**3조 빵빵하게**

**한세 비트 고급 32기 3조**

**[빵빵하게]**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Picture** | **Name** | **Student Number** | **Major** | **E-mail** |
|  | 정태준 | 201455044 | 전자소프트웨어  학과 | rnsemfa@ naver.com |
|  | 권오셈 | 201455003 | ICT 디바이스  학과 | dhtpa123@ naver.com |
|  | 신민식 | 201455025 | ICT 디바이스  학과 | alstlreowk@naver.com |
|  | 김선일 | 201710655 | ICT 디바이스  학과 | jjyk0676@ naver.com |
|  | 박혜미 | 201710654 | 전자소프트웨어  학과 | miugxc31@ naver.com |

**목차**

**1절 프로젝트개요**

**1.1 개발 배경 및 목적**

1.1.1 개요

1.1.2 설계 시 주안점

**1.2 추진체계 및 일정**

1.2.1 팀 구성

1.2.2 개발 일정

**1.3 부문별 수행업무 및 담당자**

**2절 관련 배경지식**

**2.1 딥러닝 머신러닝**

2.1.1 머신러닝의 개념

2.1.2 딥러닝의 개념

2.1.3 딥러닝의 구성

**2.2 영상인식**

2.2.1 OpenCV란?

2.2.2 OpenCV의 기능

**2.3 서버 클라이언트**

2.3.1 메이븐의 개념

2.3.2 netty.io란?

**3절 프로젝트 구성 및 구현**

**3.1 프로젝트 구상도**

**3.2 소프트웨어 설계 및 내용**

3.1.1 딥러닝

3.1.2 OpenCV

3.1.3 서버 / 클라이언트

**3.3 하드웨어**

**4절 결론 및 개선방안**

**4.1 결론 및 기대효과**

4.1.1 결과

4.1.2 기대효과

**4.2 문제점 및 개선 방안**

4.2.1 문제점

4.2.2 개선 방안

**5절 참고문헌 및 사이트**

**1절 프로젝트 개요**

**1.1 개발 배경 및 목적**

**1.1.1 개요**

비대면 서비스가 각광받으며 시장 또한 점차적으로 확대되고 있다. 대표적인 비대면 서비스로 무인 계산대가 각광받는다. 시장조사업체에 따르면 세계 무인 계산대 시장규모가 734억 달러에 이를 것으로 전망되었다. 소비자도 또한 트렌드 모니터의 조사에 따르면 점원 도움이 없는 쇼핑 선호도를 묻는 질문에 전체의 85.9%가 선호한다고 답했다. 무인 계산대 이용 경험으로는 편리함(42.7%), 빠른 이용속도(41.4%), 쉬운 조작방법(40.7%)들이 주된 이유다. 그렇지만 무인 계산대가 잘 이용되지 않는 곳이 있다. 바로 동네 빵집이다. 바코드가 없는 빵의 특징상 점원이 직접 빵을 분류하여 계산 포스기에 입력하고 계산하는 방식이다. 그렇기에 빵을 구분하고 계산을 도와주는 무인 계산대를 만든다면 직원이 일일이 계산하는 것보다 빠르고 편리하게 계산할 수 있을것이다. 영상인식을 통하여 빵을 인식하고 분류하여 계산 포스기로 전송한다면 속도와 시간이 많이 감축되어 도움이 될 것이다.

**1.1.2 설계 시 주안점**

핵심은 점원의 도움이 없이 빵을 계산할 수 있어야 하는 것이다. 빵을 인식하고 계산으로 유도하는 과정에 중점을 두었다. 빵들을 구분하고 분류하는 것이 가장 중요한 점이므로 빵을 하나하나의 객체로 인식해야 한다. 또한 빵을 잘못 판단하는 경우가 없게 확실한 학습이 필요하며 빵의 종류에 대한 정보를 포함시켰다. 속도가 생명이므로 속도를 올리기 위해 주안점을 두었다.

**1.2 추진체계 및 일정**

**1.2.1 팀 구성**

조장: 정태준

조원: 김선일, 권오셈, 박혜미, 신민식

**1.2.2 개발 일정**

개발 기간: 2019년 11월 ~ 2020년 1월 (약 3개월)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **작업 내용** | **과제 수행 기간 (주)** | | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9~11** |
| **주제 선정** |  |  |  |  |  |  |  |  |  |
| **역할 분담 및 공부** |  |  |  |  |  |  |  |  |  |
| **소프트웨어 및 하드웨어 설치** |  |  |  |  |  |  |  |  |  |
| **딥러닝 학습코드 작성** |  |  |  |  |  |  |  |  |  |
| **OpenCV 코드 작성** |  |  |  |  |  |  |  |  |  |
| **수정 및 마무리** |  |  |  |  |  |  |  |  |  |

**[그림 1 개발 일정 표]**

**1.3 부문별 수행업무 및 담당자**

|  |  |  |
| --- | --- | --- |
| **담당업무** | **사용언어 / 툴** | **담당자** |
| 딥러닝 / 머신러닝 | python, tensorflow, keras | 정태준, 권오셈 |
| OpenCV를 이용한 영상처리 | python, | 박혜미, 권오셈, 신민식 |
| 서버 / 클라이언트 | java, android studio, python | 김선일, 정태준 |
| 하드웨어 / 주변장치 | web cam, 블록 | 신민식, 김선일, 박혜미 |

**[그림2 업무 분담표]**

**2절 관련 배경지식**

**2.1 딥러닝 머신러닝**

**2.1.2 머신러닝의 개념**

**1. 머신러닝**

머신러닝(machine learning)이란 인공지능 연구 과제 중의 하나로, 인간의 뇌가 자연스럽게 수행하는 학습이라는 능력을 컴퓨터로 구현하는 방법이다. 수많은 데이터를 학습시키고 패턴을 찾아내게 한 후 찾아낸 패턴은 기반으로 다른 데이터를 분류하거나 미래를 예측하는 것이다. 최근에는 이러한 머신러닝을 활용해 문자 인식, 음성 인식, 바둑 또는 장기 등의 게임 전략, 의료 진단, 로봇 개발 등이 이루어 지고 있다. 머신러닝이 주목받고 있는 이유는 이처럼 다양한 분야에서 활용될 수 있기 때문이다.

**2. 머신러닝의 종류**

머신러닝에는 여러 가지 종류가 있다. 크게 “교사 학습”, “비교사 학습”, “강화 학습”으로 나눌 수 있다. 교사 학습은 문제를 주고 답을 알려주는것과 같은 방식으로 데이터와 답을 함께 입력하여 학습시키는 방식이다. 비교사 학습은 반대로 데이터만 입력하고 답을 알려주지 않는다. 일반적으로 사람도 제대로 알 수 없는 본질적인 구조 등을 확인하기 위해 사용된다. 강화 학습은 교사 학습과 비슷하지만 완전한 답을 제공하지 않는다. 강화 학습에서는 환경이 변화하면서 행동의 주체가 어떤 보상을 받게 되고 이를 기반으로 행동의 주체는 더 많은 보상을 얻는 방향으로 행동을 학습하게 된다. 각각의 학습은 원하는 목표에 따라 이용된다.

**3. 머신러닝의 응용분야**

머신러닝으로 할 수 있는 일은 크게 다섯가지이다.

**클래스분류** - 특정 데이터에 레이블을 붙여 분류한다. 스팸 메일 분류, 필기 인식 등.

**클러스터링(그룹 나누기)** - 값의 유사성을 기반으로 데이터를 여러 그룹으로 나눈다. 사용자의 취향을 그룹으로 묶어 사용자에 맞는 광고를 제공하는 등.

**추천** - 특정 데이터를 기반으로 다른 데이터를 추천하는 것이다. 사용자가 인터넷 서점에서 구매한 책들을 기반으로 다른 책을 추천하는 것 등.

**회귀** - 과거의 데이터를 기반으로 미래의 데이터를 예측하는 것. 판매 예측, 주가변동 등등 예측하는 것 등.

**차원 축소** - 데이터의 특성을 유지하며 양을 줄이는 것이다. 데이터를 시각화 하거나 구조를 추출해서 용량을 줄여 계산을 빠르게 하거나 메모리는 절약하는 것 등.

**2.1.2 딥러닝의 개념**

**1. 딥러닝의 등장**

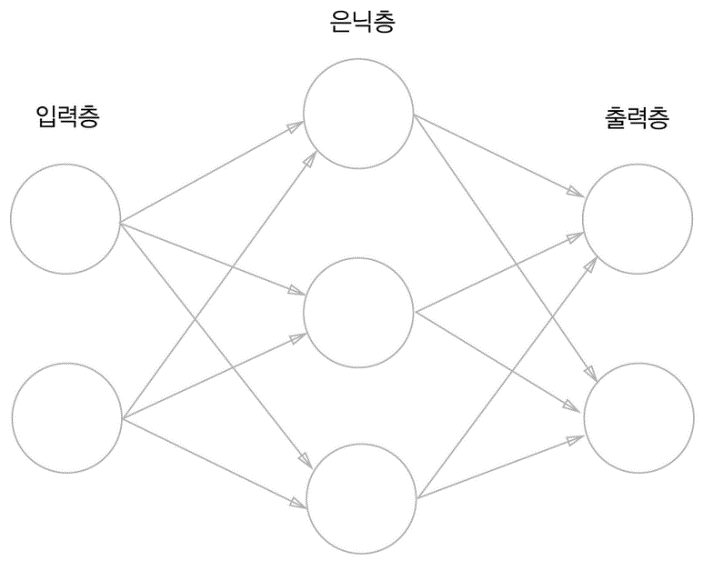
딥러닝(Deep Learning)은 머신러닝의 한 종류로서 여러 층을 가진 신경망(Neural Network)을 사용해 머신러닝을 수행하는 것을 의미한다. 최근 딥러닝이 굉장히 많은 주목을 받고 있다. 딥러닝이 주목받기 시작한 때는 2010년에 개최된 이미지 인식 대회 “ILSVRC(ImageNet Large Scale Visual Recognition Competition)”이후 라고 말한다. 2010년부터 매년 개최되고 있는 이 대회는 다양한 대규모 영상 인식/분류 소프트웨어가 출전하고 있다. 그 중 2012년에 등장한 딥러닝을 이용한 팀이 굉장한 성과를 보여주었고 그 후부터 딥러닝이 본격적으로 주목을 받고 많은 발전을 이루게 되었다. 딥러닝은 이미지 인식, 음성 인식 분야뿐 아니라 자연 언어 처리 등의 다양한 분야에 활용되며 큰 성과를 내고 있다. 딥러닝의 개념은 1980년대부터 있었지만 현대에 와서 컴퓨터의 성능이 좋아지며 비즈니스적으로 크게 성공하였기에 지금 같은 관심을 받게 된 것이다.

**2. 딥러닝과 기존의 이미지 인식과의 차이**

이미지 인식과 딥러닝의 결정적인 차이는 “특징량 추출” 이다. 기존의 방법은 사람이 이미지 내부에 있는 특징을 하나하나 지정해야 했다. 하지만 딥러닝에서는 학습 데이터에서 기계가 자동으로 특징을 추출하는 점이 장점이다. 그렇기에 딥러닝을 이용하면 사람의 수고가 없어지고 기계가 자동으로 특징을 학습한다.

**3. 신경망**

신경망(Neural Network)는 인간의 신경망을 본떠 만든 네트워크 구조를 의미한다. 컴퓨터에게 학습 능력을 갖게 해서 여러 가지 문제를 해결하기 위한 접근 방법이다. 인간의 뇌에는 수만은 신경 세포(뉴런)이 있고 하나의 뉴런은 다른 뉴런에게서 신호를 받고, 다른 뉴런에게 신호를 전달하는 역할을 맡고 있다. 뇌는 이러한 신호의 흐름을 기반으로 다양한 정보를 만들어 내고 이를 컴퓨터로 구현한 것이 바로 신경망이다.

[](https://blog.naver.com/PostView.nhn?blogId=lego7407&logNo=221680907171&parentCategoryNo=&categoryNo=92&viewDate=&isShowPopularPosts=true&from=search)

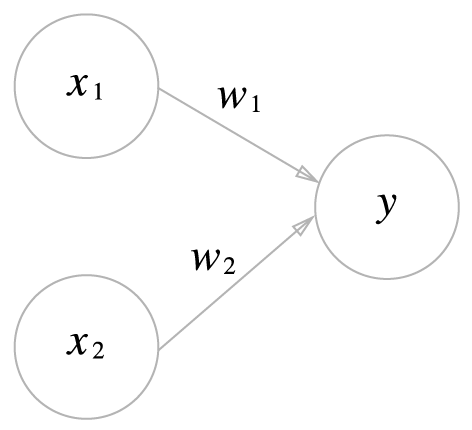
**[그림3 신경망]**

위의 그림에서 입력층에 학습시키고 싶은 데이터를 입력하고 데이터들이 입력층, 중간층(은닉층), 출력층을 지나며 처리가 일어나고 최정적인 결과가 출력된다. 뉴런을 본뜬 유닛은 여러 입력을 받지만 이러한 입력에 따라 하나의 다른 유닛으로만 출력을 보낸다. 이러한 신경망을 3개 이상 중첩하면 깊은 신경망(Deep Neural Network: DNN)이라고 부르는데 이를 활용한 기계 학습이 딥러닝이다. 딥러닝은 대량의 데이터를 입력해서 학습시키고, 학습이란 이러한 뉴런의 접속 가중치를 조정하는것이다.

**2.1.3 딥러닝의 구성**

**1. 퍼셉트론**

퍼셉트론(Perceptron)은 딥러닝의 기원이 되는 알고리즘으로 1957년에 프랑크 로젠블란트가 고안한 것으로서, 다수의 신호를 입력 받아 하나의 신호를 출력한다. 신호가 흐를 때는 1, 신호가 흐르지 않을 때는 0을 출력한다. 비교적 단순한 구조를 가지고 있지만 현재 기계학습의 기초가 되는 중요한 개념이다.

[](https://blog.naver.com/PostView.nhn?blogId=lego7407&logNo=221680894247&parentCategoryNo=&categoryNo=92&viewDate=&isShowPopularPosts=true&from=search) 그림은 2개의 입력 신호를 받은 퍼셉트론의 예시이다.

​

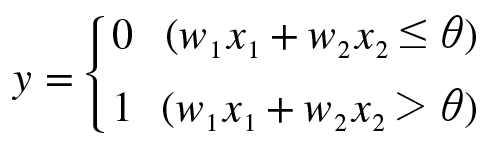
1. x1, x2: 입력신호(입력값)

2. w1, w2: 가중치

3. y: 출력값

4. 원: 노드 혹은 뉴런이라 부름

**[그림4 퍼셉트론]**

[](https://blog.naver.com/PostView.nhn?blogId=lego7407&logNo=221680894247&parentCategoryNo=&categoryNo=92&viewDate=&isShowPopularPosts=true&from=search)

- 입력 신호가 뉴런에 보내질 때 각각 고유한 가중치가 곱해진다.

- 뉴런에서 보내온 신호의 총합이 정해진 한계를 넘어설 때만 1을 출력한다.

- 그 한계를 임계 값이라고 표현하고, 세타로 표시한다.

- 가중치가 클수록 해당 신호가 그만큼 더 중요하다는 뜻이다.

​

**2. 활성화 함수**

입력 신호의 총합을 출력 신호로 변환하는 함수를 활성화 함수라고 한다. 신경망에서 입력받은 데이터를 다음 층으로 출력할지를 활성화 함수가 결정한다. 활성화함수 종류는 여러가지가 있다. 부드러운 곡선인 시그모이드(Sigmoid) 함수, 계단형태의 계단함수, 입력을 그대로 출력하는 항등함수, 0이하일때는 0을 출력하고 0보다 크면 값을 그대로 출력하는 ReLU(Rectified Linear Unit)함수, 지수함수를 이용해 클래스 분류 문제를 풀 때 점수 벡터를 클래스 별 확률로 변환하기 위해 사용되는 소프트맥스(Softmax)함수가 있다.

**3. 손실 함수**

신경망 학습에서는 현재의 상태를 하나의 지표로 표현할 수 있다. 그리고 그 지표를 가장 좋게 만들어주는 가중치 매개변수의 값을 탐색할 수 있다. 신경망도 하나의 지표를 기준으로 최적의 매개변수 값을 탐색하게 된다. 신경망 학습에서 사용하는 지표는 손실함수라고 한다. 이 손실함수는 임의의 함수를 사용할 수도 있지만 일반적으로는 평균 제곱 오차와 교차 엔트로피 오차를 사용한다. 즉, 손실함수는 신경망 성능의 '나쁨'을 나타내는 지표로, 현재의 신경망이 훈련 데이터를 얼마나 잘 처리하지 못하느냐를 나타낸다.

**2.2 영상인식**

**2.2.1 OpenCV란?**

## OpenCV(Open Source Computer Vision)는 C/C++ 언어를 사용해서 실시간 영상처리를 목적으로 개발된 오픈소스 라이브러리입니다. 또한 C++, C, Python 및 Java와 같은 다양한 인터페이스를 지원하며 Windows, Linux, Mac OS, iOS 및 Android같은 다양한 OS를 지원합니다.

* 1. OpenCV는 알고리즘 상으로 계산 효율성과 실시간 응용 프로그램에 중점을 두고 설계되었기 때문에 간단하게 OpenCV에서 제공되는 API를 사용하여 코딩하여도 실시간 프로세싱이 가능한 어플리케이션을 만들 수 있기 때문에 최적화나 알고리즘을 생각하지 않고도 품질 좋은 상용 프로그램을 만들 수 있습니다.
  2. 또한, OpenCV는 멀티 코어 프로세싱을 지원하기 때문에 다양한 상황에 응용이 가능합니다.

**2.2.2 OpenCV의 주 기능**

1. 윤곽선 검출 - 이미지안의 여러 사물에 대해 각각의 윤곽선을 만들어주고 구분해 줌.

2. 노이즈 제거 - 이미지 안의 잡음을 제거해준다.

3. 이미지 스티칭을 이용한 파노라믹 사진 제작 - 여러 사진들을 합쳐 하나의 자연스럽고 매끄러운 사진으로 변환시켜준다.

위에 기능보다 더 다양한 영상과 관련된 많은 프로세싱을 지원한다.

​

**2.3 서버 클라이언트**

**2.3.1 메이븐의 개념**

**1. 메이븐**

메이븐은 프로젝트 구조와 내용을 기술하는 선언적 접근방식의 오픈소스 빌드 툴이다.

컴파일과 동시에 빌드를 수행할 수 있으며 테스트를 병행하거나 서버측 디플로이 자원을 관리할 수 있는 환경을 제공한다 하지만 아무래도 개발자들에게 가장 큰 장점은 프로젝트의 종속 라이브러리들과 그 라이브러리에 영향을 미치는 Dependency자원까지 관리할수있다는 점 이다. 즉 jar파일을 다운받아 프로젝트에 추가할 경우 그것과 연관된 다른 종속 라이브러리 또한 다 찾아야 하는 불편함을 Maven을 통해서 일관성 있는 라이브러리간의 의존관계(의존성)관리를 할 수 있다는 점이다.

이는 단순히 라이브러리리 뿐 아니라 프로젝트별 모듈의 의존성 또한 관리가 된다는 뜻이기도 하다. 메이븐은 프로젝트 전반의 리소스 관리와 Configuration파일, Doc생성 및 이와 관련된 표준 디렉터리 구조를 처음부터 일관된 형태로 구성하여 진행하기 때문에 프로젝트 관리 및 배포 역할을 하는 다른 툴 들과의 연계에서도 뛰어난 유연성을 보여준다. 단점은, 버전별로 이클립스에서 구동되는 방식이 약간 호환성이 떨어진다는 점인데, 사용된 플러그인의 문제인지 메이븐 자체의 하위 호환성 문제인지는 모르겠다. 가끔 저장소 접근에 관한 문제도 발생한다고 알려져 있다. 메이븐을 학습하기 위해서는 메이븐을 설치하고 POM(Project Object Model)을 작성한 후 각종 빌드 스크립트 혹은 명령어를 통해적용할수있으며, 프레임워크 나 라이브러리도 쉽게 적용할수있다.

기본적인 메이븐의 주요 디렉토리이다.

src/main/java : 자바 소스 파일 위치 시킨다. 이 하위에 org.gliderwiki 와 같은 패키지를 배치한다.

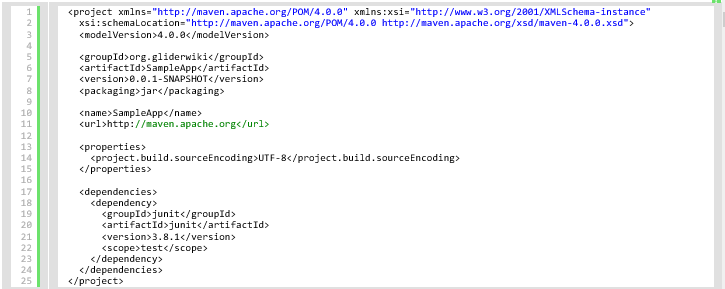
src/main/resources: 프로퍼티나 XML 등 리소스 파일이 위치한다.

src/main/webapp: Web Project 일 경우 WEB-INF등 웹 어플리케이션 리소스를 위치시킨다.

src/test/java: JUNIT등의 테스트 파일이 위치한다.

POM 파일의 기본구성이다.

STS상에서 MAVEN 프로젝트를 생성하거나 Spring Template로 프로젝트를 생성하게 되면 루트에 pom.xml이 위치한다. Pom 파일에는 프로젝트 정보가 표시되며, 이름, url, 이슈트레킹 시스템에 대한 명시, CI 서버 정도등 프로젝트에 필요한 시스템에 대한 명세가 가능하다.



**[그림5 pom.xml]**

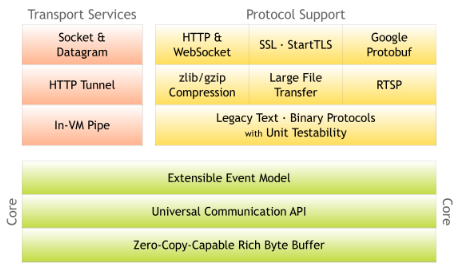
여기서 핵심은 dependencies항목이다. Dependencies 항목은 어플리케이션을 컴파일하고, 테스트하고, 실행하는데 필요한 라이브러리 목록을 기술한다. 기본적으로 생성된 메이픈 프로젝트는 Junit라이브러리가 test scope로 설정되어 있다. 바로 이처럼 dependency 설정을 통해 의존관계에 있는 라이브러리를 추가할 수 있다.

**2.3.2 netty.io란?**

**1. netty.io의 개념**

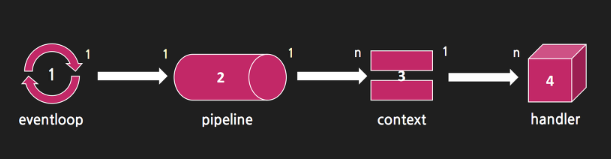
Netty는 프로토콜 서버 및 클라이언트와 같은 네트워크 어플리케이션을 빠르고 쉽게 개개발하는 것 가능하게 해주는 NIO클라이언트 서버 프레임워크이다. Netty는 TCP, UDP소켓 서버 개발과 같은 네트워크 프로그래밍을 매우 간단하고 능률적으로 만들어준다.

네티는 유지하기 쉬운 높은 성능의 프로토콜 서버 및 클라이언트를 신속한 개발을 위해 비동기 이벤트 드리븐 네트워크 어플리케이션 프레임워크 이다. 네티는 프로토콜 서버 및 클라이언트 같은 네트워크 어플리케이션의 빠르고 쉬운 개발을 가능하게 하는 NIO클라이언트 서버 프레임워크이다. 이는 TCP 또는 UDP소켓 서버와 같은 네트워크 프로그래밍을 간단하게 할 수 있다. 또한 자바 네트워크 어플리케이션 프레임워크이며, 비동기식 이벤트기반으로 운영되며 I/O처리를 위한 추상화 API를 제공한다. 이벤트 모델과 스레드 모델을 제공하며 유연한 이벤트 처리를 위해 bi-directional chain of responsibility patten을 사용하며, 직접 구현한 버퍼풀을 이용한 성능 이점을 제공한다.



**[그림6 netty 구성요소]**

핵심 컴포넌트에는 (eventloop, pipeline)이 있으며, 이벤트 루프에서 이벤트를 받아 핸들러에 전달하는 pipeline이 inbound event(이벤트 루프가 발생시킨 이벤트(소켓연결, 데이터 수신 등)을 작성한 inbound event handler에게 전달하며 outbound event사용자가 요청한 동작(쓰기, 읽기, 일시중단, 종료등)을 작성한 ounbound event handler에게 전달하며 최종적으로 이벤트 루프에 전달되어 I/O가 수행되도록 한다.



**[그림7 netty.IO 작동원리]**

**3절 프로젝트 구성 및 구현**

**3.1 프로젝트 구상도**

학습용 데이터 파일 생성

학습시킬 이미지 수집(data set)

학습을 바탕으로 예측 테스트

데이터 파일로 학습

인식한 빵을 이미지 파일로

저장

OpenCV를 이용해

빵을 인식

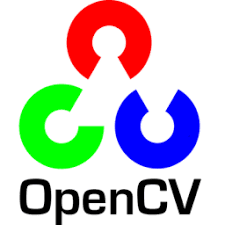
예측한 값 서버에 전달

저장된 이미지 파일을 이용해 예측

클라이언트에 출력

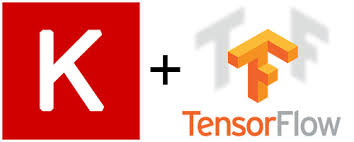
빵 가격 계산 후

클라이언트에 송신









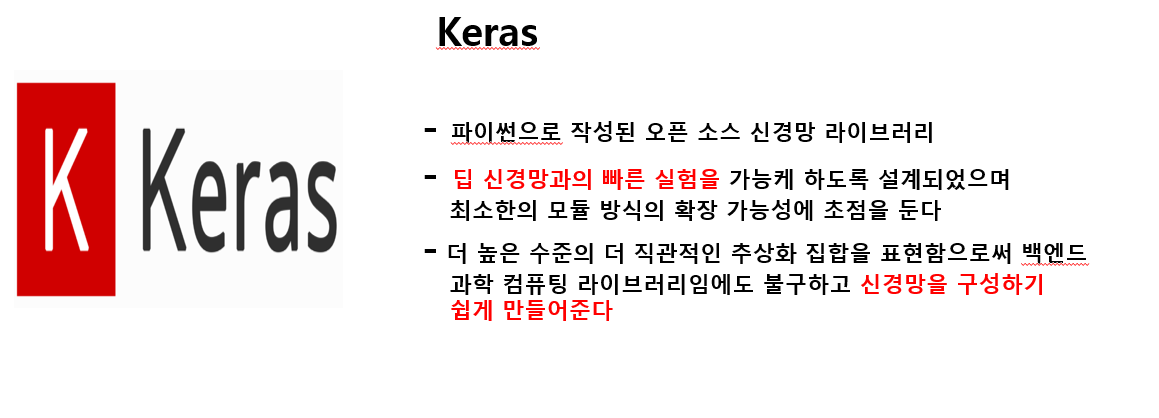
**3.2 소프트웨어 설계 및 내용**

**3.1.1 딥러닝**

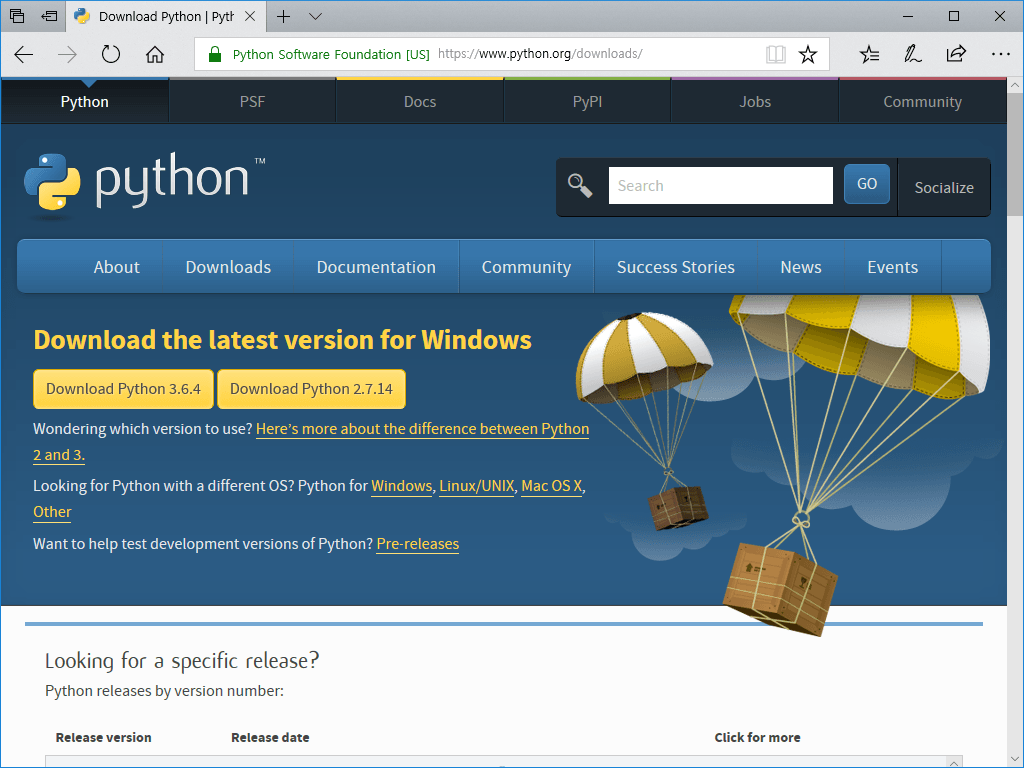
딥러닝 기술을 이용하기 위한 툴로 python을 설치해야 한다. 또한 python에서 추가적으로 사용할 라이브러리로 tensorflow 와 keras도 추가 설치해주어야 한다.



**[그림8 tensorflow]**

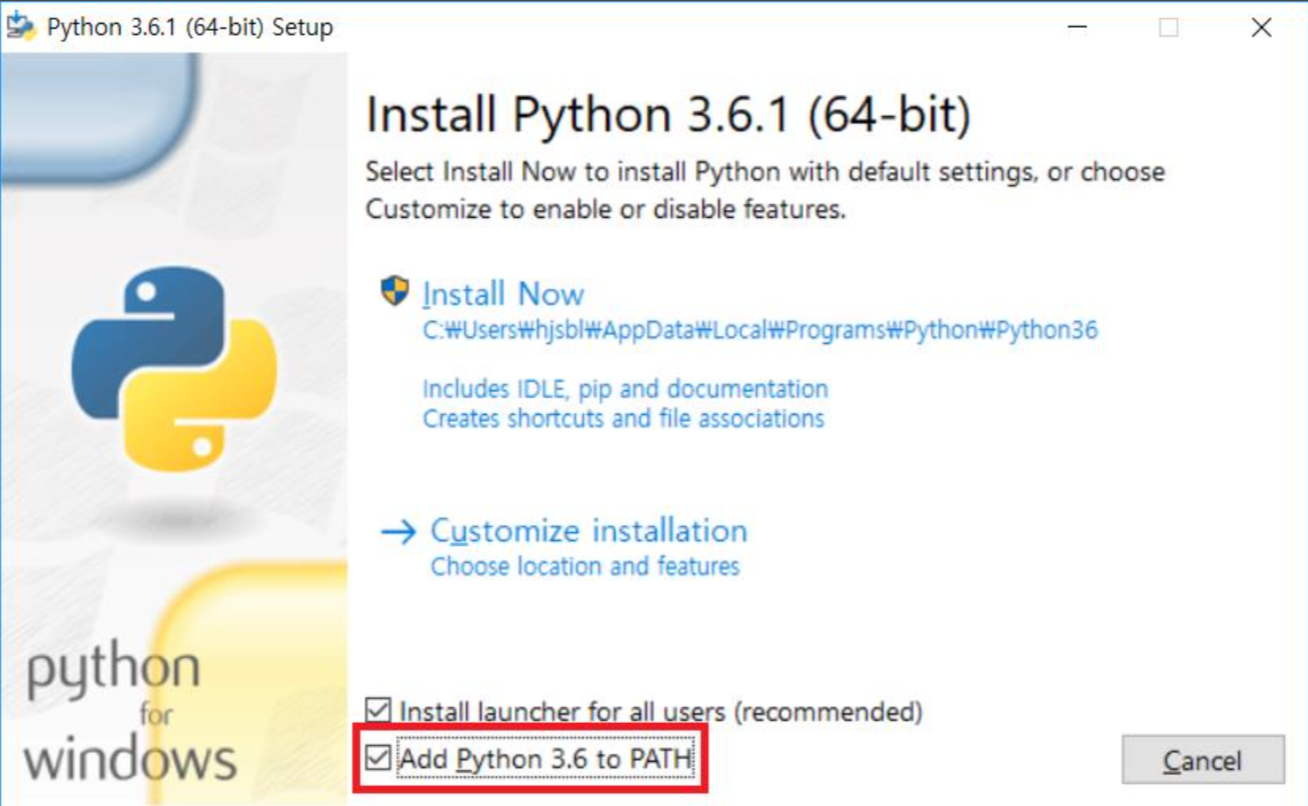
**[그림9 keras]**

우선 파이썬 공식 홈페이지의 다운로드 페이지([http://www.python.org/downloads](http://www.python.org/downloads/))에서 윈도우용 파이썬 언어 패키지를 다운로드한다. 다음 화면에서 Python 3.x로 시작하는 버전 중 3.6 version에 64-bit 이상의 버전으로 설치해야 한다.



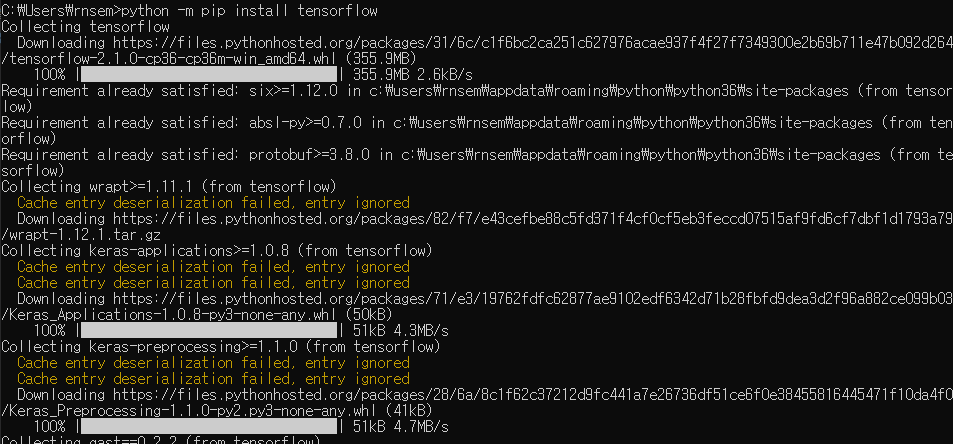
**[그림10 파이썬 다운로드]**

인스톨러를 실행시킨 후 아래의 체크박스를 체크해두고 설치한다.



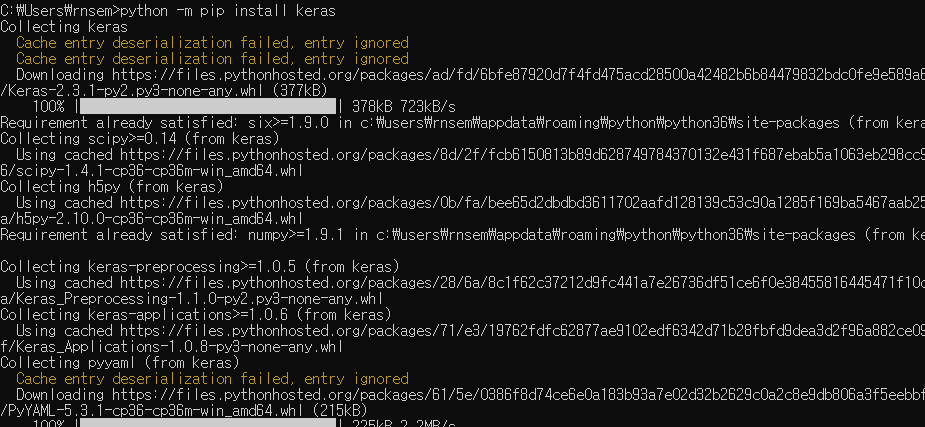
**[그림11 파이썬 설치]**

Tensorflow 설치를 위해 cmd창을 실행시키고 pip install tensorflow 명령어를 통해 설치한다.



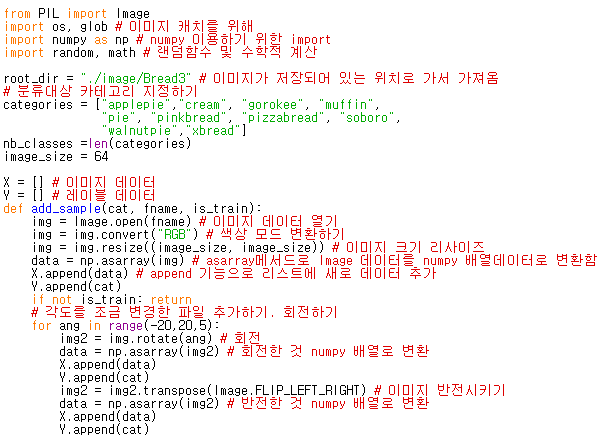
**[그림12 tensorflow 설치]**

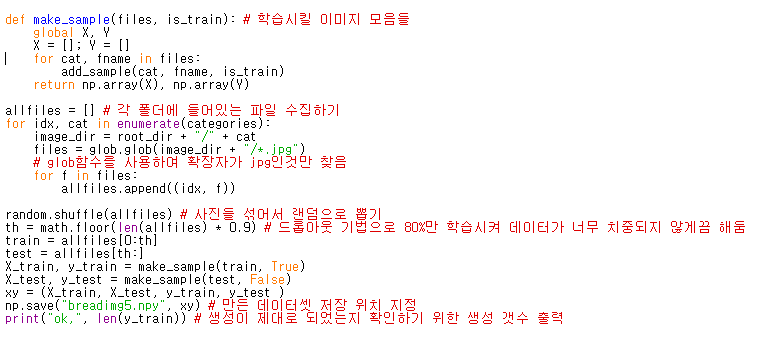
Keras 설치도 마찬가지로 cmd창에서 pip install keras 명령어로 설치가 가능하나 tensorflow 라이브러리를 먼저 설치해야만 가능하다.



**[그림13 keras 설치]**

이 코드는 python으로 작성된 이미지파일들을 데이터화 시키는 코드이다. 또한 이미지를 반전시키거나 회전시켜 부족한 이미지의 개수를 늘려주며 다양한 이미지를 만들어 정밀도를 높이는 파일이다.





훈련 데이터와 따로 구분해둔 더미 데이터들(테스트 데이터로 표현함)을 다 같이 반전하거나 회전하여 숫자를 늘린 후 섞어서 구분하면 훈련데이터와 테스트 데이터들이 같은 경우가 많이 발생하여 정답율이 굉장히 높아지는 오류가 발생할 수 있다.

그렇기에 드롭아웃(drop out) 기법을 이용하는것과 동시에 훈련 데이터만 구분하여 반전 및 회전시켜 정밀도를 높임으로써 훈련 데이터와 나중에 테스트할 데이터들에 차이점을 두어 성능을 향상시켰다.

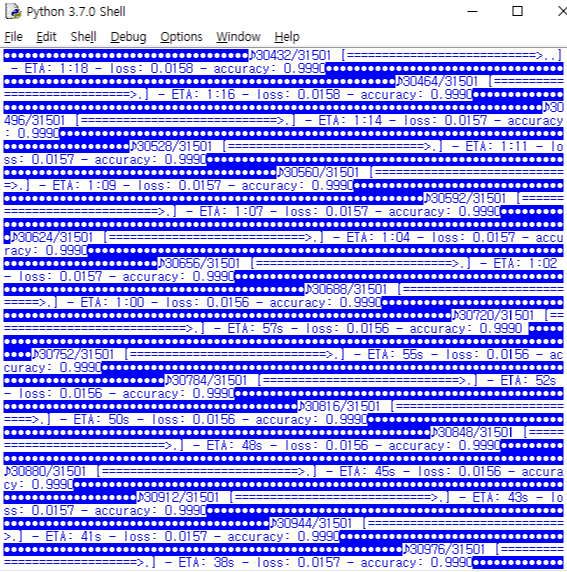
코드를 통해 이미지파일을 데이터셋으로 만들어 둔 npy파일이다.



**[그림14 데이터셋 파일]**

만들어 둔 데이터셋 파일을 기반으로 딥러닝, 학습을 시작한다.



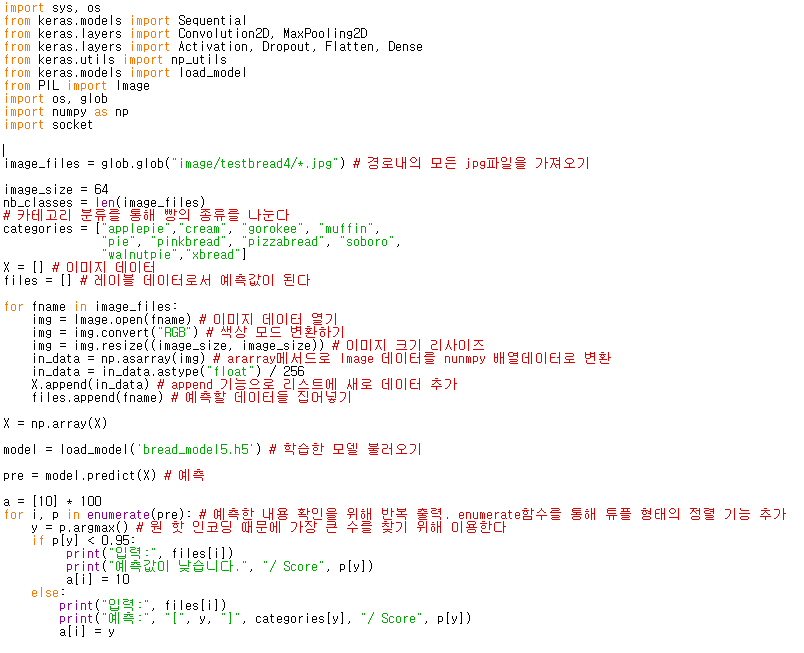
코드를 실행하면 학습한다. 다음 사진은 학습중인 모습이다.

**[그림15 딥러닝 학습 중]**

학습하고 나면 학습된 데이터의 내용이 저장되어 있는 h5파일이 만들어진다.

**[그림16 h5파일]**

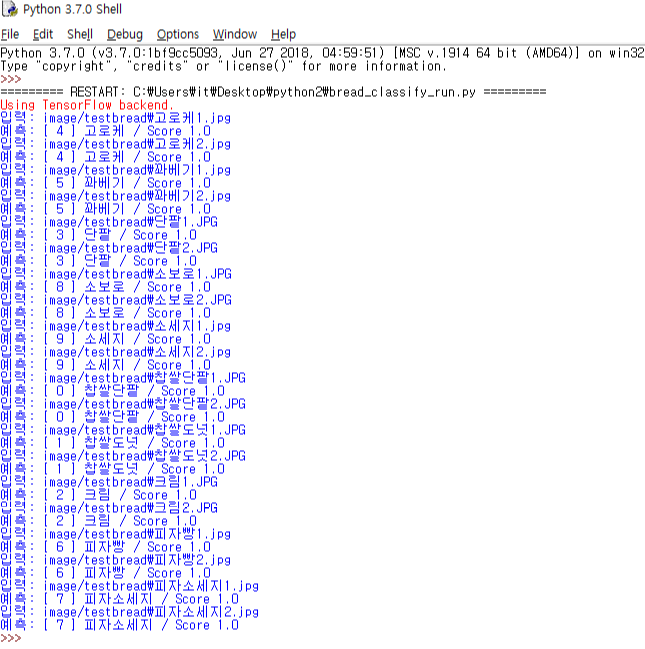
마지막으로 만들어 둔 h5파일을 기반으로 분류해내는 딥러닝 코드를 작성하면 된다.



glob함수를 이용해 지정한 위치에 있는 jpg파일들을 전부 가져오고, 가져온 파일들과 지정해둔 모델인 h5파일을 비교 분석한다. 미리 카테고리를 정해주어서 분류될 파일들의 종류를 지정해주고 만들어 둔 배열에 결과 값을 순서대로 넣어준다. 예측 후 정확도를 출력하는데, 95%보다 낮은 정확도는 “예측 값이 낮습니다” 라는 문구만 출력할 뿐, 지정된 카테고리에 해당되는 값은 출력되지 않는다. 95%이상의 정확도의 결과값은 그 예측 카테고리와 함께 출력된다.



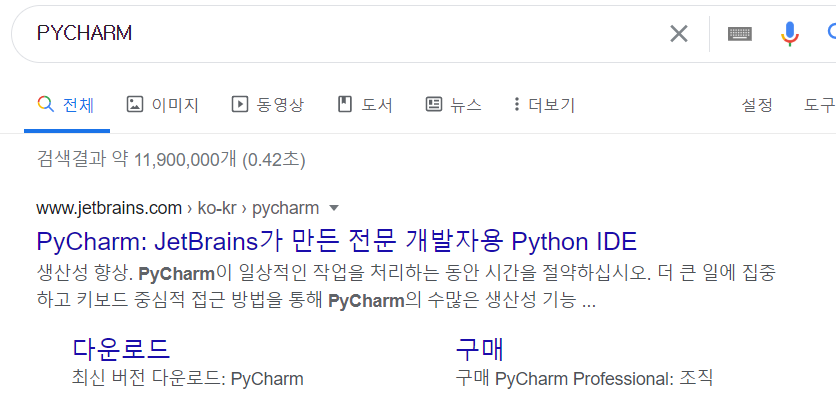
**[그림17 예측 샘플 자료]**

****

**[그림18 샘플의 예측 결과]**

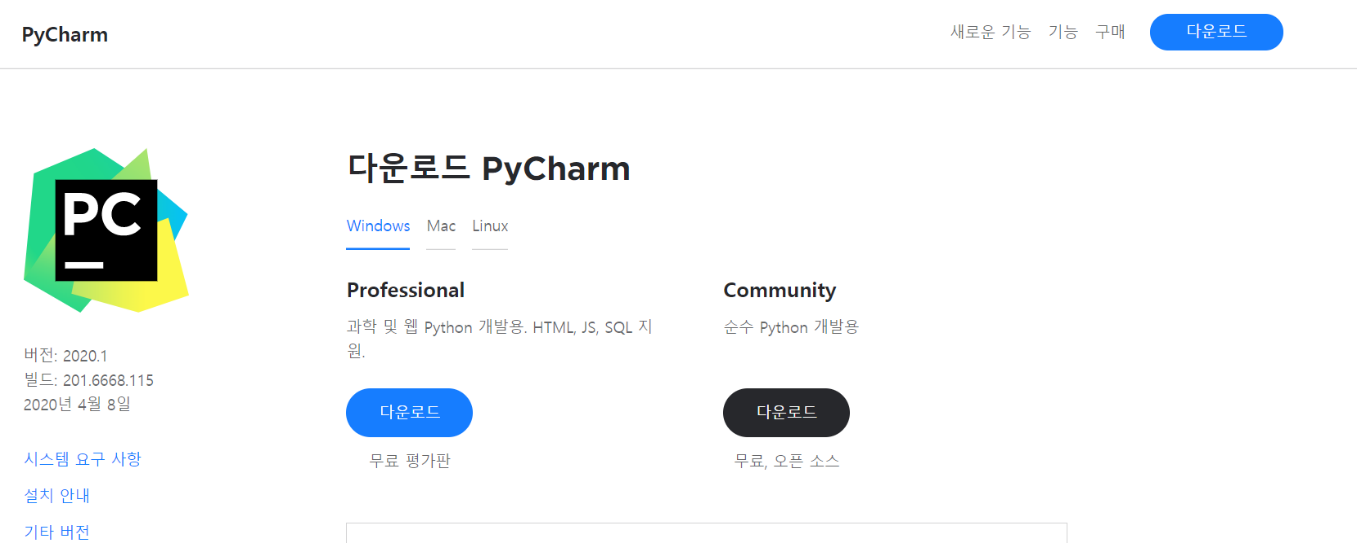
**3.1.2 OpenCV**

빵을 인식하기 위해서 영상 처리를 위한 OpenCV 사용하기 위해서는 Python이 필요하다.

파이썬 통합개발환경인 PyCharm을 사용하여 진행하였다. 다음은 PyCharm 설치를 하는 방법이다.

**[그림19 파이참 다운로드]**

링크를 눌러서 홈페이지에 들어간 후 다운로드를 한다.



**[그림20 파이썬 다운로드2]**

다운로드가 끝난 후, 빵을 cam으로 찍은 후 분리해서 저장할 bread 폴더를 만든다.

path='bread' 로 bread 폴더로 경로를 지정해 주었다. 사진의 위치와 정해 둔 경로가 다르면 오류가 생기므로 주의한다.

시작하기 전에 빵을 찍는 환경을 구상한후 미리 setting하여 동일한 조건에서 이미지를 제작하였다. 빵을 내려 놓는 바닥 부분 즉, 기준 색상이 되는 하얀 바닥 사진을 미리 찍어 코드에 집어넣어 빵을 찍은 사진을 하얀 배경 사진의 RGB값과 비교하여 코드를 구현했다.

이 함수는 사진을 새로 찍을 때마다 폴더가 비어져 있어야 하기 때문에 코드 처음 실행 부분에 bread 폴더의 내용을 제거해주는 함수를 넣었다.

|  |
| --- |
| def removeExtensionFile(filePath,fileExtension):  if os.path.exists(filePath):  for file in os.scandir(filePath):  if file.name.endswith(fileExtension):  os.remove(file.path)  return 'Remove File:'+fileExtension  else:  return 'Directory Not Found' print(removeExtensionFile('C:\bread','jpg')) |

def Summing(img,row,col):  
 sum = 0  
 for i in range(0,3):  
 sum += img[row][col][i]  
  
  
 return sum  
  
  
def UpChange(ori,new):  
 if ori <= new:  
 return ori, new  
 else:  
 return new, ori  
-------------------------------------------

for i in range(0,rows):  
 j=0  
 while(1):  
 k=0  
 while(1):  
 ori=Summing(mean,i,j)  
 k=k+1  
 if k == 3:  
 break  
 if new == 0:  
 new = ori  
 ori,new = UpChange(ori, new)  
 new = ori  
 j=j+1  
 if j == cols:  
 break

위 소스코드는 RGB 최대값을 찾아내서 더 작은 값 이랑 바꾸는 기능이다. (255,255,255)면 흰색이므로 흰색에 가까운 값을 찾는 과정이다.

전체 소스코드이다. 함수의 기능은 주석으로 설명하였다. 전체 코드의 흐름은 다음과 같다.

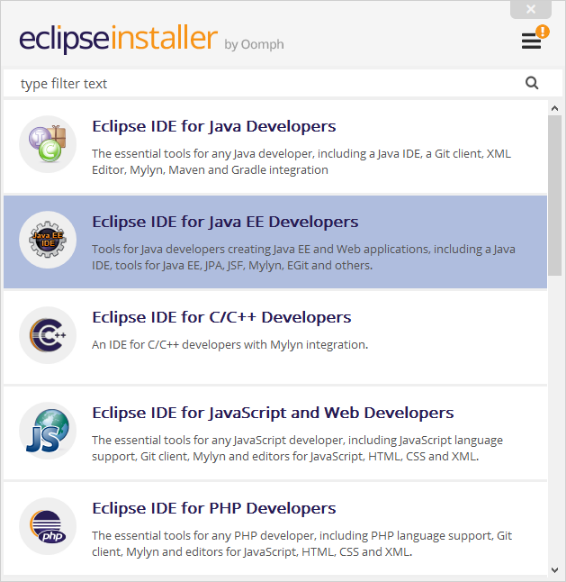
저장 폴더의 파일들을 삭제하여 폴더를 비운다. 그 후 설치해둔 cam을 이용해 캡쳐 기능으로 원하는 위치의 사진을 찍는다. 기존에 찍어 둔 배경 이미지를 불러와 배경 이미지의 RGB값을 분석한다. cam으로 찍은 이미지를 불러온 후 바이너리 이미지로 변환한다. 확실한 배경과 확실한 전경을 구분하는 단계를 거친다. 구분된 전경을 레이블링, markers, watershed 기능으로 하나씩 분리하여 사진으로 만든다. 분리한 bread 이미지를 특정 폴더에 저장한다. 딥러닝 단계에서 저장해둔 폴더에 접근해 이미지를 이용한다. 그 후 프로그램이 다시 이용될 때 처음에 있는 폴더안의 파일들을 삭제하는 기능으로 그 전의 결과를 전부 제거한 후 새로 저장한다.

import cv2  
import numpy as np  
import os  
#폴더 내 특정 확장자 파일 삭제 함수  
def removeExtensionFile(filePath,fileExtension):  
 if os.path.exists(filePath):  
 for file in os.scandir(filePath):  
 if file.name.endswith(fileExtension):  
 os.remove(file.path)  
 return 'Remove File:'+fileExtension  
 else:  
 return 'Directory Not Found'  
print(removeExtensionFile('C:\bread','jpg'))

def Summing(img,row,col):  
 sum = 0  
 for i in range(0,3):  
 sum += img[row][col][i]  
  
  
 return sum  
  
  
def UpChange(ori,new):  
 if ori <= new:  
 return ori, new  
 else:  
 return new, ori  
  
  
  
cap = cv2.VideoCapture(1)  
print('width :%d, height : %d' % (cap.get(3), cap.get(4)))  
  
while(True):  
 ret, frame = cap.read()  
 if(ret):  
 gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 cv2.imwrite('waka.jpg', frame)  
 cv2.waitKey(1000)  
 break  
if frame is not None:  
 cv2.destroyAllWindows()  
else:  
 print('No image file')  
  
  
#------------------------------------------------  
 #배경 이미지 읽기  
img = cv2.imread('white.jpg')  
rows, cols = img.shape[:2]  
  
mean = cv2.pyrMeanShiftFiltering(img, 10, 80) #두번째 인자는 컬러풀한거 (낮을 수록 컬러풀해짐 높으면 색이 죽음, 두번쨰 인자는 흐릿한 정도)  
  
new = 0  
  
for i in range(0,rows):  
 j=0  
 while(1):  
 k=0  
 while(1):  
 ori=Summing(mean,i,j)  
 k=k+1  
 if k == 3:  
 break  
 if new == 0:  
 new = ori  
 ori,new = UpChange(ori, new)  
 new = ori  
 j=j+1  
 if j == cols:  
 break  
  
  
  
perfect = new // 2.8  
  
# cam으로 찍은 사진 불러오기  
img = cv2.imread('waka.jpg')  
  
rows, cols = img.shape[:2]  
  
# 표면을 흐릿하게 피라미드 평균 시프트 적용  
mean = cv2.pyrMeanShiftFiltering(img, 30, 70) #두번째 인자는 컬러풀한 거 (낮을 수록 컬러풀해짐 높으면 색이 죽음, 두번째 인자는 흐릿한 정도)  
  
# 바이너리 이미지 변환(grayscale image로 변환)  
gray = cv2.cvtColor(mean, cv2.COLOR\_BGR2GRAY)  
gray = cv2.GaussianBlur(gray, (5,5), 0)  
  
\_, thresh = cv2.threshold(gray, perfect, 255, cv2.THRESH\_BINARY\_INV)  
  
  
# 거리 변환 결과값의 절반 이상을 차지한 영역은 확실한 전경으로 설정  
ret, sure\_fg = cv2.threshold(thresh, 0.5\*thresh.max(), 255,0) #스레시홀드에 대한 정의 정확히 알자, 255,0 으로 이진화 시키는 것  
  
# 거리 변환 결과를 반전해서 확실한 배경 찾기  
\_, bg\_th = cv2.threshold(thresh, 0.3\*thresh.max(), 255, cv2.THRESH\_BINARY\_INV) #THRESH\_BINARY\_INV 이거 반전효과이다.  
bg\_dst = cv2.distanceTransform(bg\_th, cv2.DIST\_L2, 5)  
  
  
bg\_dst = ( (bg\_dst / (bg\_dst.max() - bg\_dst.min())) \* 255 ).astype(np.uint8)  
ret, sure\_bg = cv2.threshold(bg\_dst, 0.3\*bg\_dst.max(), 255,cv2.THRESH\_BINARY)  
#이 코드는 간단하게 생각해서 반전되기 전의 값보다 0.3배 커진 값을 255값으로 둔 것이다.(배경을 구분 짓기 위해서)  
  
  
# 불확실한 영역 설정 : 확실한 배경을 반전해서 확실한 전경을 빼기  
ret, inv\_sure\_bg = cv2.threshold(sure\_bg, 127, 255,cv2.THRESH\_BINARY\_INV) #여기서 두번째 인자 값은 255보다 작은 값이면  
 #크게 의미가 없는 것 같다.  
  
unkown = cv2.subtract(inv\_sure\_bg, sure\_fg) #색 전환 한 배경에서 전경을 뺀 것이다.  
  
#cv2.imshow('unkown', unkown)  
#일단 여기서 255인 값들은 전경과 배경을 구분 지을 수 없는 부분이다.  
  
# 연결된 요소 레이블링  
\_, markers = cv2.connectedComponents(sure\_fg) #흰색부분을 부분부분 ndarray식으로 반환해준다.  
  
  
# 레이블링을 1씩 증가 시키고 0번 레이블 알 수 없는 영역을 0번 레이블로 설정  
  
markers = markers+1 #이건 markers에 인덱스를 하나 추가시키고 값도 하나씩 증가!  
  
markers[unkown == 255] = 0 #unkwon 에서 흰색부분에 \*위치\* 들을 0으로 만든다.  
#배경과 전경을 명확히 알 수 없는 부분을  
  
  
   
colors = []  
marker\_show = np.zeros\_like(img) #0으로 초기화된 배열생성 row, column은 똑같이 하고!  
  
for mid in np.unique(markers): # 선택한 마커 아이디 개수 만큼 반복  
 color = [int(j) for j in np.random.randint(0,255, 3)] #3개씩 나뉘어서 배열로 저장!  
 colors.append((mid, color)) #튜플로 만드는 것이고, [(0,[x, y, z]),~~~~] 이런 식으로 colors에 저장된다.  
  
 marker\_show[markers==mid] = color  
 coords = np.where(markers==mid)  
 #coords에는 markers에서 불러온 위치 값에서 mid와 같은 값을 찾아내 x, y좌표를 받아들이는 것이다.  
 x, y = coords[1][0], coords[0][0]  
 cv2.putText(marker\_show, str(mid), (x+20, y+20), cv2.FONT\_HERSHEY\_PLAIN, \  
 2, (255,255,255))  
 #putText는 그냥 str(mid) 에서 받아들인 문자열을 (x+20, y+20) 위치에 출력하는 것이다.  
# 레이블링이 완성된 마커로 워터 쉐드 적용  
markers = cv2.watershed(img, markers)  
  
for mid, color in colors: # 선택한 마커 아이디 개수 만큼 반복  
 marker\_show[markers==mid] = color  
 coords = np.where(markers==mid)  
 if coords[0].size <= 0 :  
 continue #coords[0] 사이즈가 0이 넘으면  
 x, y = coords[1][0], coords[0][0]  
 cv2.putText(marker\_show, str(mid), (x+20, y+20), cv2.FONT\_HERSHEY\_PLAIN, \  
 2, (255,255,255))  
marker\_show[markers==-1] = (0,255,0) #markers == -1 이거 경계선인거 같다. 0,255,0 이면 bgr에서 g만 255니깐.  
  
img[markers==-1] = (0,255,0)  
  
# 빵 추출을 위한 마스킹 생성  
mask = np.zeros((rows, cols), np.uint8)  
# 배경 마스크 생성  
mask[markers != 1] = 255  
# 배경 지우기  
nobg = cv2.bitwise\_and(img, img, mask=mask)  
# bread만 있는 라벨 생성 (배경(1), 경계(-1) 없는)  
bread\_label = [l for l in np.unique(markers) if (l != 1 and l !=-1)]  
# bread 라벨 순회 하면서 bread 영역만 추출  
for i, label in enumerate(bread\_label):  
 mask[:,:] = 0  
 # 해당 bread 추출 마스크 생성  
 mask[markers ==label] = 255  
 # bread 영역만 마스크로 추출  
 bread = cv2.bitwise\_and(img, img, mask=mask)  
 # bread 하나만 있는 곳에서 최외곽 컨투어 추출  
 contour, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL,\  
 cv2.CHAIN\_APPROX\_NONE)  
 # bread를 감싸는 사각형 좌표  
 x,y,w,h = cv2.boundingRect(contour[0])  
 # bread 영역만 추출해서 출력  
 breads = bread[y:y+h, x:x+w]  
 path='c:\bread'  
 cv2.imwrite(os.path.join(path,'bread%d.jpg'%(i+1)), breads)  
  
  
cv2.waitKey()  
cv2.destroyAllWindows()

**3.1.3 서버 / 클라이언트**

서버를 구축하는 과정에서 eclipse를 이용하기에 먼저 eclipse부터 설치를 진행한다. [www.eclipse.org](http://www.eclipse.org) 에서 eclipse 파일을 받는다.



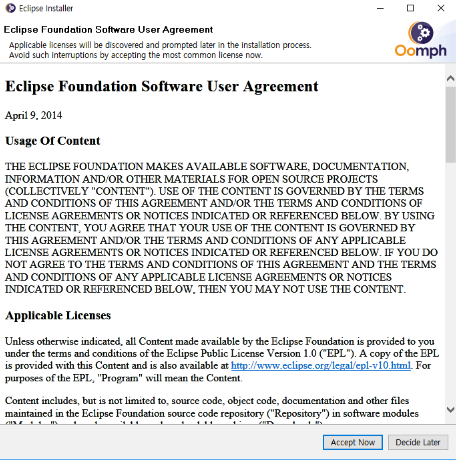
**[그림21 eclipse 설치화면]**

웹프로그래밍을 할 수 있도록 Eclipse IDE for Java EE Developers를 선택하여 설치한다. (필요하면 나중에 플러그인으로 다른 기능도 모두 설치할 수 있다. 하지만 하나의 IDE에 여러 기능을 설치하는 것보다, PHP 개발등이 필요하다면 PHP 용 Eclipse를 하나 새로 설치하는 게 나을 수도 있다. 해보고 자신이 편한대로 개발목적에 맞게끔 설치하면 되겠다.)



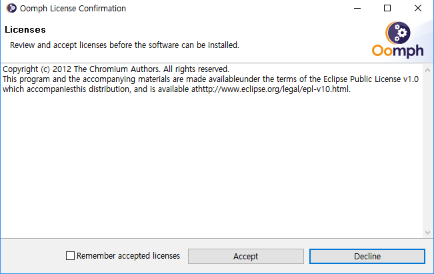
**[그림22 eclipse 설치화면]**

Eclipse의 설치 위치를 지정한다.



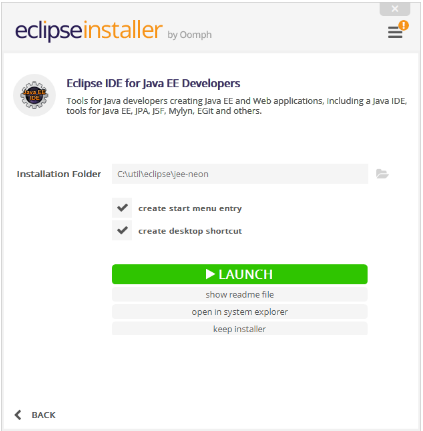
**[그림23 eclipse 설치화면]**

라이선스에 동의를 실시한다.



**[그림24 eclipse 설치화면]**

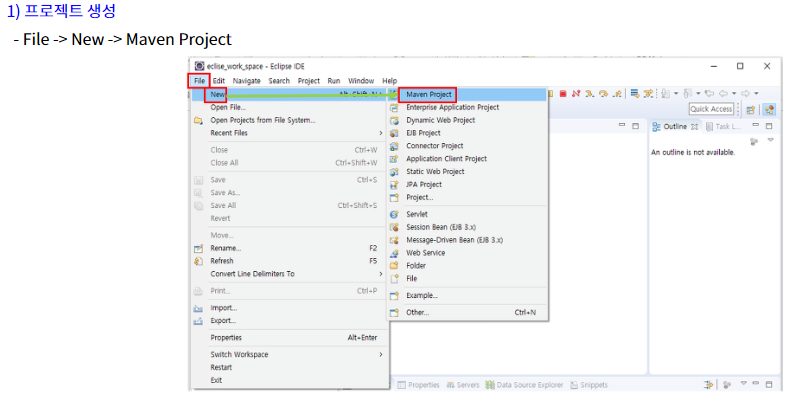
움프(Oomph)라 불리는 새로운 프로젝트는 IDE를 좀더 쉽게 설치할 수 있게 하고, 프로젝트 공간을 프로비저닝 한다. 과거와 달리 새롭게 적용되어 지금 사용하고 있는 설치 프로그램도 Oomph를 사용하여 만들어졌다고 한다.



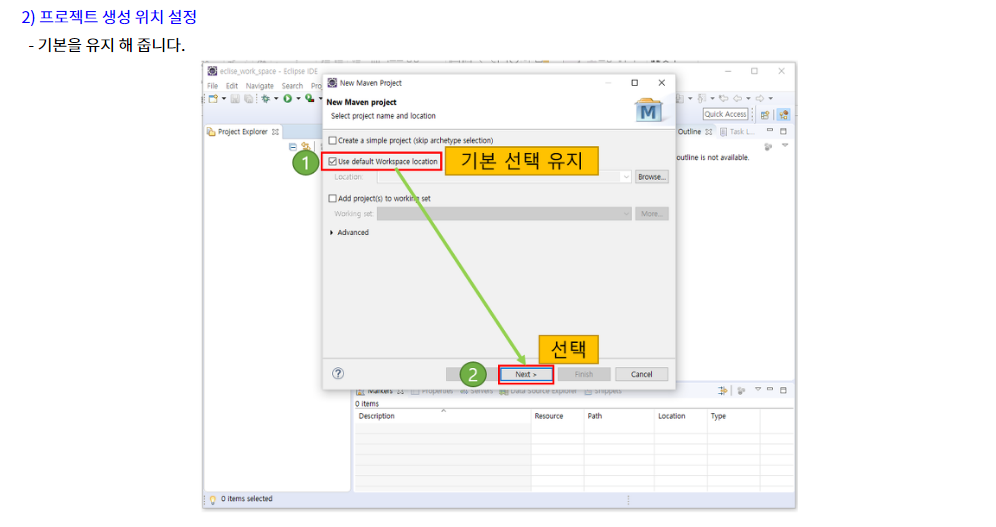
**[그림25 eclipse 설치화면]**

LAUNCH를 눌러서 실행한다.

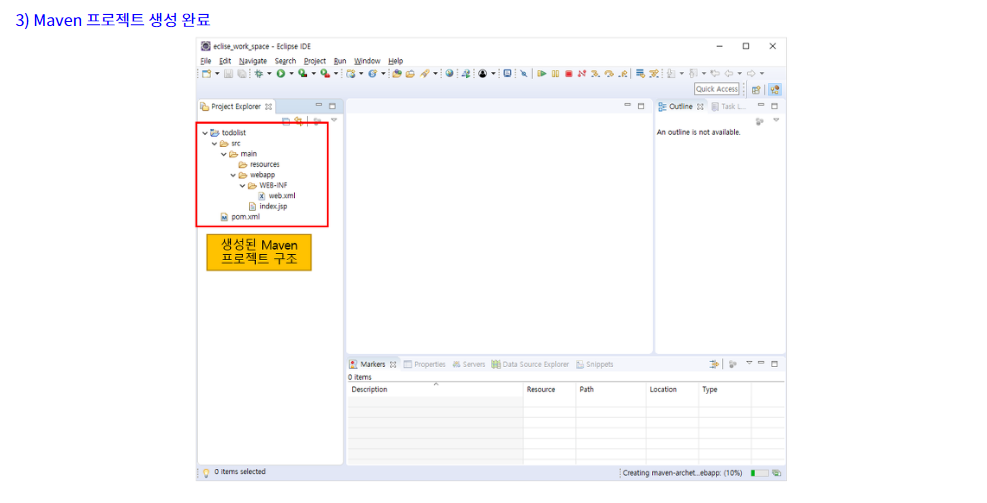
서버를 구축하는데 있어서 java를 사용하여 netty.io 프레임워크를 이용하여 서버를 구축하였기에 간단하게 eclipse에서 maven Project를 생성해 주어야한다.



**[그림26 maven project 생성파일 1]**



**[그림27 maven project 생성파일 위치설정]**



**[그림28 maven project pom.xml 위치]**

이렇게 maven project 생성이 완료되면 pom.xml에 netty.io 프레임워크를 다운받는데 필요한 로직을 옮겨 준다. 이때 dependencies에 오류가 날수 있으니 꼭 </> </>로 잘 막아주고 열어주자.

|  |
| --- |
| **<project>xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">**  **<modelVersion>4.0.0</modelVersion>**  **<groupId>NettyExam</groupId>**  **<artifactId>NettyExam</artifactId>**  **<version>0.0.1-SNAPSHOT</version>**  **<name>NettyExam</name>**  **<build>**  **<sourceDirectory>src</sourceDirectory>**  **<plugins>**  **<plugin>**  **<artifactId>maven-compiler-plugin</artifactId>**  **<version>3.8.0</version>**  **<configuration>**  **<release>12</release>**  **</configuration>**  **</plugin>**  **</plugins>**  **</build>**  **<!-- https://mvnrepository.com/artifact/io.netty/netty-all -->**  **<dependencies>**  **<dependency>**  **<groupId>io.netty</groupId>**  **<artifactId>netty-all</artifactId>**  **<version>4.1.43.Final</version>**  **</dependency>**  **<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->**  **<dependency>**  **<groupId>mysql</groupId>**  **<artifactId>mysql-connector-java</artifactId>**  **<version>5.1.48</version>**  **</dependency>**    **</dependencies>**    **</project>** |

이렇게 dependencies 부분에 netty.io에 관련된 로직 즉 사이트 경로를 입력하여주면 eclipse내에서 관련 필요한 모듈들의 설치를 진행한다.

서버 MAIN소스이다.

|  |
| --- |
| **package com.terry.server;**  **import com.terry.sample.DiscardServer;**  **import io.netty.bootstrap.ServerBootstrap;**  **import io.netty.channel.ChannelFuture;**  **import io.netty.channel.ChannelInitializer;**  **import io.netty.channel.ChannelOption;**  **import io.netty.channel.EventLoopGroup;**  **import io.netty.channel.nio.NioEventLoopGroup;**  **import io.netty.channel.socket.SocketChannel;**  **import io.netty.channel.socket.nio.NioServerSocketChannel;**  **import io.netty.handler.codec.DelimiterBasedFrameDecoder;**  **import io.netty.handler.codec.Delimiters;**  **import io.netty.handler.codec.string.StringDecoder;**  **import io.netty.handler.codec.string.StringEncoder;**  **public class BreadServer {**  **private int port;**  **public BreadServer(int port) {**  **this.port = port;**  **}**  **public void run() throws Exception {**  **EventLoopGroup bossGroup = new NioEventLoopGroup(); // (1)**  **EventLoopGroup workerGroup = new NioEventLoopGroup();**  **try {**  **ServerBootstrap b = new ServerBootstrap(); // (2)**  **b.group(bossGroup, workerGroup)**  **.channel(NioServerSocketChannel.class) // (3)**  **.childHandler(new ChannelInitializer<SocketChannel>() { // (4)**  **@Override**  **public void initChannel(SocketChannel ch) throws Exception {**  **ch.pipeline().addLast(new DelimiterBasedFrameDecoder(8192, Delimiters.lineDelimiter()));**  **ch.pipeline().addLast(new StringDecoder());//바이트로 들어오는것을 스트링으로 변환해준다(데이터를 수신할때)**  **ch.pipeline().addLast(new StringEncoder());//데이터를 송신할때 스트링을 송신하면 byte형식으로 묶어주는것이다.**  **ch.pipeline().addLast(new BreadServerHandler());**  **}**  **})**  **.option(ChannelOption.SO\_BACKLOG, 128) // (5)**  **.childOption(ChannelOption.SO\_KEEPALIVE, true); // (6)**  **// Bind and start to accept incoming connections.**  **ChannelFuture f = b.bind(port).sync(); // (7)**  **// Wait until the server socket is closed.**  **// In this example, this does not happen, but you can do that to gracefully**  **// shut down your server.**  **f.channel().closeFuture().sync();**  **} finally {**  **workerGroup.shutdownGracefully();**  **bossGroup.shutdownGracefully();**  **}**  **}**  **public static void main(String[] args) throws Exception {**  **int port = 33060;**    **new BreadServer(port).run();**  **}**  **}** |

서버 HANDLER부분이다.

|  |
| --- |
| **package com.terry.server;**  **import com.terry.business.DoBread;**  **import io.netty.channel.Channel;**  **import io.netty.channel.ChannelHandlerContext;**  **import io.netty.channel.ChannelInboundHandlerAdapter;**  **import io.netty.channel.group.ChannelGroup;**  **import io.netty.channel.group.DefaultChannelGroup;**  **import io.netty.util.concurrent.GlobalEventExecutor;**  **public class BreadServerHandler extends ChannelInboundHandlerAdapter{**  **private static final ChannelGroup channelGroup = new DefaultChannelGroup(GlobalEventExecutor.INSTANCE);**    **@Override**  **public void handlerAdded(ChannelHandlerContext ctx) throws Exception {**  **System.out.println("handlerAdded of [SERVER]");**  **Channel incoming = ctx.channel();**  **for (Channel channel : channelGroup) {**  **//사용자가 추가되었을 때 기존 사용자에게 알림**    **}**  **channelGroup.add(incoming);**  **}**  **@Override**  **public void handlerRemoved(ChannelHandlerContext ctx) throws Exception {**  **// TODO Auto-generated method stub**  **System.out.println(ctx.channel().toString() +" handlerRemoved");**  **}**  **@Override**  **public void channelActive(ChannelHandlerContext ctx) throws Exception {**  **// TODO Auto-generated method stub'**  **System.out.println(ctx.channel().toString() +" channelActive");**  **}**  **@Override**  **public void channelInactive(ChannelHandlerContext ctx) throws Exception {**  **// TODO Auto-generated method stub**  **System.out.println(ctx.channel().toString() +" channelInactive");**  **}**  **@Override**  **public void channelRead(ChannelHandlerContext ctx, Object msg) { // (2)**    **String message = null;**  **message = (String)msg;**  **if("bye".equals(message.toLowerCase())) { //대문자인지 소문자인지 모르기때문에 무조건 소문자로 바꿔준다.**  **ctx.close();**    **}else if("1".equals(message))**  **{**  **System.out.println("newyork cheese, 2500");**  **}else if("2".equals(message))**  **{**  **System.out.println("gorokee, 1700");**  **}else if("3".equals(message))**  **{**  **System.out.println("kaechar, 900");**  **}else if("4".equals(message))**  **{**  **System.out.println("kwabaki, 900");**  **}else if("5".equals(message))**  **{**  **System.out.println("mini donut, 1000");**  **}else if("6".equals(message))**  **{**  **System.out.println("pizza, 1700");**  **}**  **else if("7".equals(message))**  **{**  **System.out.println("soboro, 900");**  **}else if("8".equals(message))**  **{**  **System.out.println("sosage, 1300");**  **}**  **else if("9".equals(message))**  **{**  **System.out.println("x bread, 1300");**  **}**  **else if("10".equals(message))**  **{**  **System.out.println("none");**  **}**  **else if("0".equals(message))**  **{**  **System.out.println("shucream, 700");**  **}**        **else {**  **System.out.println(message);**  **DoBread doBread = new DoBread(message);**  **String result = doBread.getBread();**  **System.out.println("channelRead of [SERVER]" + message);**  **Channel incoming = ctx.channel();**  **for (Channel channel : channelGroup) {**  **if (channel != incoming) {**  **//메시지 전달.**  **channel.writeAndFlush(message);**  **System.out.println(message);**  **}**  **}**  **}**  **}**  **@Override**  **public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) { // (4)**  **// Close the connection when an exception is raised.**  **cause.printStackTrace();**  **ctx.close(); //**  **}**  **}** |

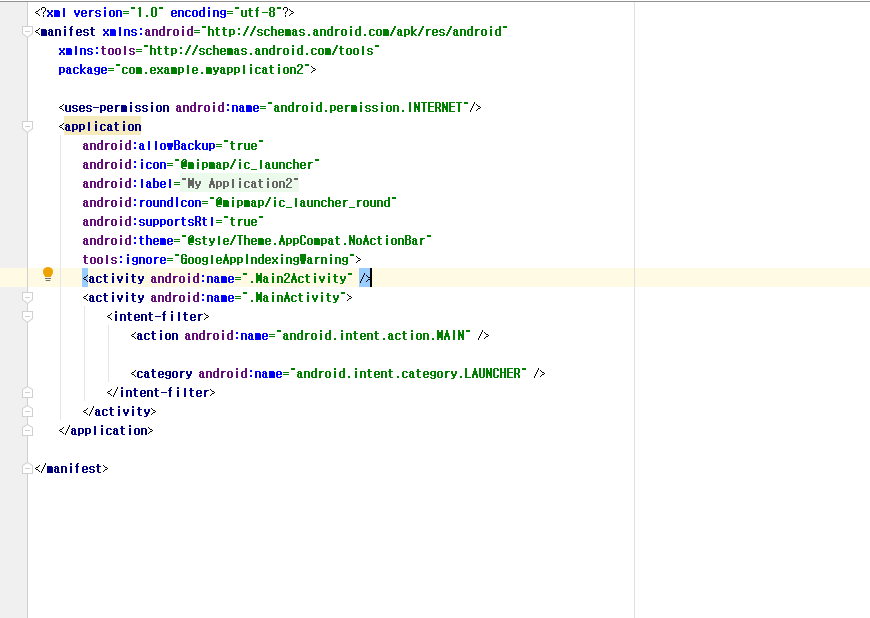
클라이언트는 안드로이드 스튜디오를 이용하여 안드로이드 어플로 제작하였다.

(<https://developer.android.com/studio/>) 링크를 타고 이동해 최신버전의 64bit로 설치해주면 된다.

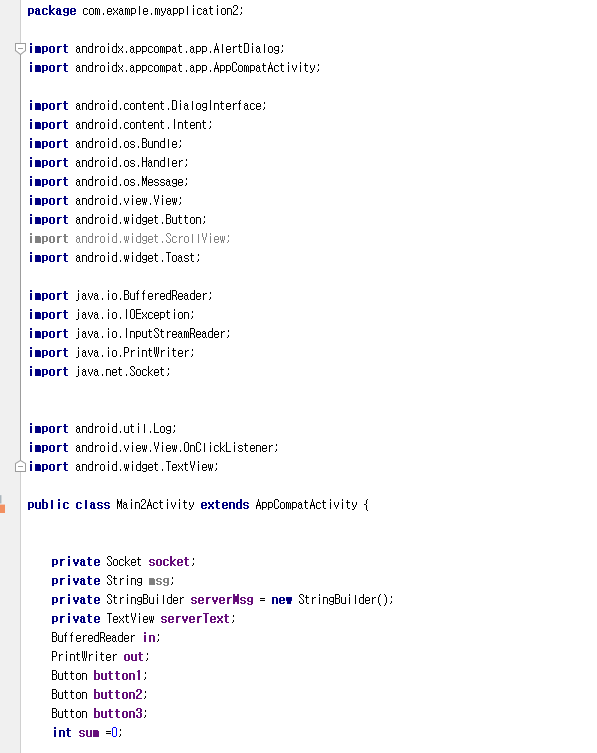
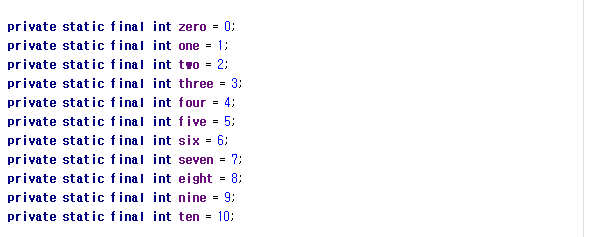
클라이언트 코드

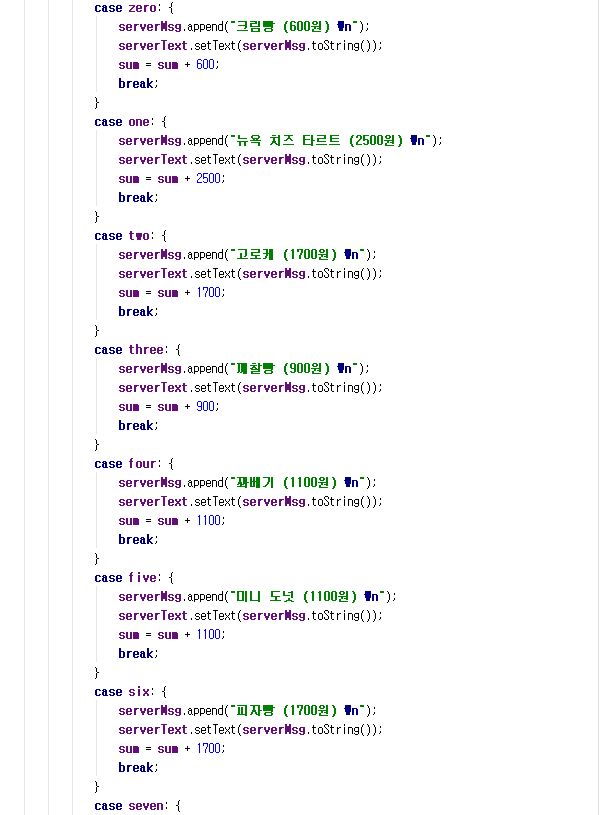
****activity\_main.xml

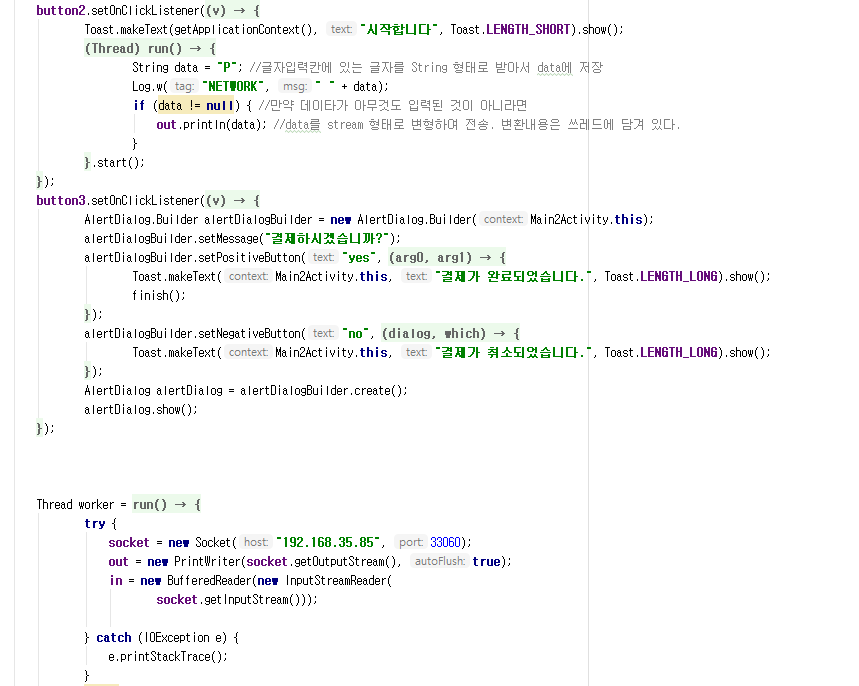
****MainActivity.java

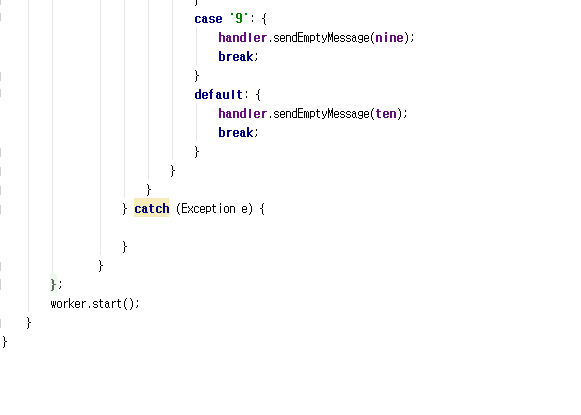
****AndroidManifest.xml

****activity\_main2.xml

Main2Activity.java



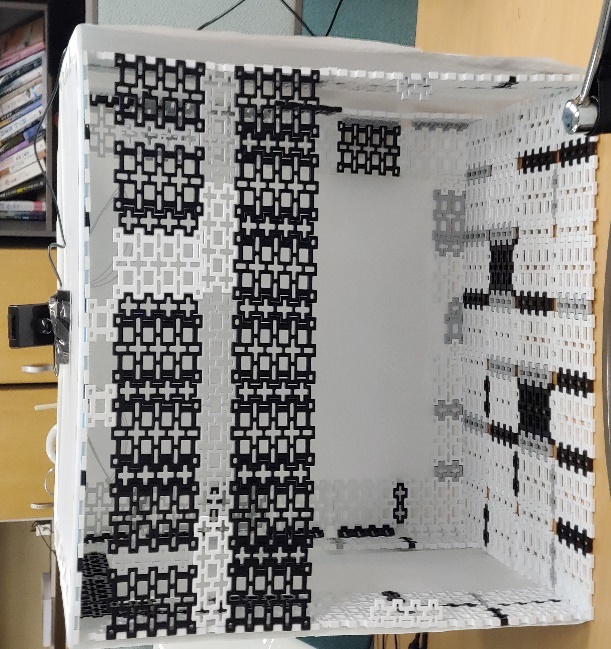




안드로이드 작성 코드는 총 5개의 파일로 이루어져 있다. 화면을 꾸며주는 xml 파일 2개와 작동하는 코드로 이루어진 java 파일 2개, 그리고 설치된 애플리케이션의 구성요소나 권한을 정의해주는 매니페스트 파일로 이루어져 있다.

**3.3. 하드웨어**

하드웨어로 간단하게 이동시키기 위해 라즈베리파이4 라는 소형 컴퓨터를 이용할 예정이었으나, 딥러닝의 기술상 기기성능에 따라 속도가 결정되므로 성능이 뛰어난 컴퓨터 및 노트북을 이용하게 되었다. 그렇기에 추가적으로 이 프로그램을 이용하려면 기기의 성능을 어느정도 맞춰주어야 한다.

이미지촬영을 위해 캠코더를 이용하였고, usb 선을 통해 PC와 연결시켜 이용하였다.

**[그림29 캠코더 및 케이스 모형]**

실전을 생각하고 만들어 둔 케이스 모형이다. 블록을 이용해 쉽사리 무너지지 않을 지지대를 만들었고, 빛을 가리기 위해 빵을 집어넣을 공간을 제외하고는 빛을 차단해 두었다. 캠코더에서 한 번에 트레이의 공간을 담기 위해 높이를 조절해 주었다. 추가적으로 항상 동일한 빛의 세기를 받게 하기 위해 조명도 넣어두었다.



**[그림30 트레이]**

빵을 집어넣고 다시 빼기 위한 트레이의 모형이다.



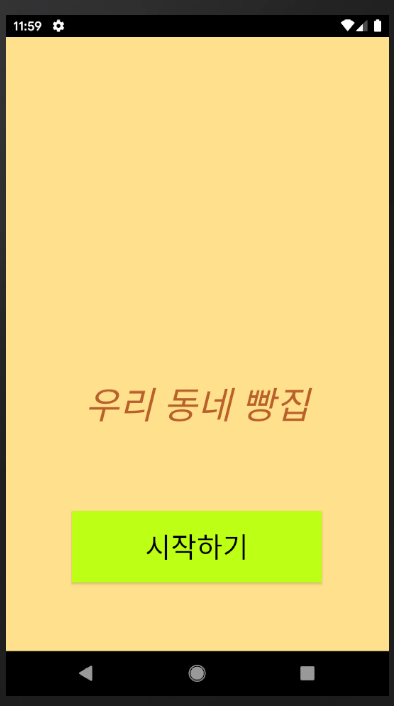
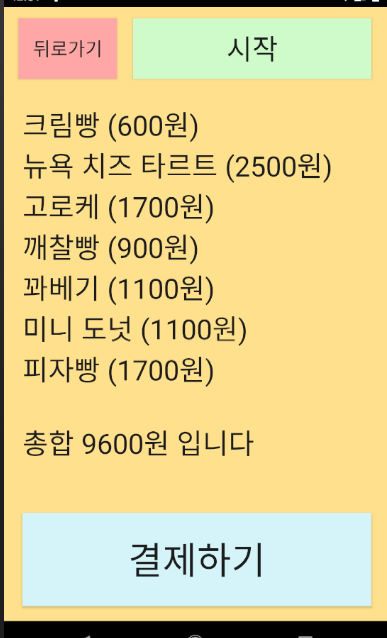
**[그림31 케이스의 전체 모습]**

**4절 결론 및 개선 방안**

**4.1 결론 및 기대효과**

**4.1.1 결과**

미리 준비해둔 이미지 데이터셋과 만들어 둔 서버 및 클라이언트와 딥러닝을 이용할 파이썬을 전부 실행시켜 둔다. 클라이언트에서 시작하기 버튼을 눌러 서버에 접속함과 동시에 서버에 특정 신호를 보낸다. 신호를 받은 서버는 특정신호를 연동된 파이썬 프로그램에 보내게 되고, 그 신호를 받은 파이썬 프로그램은 작동하기 시작한다. 작동 순서는 OpenCV기능을 먼저 이용해 cam의 실행시키고 이미지를 확보한 후 여러 작업을 거쳐 폴더에 이미지를 저장해 둔다. 폴더안의 저장된 이미지를 기반으로 가지고 있던 데이터셋과 비교 분석하는 딥러닝이 시작되고, 그 후 판별하여 어떠한 빵들이 있는지 알아낸다. 알아낸 빵들을 서버에 전달하여 주고, 서버에서 받은 정보를 기반으로 빵의 이름과 가격, 그리고 모든 가격을 합한 총 구매 금액을 클라이언트에 전송해준다. 클라이언트에서는 받은 정보를 출력해주고 계산을 유도하여 준다.



**4.1.2 기대효과**

점원이 빵을 일일이 구분하여 직접 포스기에 입력하여 계산하는 방식보다 훨씬 빠르고 간편하게 계산할 수 있다. 손님은 그저 가져온 트레이를 집어넣기만 하면 되고, 점원은 그저 버튼 하나만 클릭하면 빵의 종류와 총 계산 값이 나온다. 만들어 둔 클라이언트에 이후에 결제시스템까지 연동한다면 매우 간단하고 빠른 계산이 가능해질 것이다.

**4.2 문제점 및 개선 방안**

**4.2.1 문제점**

빠른 결제가 주 목적인 만큼 소모되는 시간이 적어야 한다. 그러나 기기의 성능에 따라 속도가 다르다는 점이 문제점 중 하나이다. 실제로 테스트 도중 기기성능이 떨어지는 하드웨어는 길게는 1분이 넘는 시간을 소모하였다. 성능이 좋은 기기는 소모되는 시간이 5초까지 떨어졌기에 성능이 더 좋은 기기면 더 짧은 시간이 소모될 것으로 예상된다.

떨어져 있는 빵들은 각각 하나의 개체로 인식하나 붙어 있는 경우는 어느정도 붙어있는가에 따라 결과가 달라진다. 심지어 빵들이 쌓여 있는 경우는 더욱 더 결과 값이 일정하지 않다. 그렇기에 빵을 인식하기 전에 어느정도 배치에 신경을 써야하는 점이 문제점이다.

**4.2.2 개선 방안**

속도의 문제는 코드의 개선 및 최적화와 기기의 성능을 올리는 것으로 문제점을 해결할 수 있다. 좀 더 성능이 뛰어난 기기를 준비하는 것으로 보안이 가능하다.

겹쳐 있는 빵을 따로 구분 짓는 방법은 분명히 있다. 조금 더 시간을 투자하여 공부를 통해 코드를 변경하여 해결할 수 있다.

**5절 참고문헌 및 사이트**

파이썬으로 만드는 OpenCV 프로젝트 / 인사이트



<https://book.naver.com/bookdb/book_detail.nhn?bid=14652444>

골빈해커의 3분 딥러닝 텐서플로맛



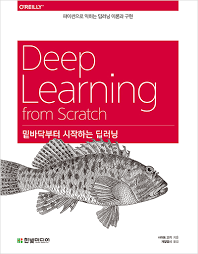
<https://book.naver.com/bookdb/book_detail.nhn?bid=12556028>

코딩셰프의 3분 딥러닝 케라스맛



<https://book.naver.com/bookdb/book_detail.nhn?bid=12892401>

밑바닥부터 시작하는 딥러닝



<https://book.naver.com/bookdb/book_detail.nhn?bid=11492334>

머신러닝, 딥러닝 실전개발 입문



<https://book.naver.com/bookdb/book_detail.nhn?bid=12129887>

네티 인 액션



<https://book.naver.com/bookdb/book_detail.nhn?bid=10462610>

안드로이드 프로그래밍

<https://book.naver.com/bookdb/book_detail.nhn?bid=13425669>