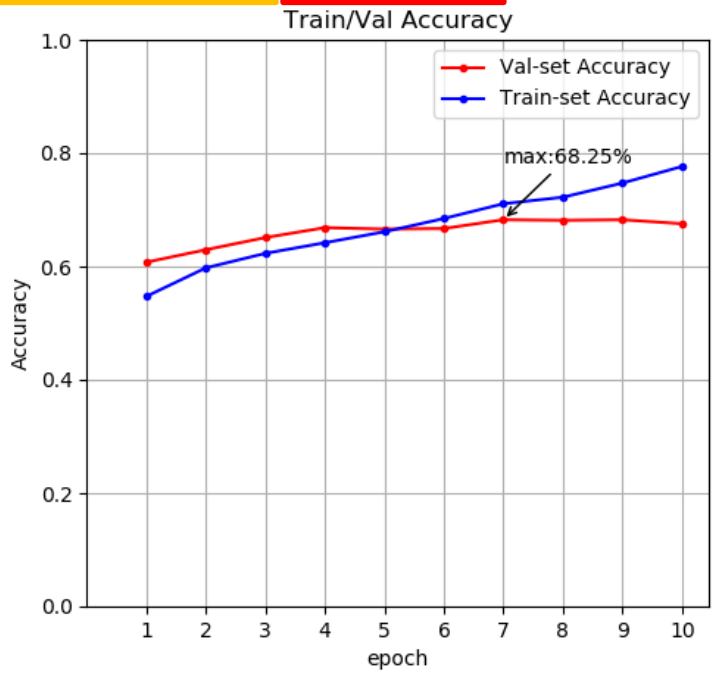
학습에 사용된 하이퍼파라미터

test dataset 정확도 (최종 성능)

batch_size:16 epochs:10 learning_rate:0.1 optimizer:SGD Test_Acc:67.64%



Train/Val Loss Graph (lower is better)



Train/Val Accuracy Graph (higher is better)

모델 구현

환경 세팅 (설치 방법)

- 요구 사항 : python 3.6 [2.X, 3.7 불가]
- 1. 명령 프롬프트 실행
- 2. 딥러닝 라이브러리 설치
  - 아래 명령어를 순서대로 입력하여 설치 진행
    - python –m pip install numpy matplotlib
    - python –m pip install tensorflow
    - python –m pip install keras
    - python –m pip install scikit-learn
  - (Anoconda 사용 시, 환경 활성화 후 입력)

- IDE는 pycharm 추천
  - https://www.jetbrains.com/pycharm/download/#section=windows
  - 위 링크에서 **Community** 버전 설치

개발 절차

- Keras : 딥러닝 라이브러리 중 하나
- 인공신경망 개발 절차
  - 1. 문제 분석
  - 2. 모델 구조 설계 (with Keras)
    - Sequential()로 모델 정의
    - add 함수로 Layer, 활성화 함수 등 추가 (구조 설계)
  - 3. 모델의 Loss, Optimizer 설정
  - 4. 모델 학습
    - batch size, epochs, learning rate 설정
    - 정의된 epochs 만큼 모델 학습
  - 5. 모델 평가
    - 예측 정확도로 모델 성능 확인
  - 6. 학습된 모델 적용 (본 프로젝트에서는 해당하지 않음)
    - 모델을 저장하고, 실제 환경에서 모델을 불러와 사용

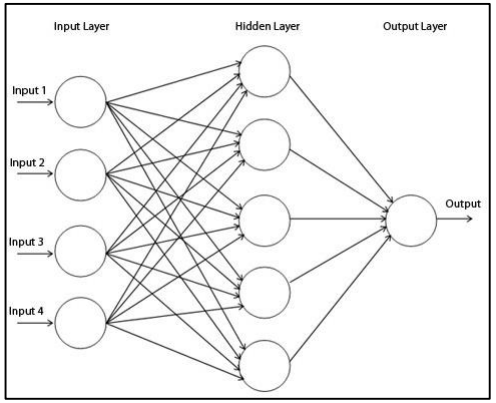
```
def get_model_sample_1(self):
    # Fully-connected layer만을 이용한
    model = Sequential() # 모델 정의
    # Flatten 함수는 다차원 배열을 벡터
    model.add(Flatten(input_shape=self
    model.add(Dense(64))  # fully-conn
    model.add(Activation('sigmoid'))
    model.add(Dropout(0.5))  # 이전 ou
    model.add(Dense(64))
    model.add(Activation('sigmoid'))
    model.add(Dense(len(self.target_cl
    model.add(Activation('softmax'))

    return model
```

모델 코드 예시

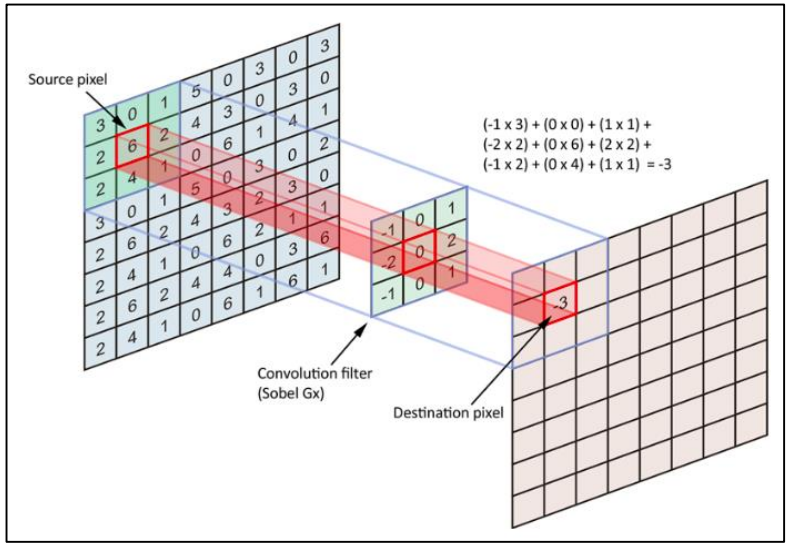
기본 Layer (in Keras)

- Dense()
  - fully-connected layer
  - 파라미터
    - unit 수



Dense Layer

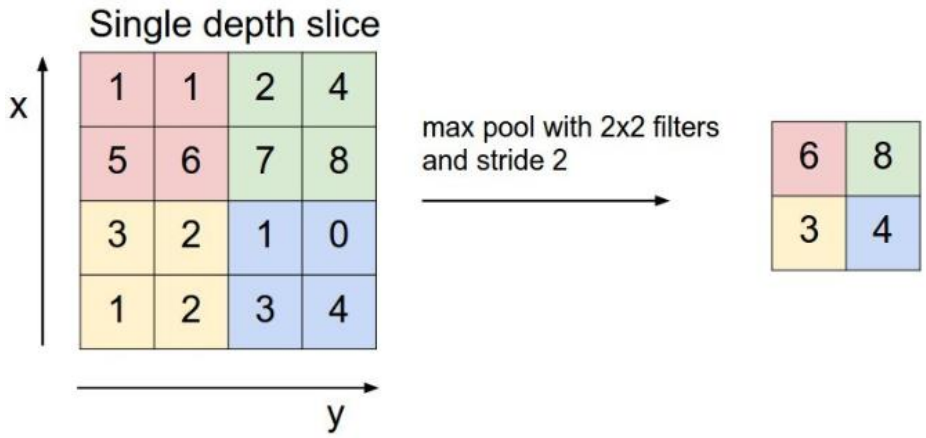
- Conv2d()
  - convolutional layer
  - 파라미터
    - 필터 크기, 필터 수
    - stride 크기, padding 방법



Conv2d Layer

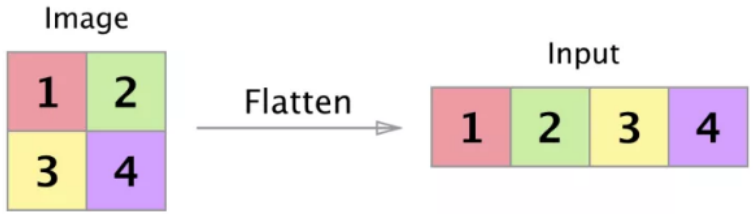
기본 Layer (in Keras)

- MaxPooling2D()
  - pooling layer
  - 파라미터
    - stride 크기
    - pooling 범위 크기



MaxPooling2D Layer

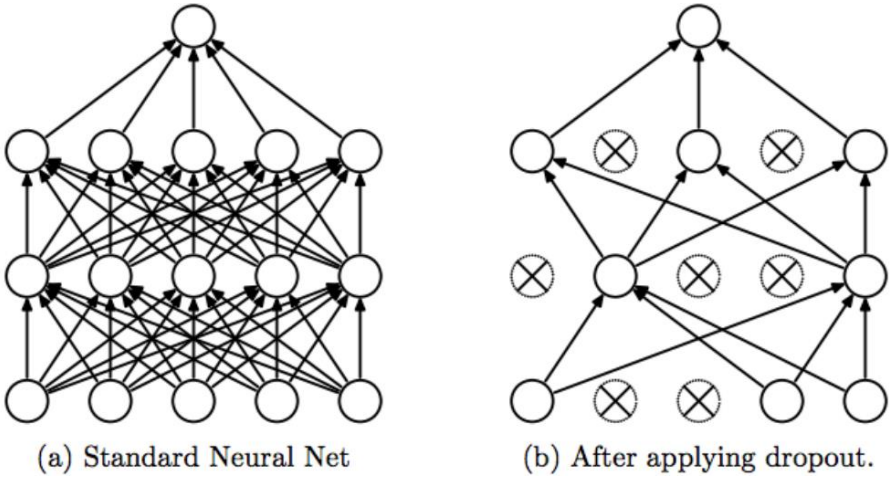
- Flatten()
  - flatten layer
  - 파라미터
    - 없음



Flatten Layer

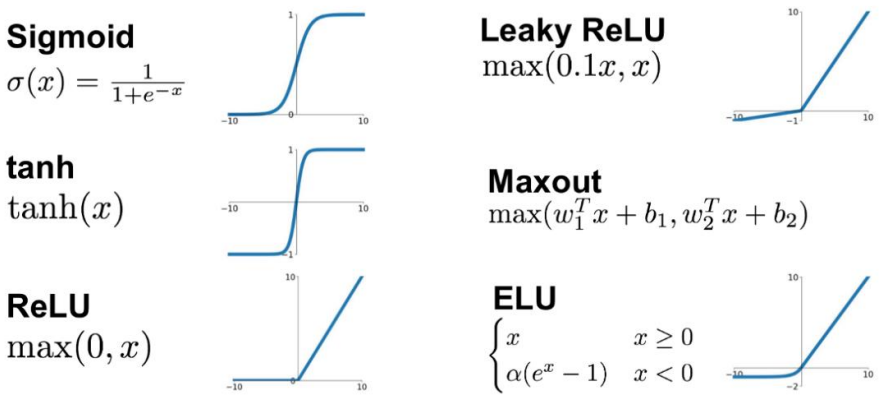
기본 Layer (in Keras)

- Dropout()
  - pooling layer
  - 파라미터
    - dropout 비율



Dropout Layer

- Activation()
  - Activation function
  - 파라미터
    - 활성화 함수 타입
    - Ex) sigmoid, tanh …



Activation Functions