

CNN을 활용한 한우 부위 분류 모델 개발

<최적 파마리터 선정을 통한 모델 최적화>

이덕희



CNN을 활용한 한우 부위 분류 모델 개발

1. 연구 목적

- 서론 및 이론적 배경

2. 연구 목표

3. 연구 방법

4. 분석 결과

5. 결론 및 시사점

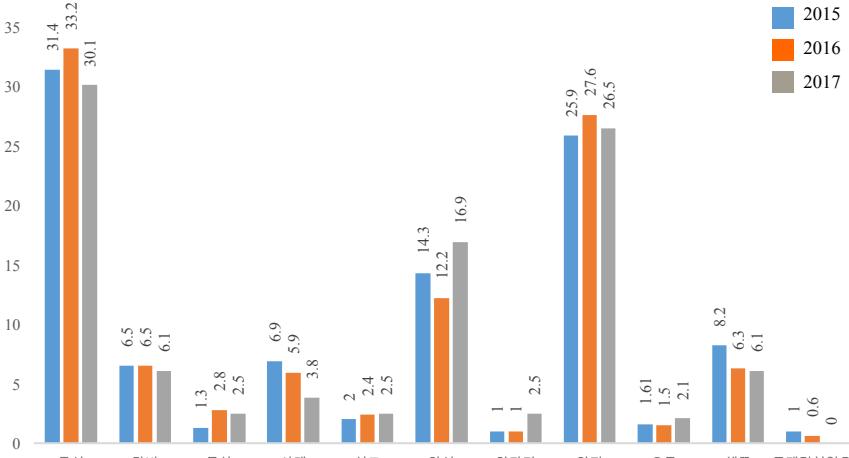


01 연구목적

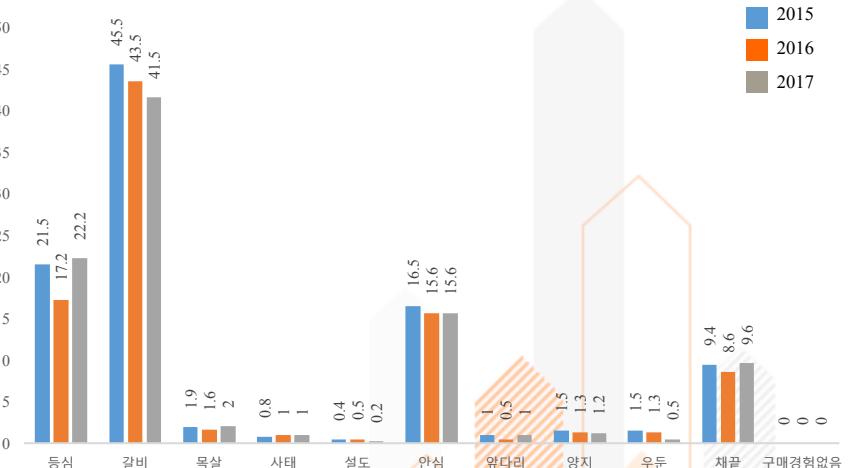
소비자들의 한우 부위 선호도 - 특정 부위 선호 경향

02 소비자 대상 연도별 한우 소비 모니터링 결과

가구 소비자 한우 주요 구입 부위



외식 소비자 한우 주요 섭취 부위

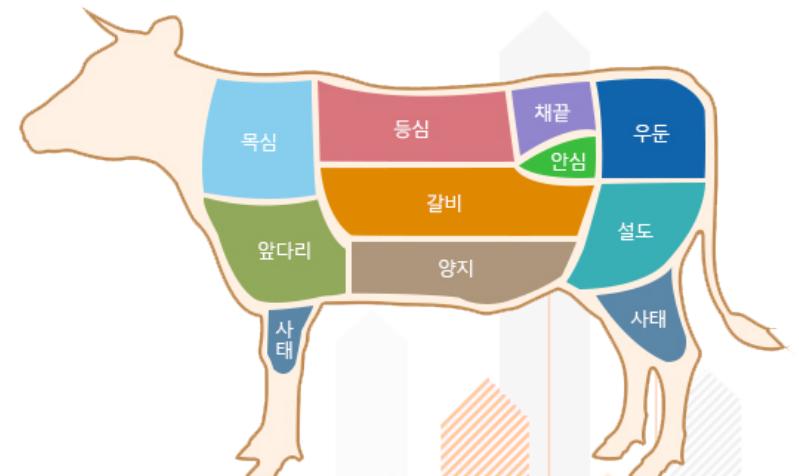
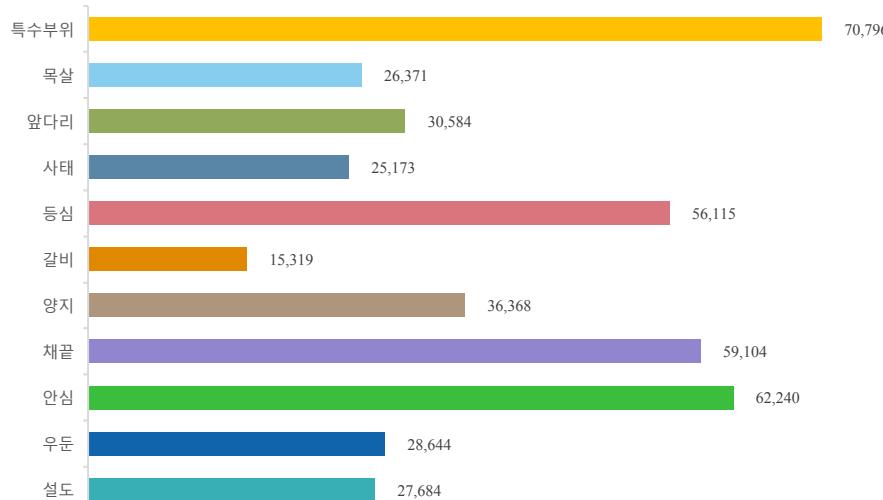


등심, 갈비, 안심, 양지, 채끝, 살치, 부채살 총 7가지 부위를 연구 표본으로 선정

01 연구목적

타 육류에 반해 부위별로 세분화되어 유통/판매 되고 있는 한우

03 2016년도 기준 한우 부위별 도매가격 (단위 : 원/kg)



선호 부위별 일정한 패턴이 존재할 것으로 예상되며, 학습 모델로써 유의미할 것으로 판단

02 연구목표

내가 어떤 부위를 먹고 있는가?



03 연구방법

표본 선정

01 표본 선정 및 기간

01 표본 선정

- 등심, 갈비, 안심, 양지, 채끝 : 대표 선호 부위
- 살치, 부채 : 채끝과 비슷한 한우 부위



02 기간

- 2018/11/01 ~ 11/15 : 비정형 데이터 수집
(google web crawling)
- 2018/11/15 ~ 11/22 : 분류 모델 개발



03 연구방법

주요 단계

02 연구 방법

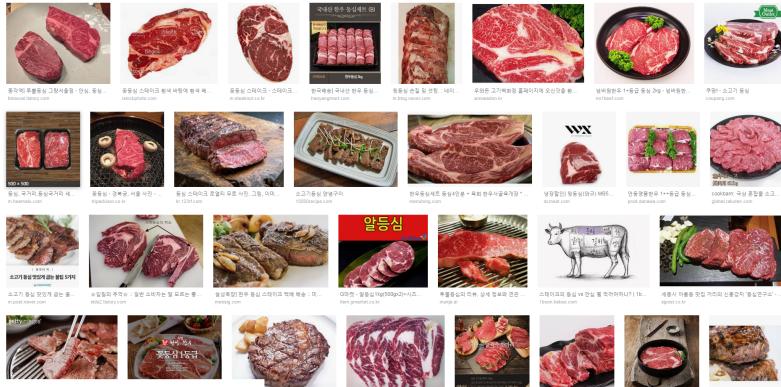
- 
- 01 비정형 데이터 정제 Direct Handling
 - 02 데이터셋 구성 I Load_IMG
 - 03 데이터셋 구성 II Resizing + Rotation + Transpose
 - 04 학습 모델 Keras
 - 05 연구 방법 Grid search/ 학습 파라미터 / 모형 / 프로세스
 - 06 파라미터 최적화 I Optimizer
 - 07 파라미터 최적화 II Learning Rate
 - 08 파라미터 최적화 III Activation function
 - 09 파라미터 최적화 IV Dropout, Weight Constraint
 - 10 파라미터 최적화 V Number of Neurons

03 연구방법

비정형 데이터 정제

Direct Handling

Selenium crawling 을 통해 얻은 데이터의 형태



형태 : 익힌 형태, 그림, 해당하지 않는 이미지 등 다수 포함

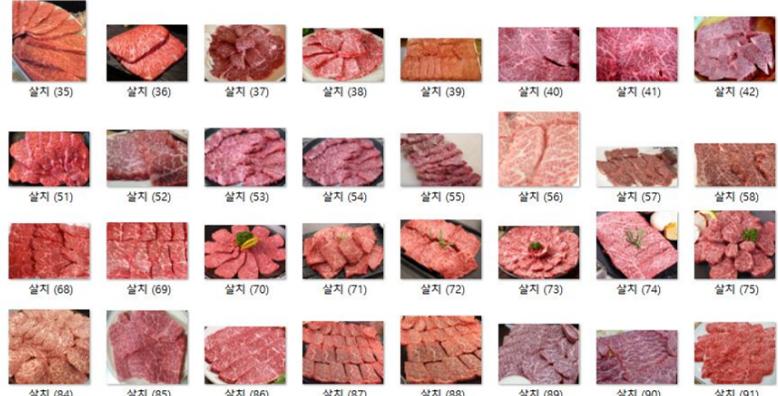
데이터 품질 : 매우 나쁨

Accuracy : 23.8

수집 속도 : 부위 1개당 20 minute

총 수집량 : 약 3000 개

Direct Handling 을 통해 얻은 데이터의 형태



형태 : 고기의 결, 마블링, 모양이 보이는 이미지

데이터 품질 : 매우 향상됨

Accuracy : 68.4

수집 속도 : 부위 1개당 8시간 소요

총 수집량 : 약 680 개

03 연구방법

데이터셋 구성 I

이미지 추출

01 load_img

01 설명

이미지를 불러오는 함수

02 타입

함수

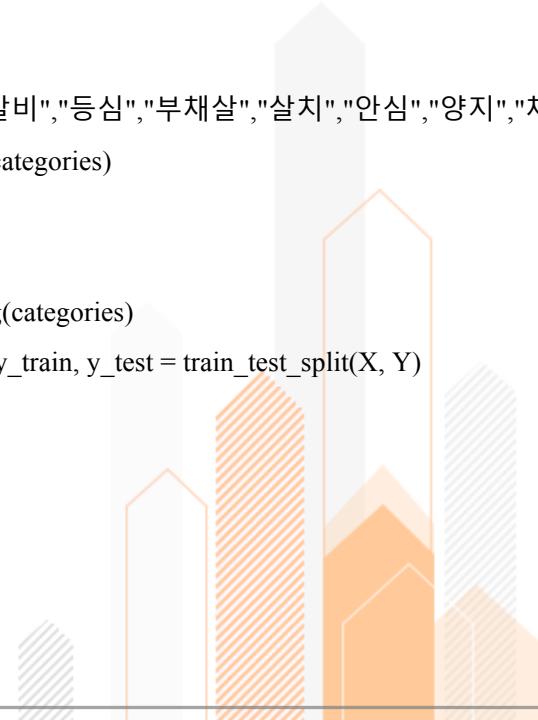
인자값	타입	기본값	리턴값	리턴 타입
categories	list	-	RGB값, 라벨값	Array

03 예시

```
categories = ["갈비", "등심", "부채살", "살치", "안심", "양지", "채끝"]  
nb_class = len(categories)  
img_size = 128
```

```
X, Y = load_img(categories)
```

```
x_train, x_test, y_train, y_test = train_test_split(X, Y)
```



03 연구방법

데이터셋 구성 II

Resizing + Rotation + Transpose

01 IMG resizing



Img_size = 128 X128

02 IMG Rotation + Transpose



680 개 → 약 1 만 개

03 연구방법

학습 모델

01 opt_param

01 설명
전역변수 선언

02 타입
dictionary

	인자값	타입	기본값	용도
optimizer	str		RMSprop	옵티마이저
activation	str		relu	활성화 함수
dropout_rate	float		0.25	drop의 정도
neurons	int		512	뉴럴 수
batch_size	Int		100	배치 사이즈
epochs	Int		50	epochs
learn_rate	float		0.001	옵티마이저 학습률
weight_constraint	int		1	dropout 가중치로 과적합 방지

03 연구방법

학습 모델

02 build_model

01 설명

모델 생성

02 타입

함수

03 예시

```
model = build_model()
```

인자값	타입	기본값	리턴값	리턴 타입
optimizer	str	opt_param['optimizer']		
learn_rate	float	opt_param['leran_rate']		
dropout_rate	float	opt_param['dropout_rate']		
neurons	int	opt_param['neurons']		
weight_constraint	int	opt_param['weight_constraint']	model	Keras.model
activation	str	opt_param['activation']		

03 연구방법

학습 모델

03 Optimizer learning rate 설정

01 설명

옵티마이저 학습률 설정

02 타입

함수

03 예시

```
opt_list = opt(opt_param['learn_rate'])
```

인자값	타입	기본값	리턴값	리턴 타입
learning_rate	float	opt_param['learn_rate']	SGD, Adagrad, Adadelta, RMSprop, Adam	array

03 연구방법

Grid search

■ Grid Search?

- 파라미터 선정에서 사용하는 딥러닝의 hyperparameter 최적화 기법
- Greedy Algorithm 방식으로 최고의 score를 탐색

■ GridsearchCV in keras

- Wrapper 성격의 클래스로 객체를 파라미터로 사용 가능
- 클래스 객체에 fit 메소드를 호출하여 다수의 모형을 생성하고 실행시켜 최적의 score값 반환
- Cross validation을 사용하여 Grid search의 정확도를 향상



03 연구방법

Grid search

- Learning rate : 각 Batch 끝의 가중치를 얼마나 갱신할 지 결정
- Activation Function : 각 뉴런의 비선형성 및 움직이는 시점 제어
- Dropout rate : Layer에 포함된 Weight 중 얼마만큼의 부분을 계산에 참여시킬지 결정
- Weight Constraint : Dense에 부과하는 가중치로 보통 Dropout rate와 같이 사용
- Neurons : 입출력을 모두 연결하는 레이어로 뉴런 수 결정

Hyperparameters	매개변수
Learning rate	0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.1, 0.2, 0.3
Activation	Softmax, Softsign, Relu, Selu, Tanh, Sigmoid, Hard_sigmoid, Linear
Dropout rate	0.0, 0.2, 0.4, 0.6, 0.8
Dense layer	8, 16, 32, 64, 128, 256, 512
Weight constraint	0, 1, 2, 3, 4

03 연구방법

학습 파라미터

05 Convolution 레이어 출력 데이터 크기 산정

$$\text{Output Height} = OH = \frac{(H + 2P - FH)}{S} + 1$$

$$\text{Output Weight} = OW = \frac{(W + 2P - FW)}{S} + 1$$

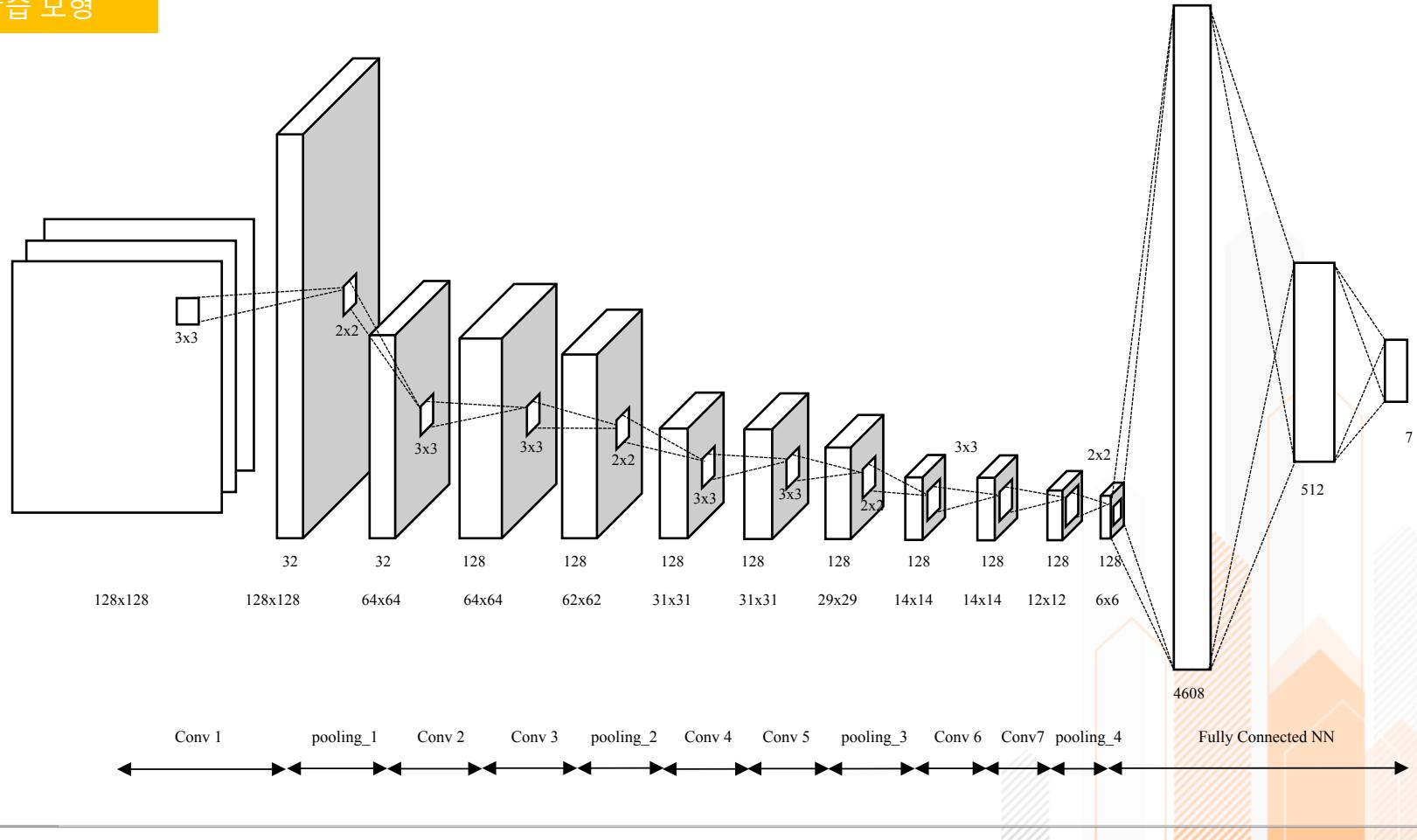
- H : 입력 데이터 높이
- W : 입력 데이터 폭
- 필터 높이 : FH
- 필터 폭 : FW
- S : Stride
- P : 패딩 크기

- Total number of parameters
: 3,139,207
- Total number of CNN training parameters
: 775,808
- Total number of FCNN training parameters
: 2,363,399

Layer(type)	Shape	Param
X.shape[1:] conv2d_1 : Conv2D (32,3,3) activation_1 : Activation max_pooling2d_1 : MaxPooling2D (2,2) dropout_1 : Dropout	input : (None, 128,128,3) output : (None, 128,128,32) output : (None, 128,128,32) output : (None, 64, 64, 32) output : (None, 64, 64, 32)	896 0 0 0 0
conv2d_2 : Conv2D (128,3,3) activation_2 : Activation conv2d_3 : Conv2D (128,3,3) max_pooling2d_2 : MaxPooling2D (2,2) dropout_2 : Dropout	output : (None, 64, 64, 128) output : (None, 64, 64, 128) output : (None, 62, 62, 128) output : (None, 31, 31, 128) output : (None, 31, 31, 128)	36992 0 147584 0 0
conv2d_4 : Conv2D (128,3,3) activation_3 : Activation conv2d_5 : Conv2D (128,3,3) max_pooling2d_3 : MaxPooling2D (2,2) dropout_3 : Dropout	output : (None, 31, 31, 128) output : (None, 31, 31, 128) output : (None, 29, 29, 128) output : (None, 14, 14, 128) output : (None, 14, 14, 128)	147584 0 147584 0 0
conv2d_6 : Conv2D (128,3,3) activation_4 : Activation conv2d_7 : Conv2D (128,3,3) max_pooling2d_4 : MaxPooling2D (2,2) dropout_4 : Dropout	output : (None, 14, 14, 128) output : (None, 14, 14, 128) output : (None, 12, 12, 128) output : (None, 6, 6, 128) output : (None, 6, 6, 128)	147584 0 147584 0 0
flatten_1 : Flatten dense_1 : Dense activation_5 : Activation dropout_5 : Dropout dense_1 : Dense activation_6 : Activation	output : (None, 4608) output : (None, 512) output : (None, 512) output : (None, 512) output : (None, 7) output : (None, 7)	2359808 0 0 0 3591 0

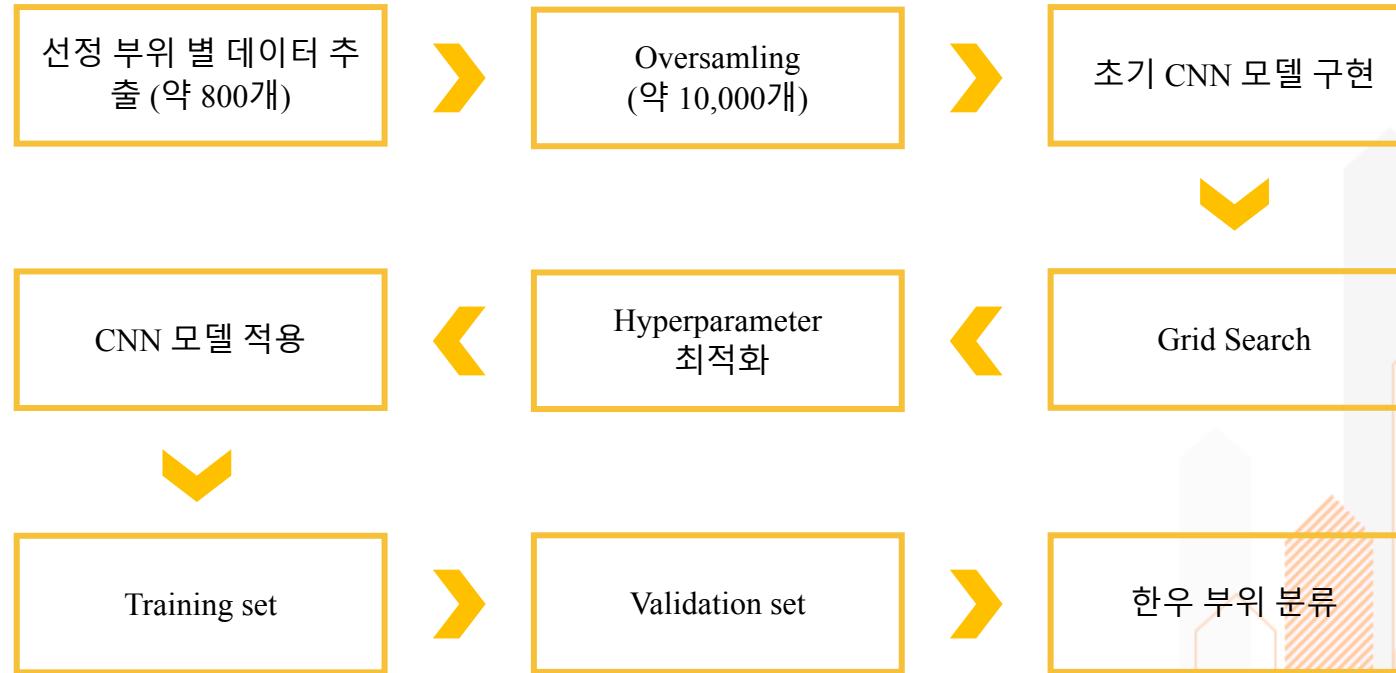
03 연구방법

학습 모형



03 연구방법

모형 프로세스



03 연구방법

파라미터 최적화 I

Optimizer

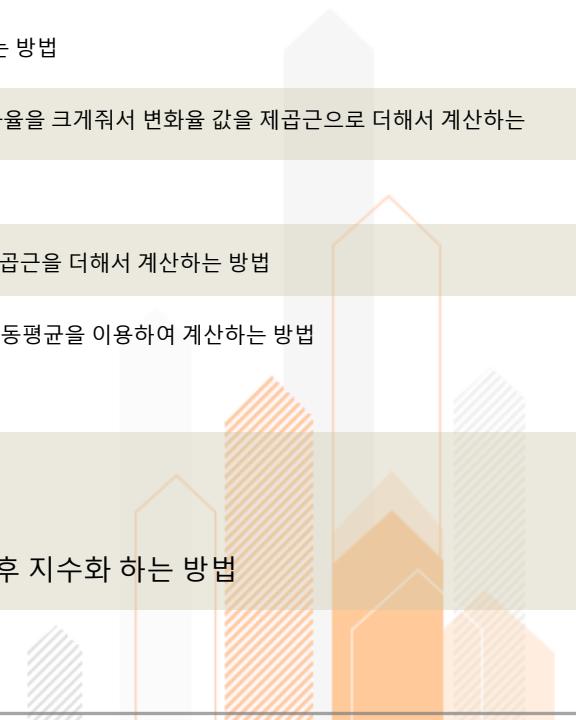
01 각 Optimizer의 특징

- 
- 01 SGD 경사하강법, 소량의 데이터에 대한 loss를 계산해서 값을 찾아내는 방법
 - 02 Adagrad 자주 변화하는 값은 변화율을 적게 주고, 크게 변화하는 값은 변화율을 크게 줘서 변화율 값을 제곱근으로 더해서 계산하는 방법
 - 03 RMSprop Adagrad에서 지수이동평균을 이용하여 계산하는 방법
 - 04 Adadelta RMSprop와 비슷한 방법으로 Stepsize에 따른 지수이동평균의 제곱근을 더해서 계산하는 방법
 - 05 Adam RMSprop 기법과 경사하강법의 Momentum 방식을 합쳐서 지수이동평균을 이용하여 계산하는 방법

Momentum : 가중치 값으로 기존 방향값을 적용해서 더하는 방법

Step size : 각각 값을 계산할 때의 적용되는 size값

지수이동평균 : 각 변수들의 중요도에 따라 비율을 정해 이동평균을 구한 후 지수화 하는 방법



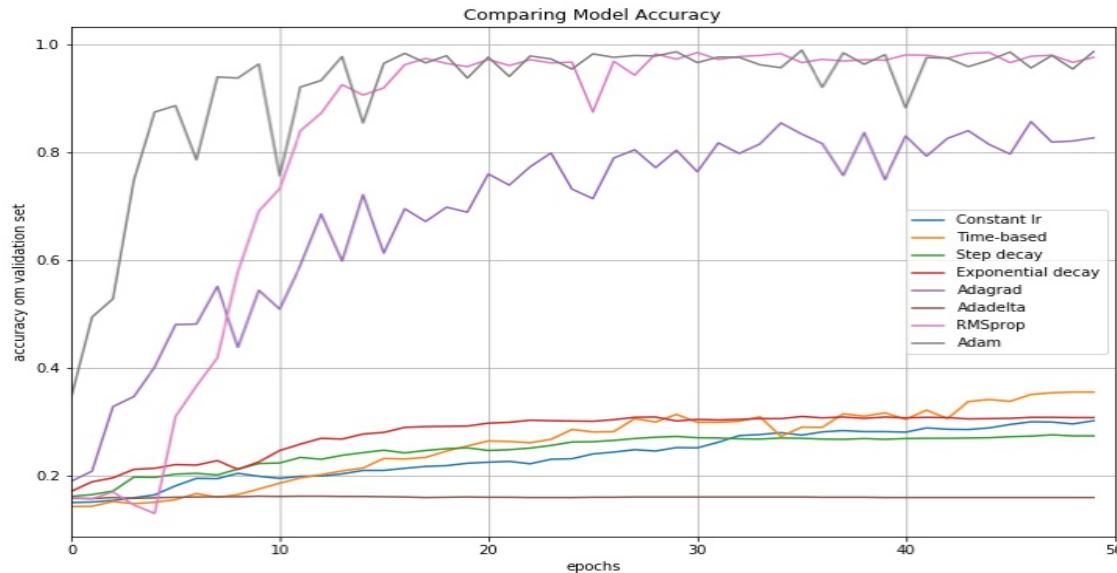
03 연구방법

파라미터 최적화 I

Optimizer

02 Optimizer 설정

Learning_Rate를 0.001로 설정하여 CNN모델에 적용 후 Accuracy가 가장 높은 Optimizer 설정

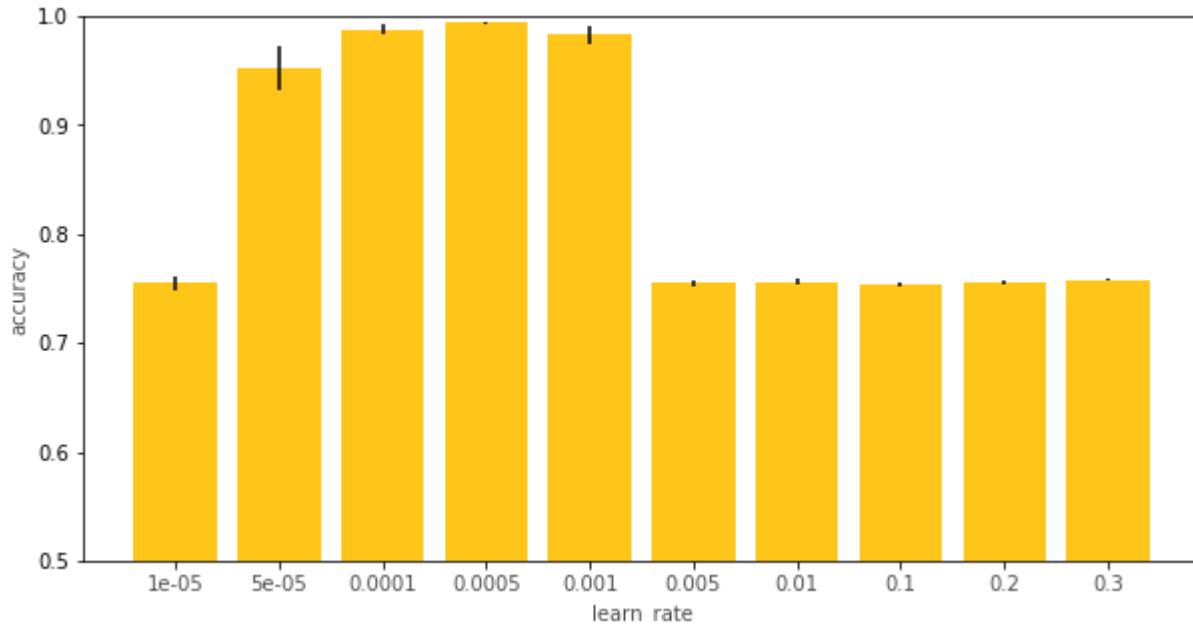


안정적으로 높게 계속 산출되는 RMSprop로 설정

03 연구방법

파라미터 최적화 II

Learning Rate

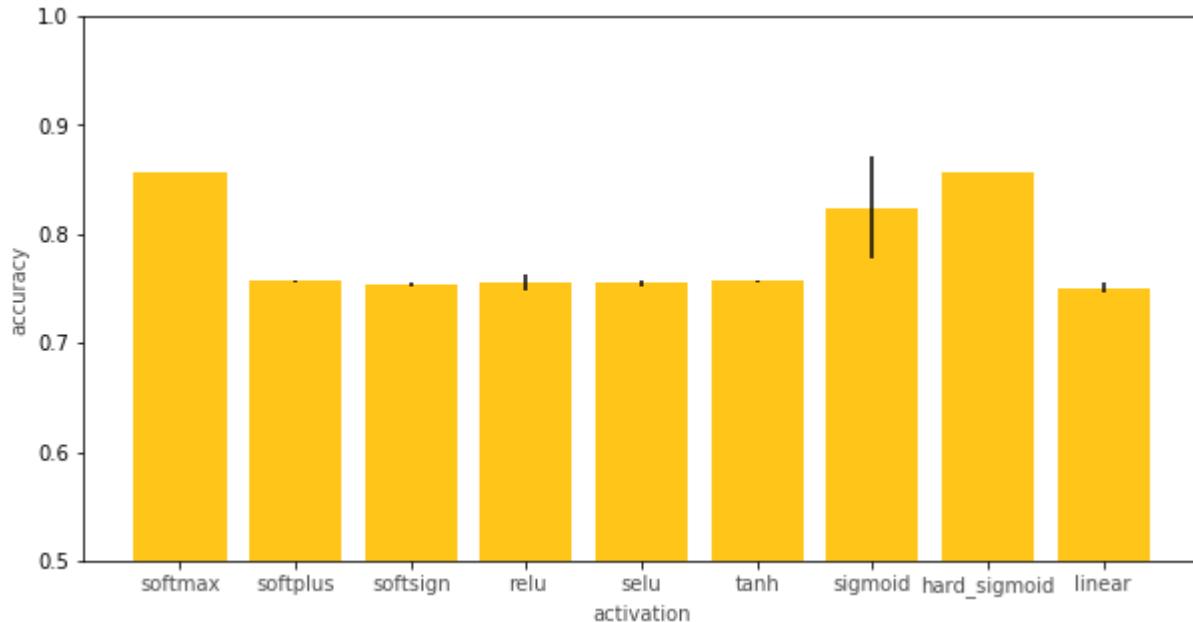


Best Score
: 0.0005

03 연구방법

파라미터 최적화 III

Activation function

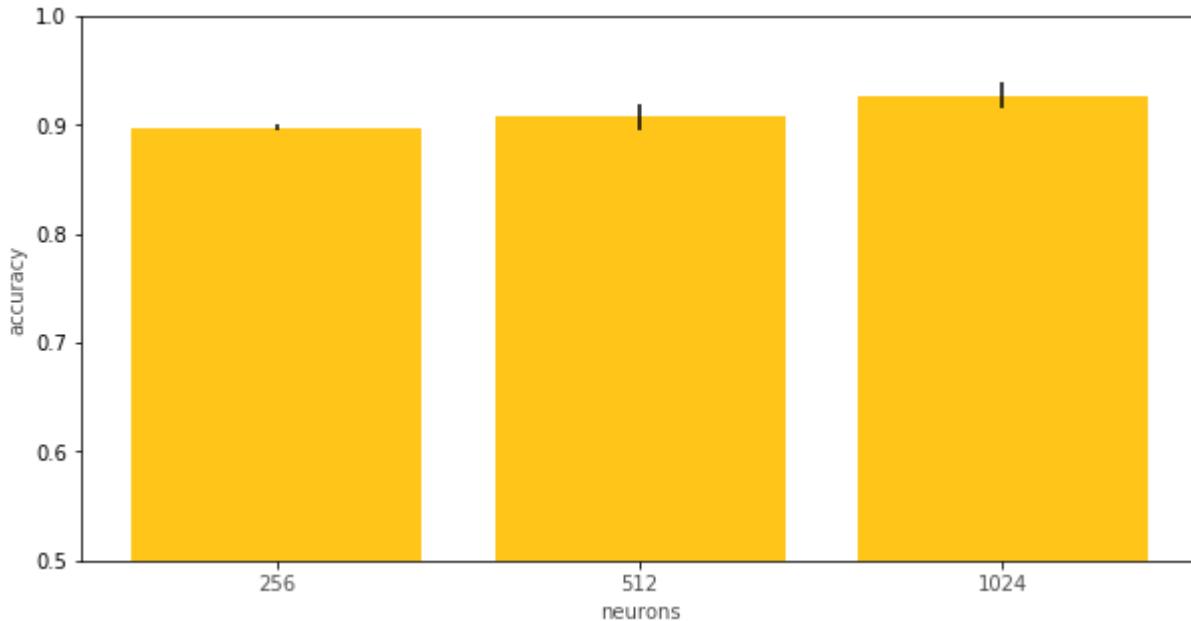


**Best Score
: softmax**

03 연구방법

파라미터 최적화 V

Number of Neurons



Best Score
: 1024

04 분석결과

분석 결과

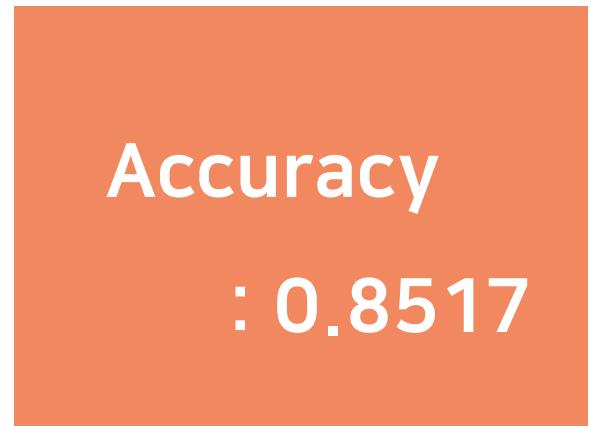
- Batch size = 100
- Epoch = 50
- Optimizer = RMSprop
- Activation = softmax
- Dropout rate = 0.5
- Weight constraint = 3
- Neurons = 1024



04 분석결과

분석 결과

- Batch size = 100
- Epoch = 50
- Optimizer = RMSprop
- Activation = softmax
- Dropout rate = 0.5
- Weight constraint = 3
- Neurons = 1024



05 결과 및 개선점

결과 및 개선점

01 결과

- 트레이닝 데이터 셋에 따라 정확도의 편차가 심함
: 정확한 데이터 품질과 양의 확보가 중요함
- Parameter에 따른 정확도의 편차가 있음
: 데이터 셋에 적합한 파라미터가 존재함
- 특정 분류(안심)에 대한 예측률이 떨어짐
: 추가적인 알고리즘 개발이 필요함
- CNN과 NN의 파마리터 개수가 10배 이상의 차이 발생
: CNN이 비교적 적은 파라미터로 높은 성과를 보임

02 개선점

- Parameter 범위를 균등 분포에서 추출하여 더 많은 변수 학습 필요
- 병렬처리로 성능 향상
- 최적화 모형을 통한 어플리케이션 활용

