


캡스톤 디자인	
작품(과제)명	Automatic Sketch Coloring service/system based on Machine Learning 머신러닝을 활용한 밑그림 자동 채색 서비스 및 시스템
1. 개발동기 목적, 필요성 및 최종 개발 목표	<p>머신러닝을 활용한 밑그림 자동 채색 서비스 및 시스템을 통해 사용자가 직접 그린 이미지 또는 카메라로 찍은 이미지를 시스템의 입력 값으로 넣었을 때 자동으로 채색 된 이미지를 출력 값으로 얻는다.</p> <p>사용자들이 본 시스템을 통해서 '머신러닝' 을 직접적으로 체험할 수 있는 콘텐츠는 없으므로 쉽게 다가 갈 수 있는 기회를 제공하고자 한다. 엔터테인먼트 적인 요소뿐만 아니라 본 '머신러닝을 활용한 밑그림 자동 채색 서비스 및 시스템'을 통해 효율적인 채색작업을 요구하는 게임산업, 웹툰 산업에 이용하여 작업의 시간을 단축시켜 경제성을 높여 사회적인 측면에서 긍정적인 효과를 높일 수 있을 것이다. 또한, 컴퓨터그래픽을 이용한 디자인영역에서 다양한 채색들의 조합에 대한 고민이 들 때 색을 추천해줌으로써 시간 감소 효과를 가져올 수 있다. 그리고 저시력자, 시각장애인, 색약인 대상으로 막연했던 채색작업에 대한 자신감을 가지고 자신이 그린 밑그림에 어울리는 색을 추천해줌으로써 작품 완성도를 높여줄 수 있다.</p> <p>최종목표는 사용자가 직접 그리는 이미지 밑그림이 실시간으로 서버에 전송되어 머신 러닝을 통해 자동 채색된 결과 값을 사용자에게 보여주는 결과를 가져 온다.</p>
2. 최종 결과물 소개	<p>기기들 간의 서버 연결이 중요하고 정보의 유실에 유의한다. TCP 소켓연결을 통한 동일 wifi 연결 및 웹을 기반으로 한 서버로 구성한다.</p> <ul style="list-style-type: none"> <li>-디스플레이와 라즈베리파이 에서 처리한 input 이미지</li> <li>-디스플레이와 라즈베리파이 간의 통신</li> <li>-라즈베리파이와 서버와의 연동 후 output 이미지</li> </ul> 

- 밑그림 채색 결과



밑 그림 input

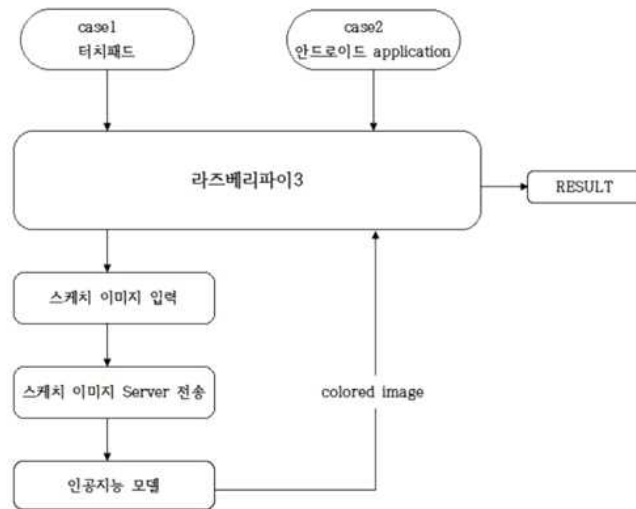
채색 그림 output



원하는 옷 또는 헤어 색상 지정 후 결과물 추출 가능

3. 프로젝트  
과정  
❶ 개념 설계

<그림 1 아키텍처>



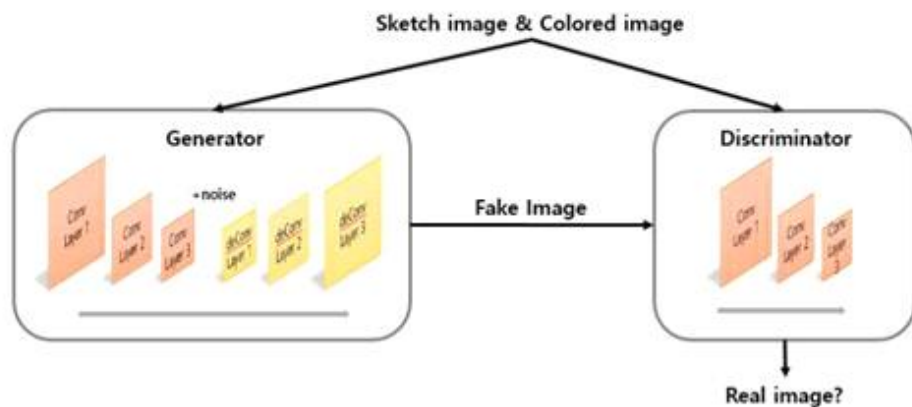
터치패드 or 안드로이드 어플리케이션으로 사용자가 직접 그린 그림을 라즈베리 파이로 전송한다. 받은 이미지를 python 에서 호환가능하도록 python 환경으로 이미지를 입력한다.

server 로 전송하기 위해 jpg-> base64 변환하여 전송한다. 텐서플로를 사용하여 만들어진 모델을 이용하여 전송된 이미지를 자동으로 채색한다.

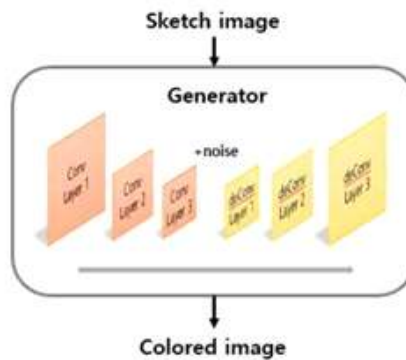
채색된이미지를 다시 라즈베리파이 3 로 전송하기위해 base64->jpg 로 변환하고 전송한다.

완료된 이미지를 터치패드 or 안드로이드로 출력한다.

<그림2 소프트웨어 순서도 convolution + GAN>



<그림3 생성된 모델을 통한 Colored image 생성>



머신러닝을 활용한 프로젝트 완성을 큰 목표로 하여 구체적으로 주제를 선정하는 과정을 통해 프로젝트를 추진 하였다. 우선 머신러닝에 대한 기본적인 이해를 기반으로 하여 사용자에게 쉽게 다가 갈 수 있는 가시적인 영역인 그림을 기반으로 하여 테두리만 있는 그림에 자동으로 채색하는 시스템을 주제로 선정하여 프로젝트를 진행하게 되었다.

일단, 밑그림에 채색할 수 있는 시스템을 구현할 수 있는 머신러닝 알고리즘 선정 하도록 하였다. 뉴럴 네트워크를 사용하여 이미지를 생성해 내는 방법 중 가장 최근에 개발된 GAN(Generative Adversarial Networks)를 선택하고 머신러닝에 학습시킬 데이터셋을 수집하였습니다

구글에서 개발한 웹크롤링 코드를 사용하여 관련 이미지의 검색 웹을 크롤링 하여 이미지를 수집하였습니다. 그리고 나중에 input 으로 쓰일 이미지로 가공하는 활동을 하였습니다. 인텔에서 개발하고 실시간 이미지 프로세싱이 가능한 오픈소스 라이브러리인 **OpenCV**(Open Source Computer Vision)를 통해 처리 하였습니다.

## ② 구현 과정

# Ubuntu 환경 python 언어 기반

# web crawling : 웹사이트에 검색 키워드를 코드에 입력하면 컴퓨터에 자동으로 저장된다.

```
from icrawler.builttin import GoogleImageCrawler
google_crawler = GoogleImageCrawler(parser_threads=2, downloader_threads=4,
                                     storage={'root_dir': 'adventure time finn'})
google_crawler.crawl(keyword='adventure time finn png', max_num=1000,
                     date_min=None, date_max=None,
                     min_size=None, max_size=None)
```

# Raspberry GUI 인 Tkinter 라이브러리의 Canvas()함수를 사용하여 터치패드나 안드로이드를 통해 사용자 touch 하여 얻은 좌표 입력 값을 gui 상에서 구현

#c 코드를 swig 라는 툴을 사용해서 python 코드로 변환하였고 웹서버에서 구현하는 코드나 머신러닝에서 사용하는코드가 python 기반이기때문에 통일성을 위해서 변환했다. python 라이브러리의 일부인 Tkinter 를 사용하면서마우스가 그림판에서 움직이도록 하는 gui 를 구현했다.

```
import Tkinter as tk
from PIL import Image, ImageDraw #python의 Tkinter를 import시킨다

class ImageGenerator:
    def __init__(self, parent, posx, posy, *kwargs):
        self.parent = parent
        self.posx = posx
        self.posy = posy
        self.size = 500
        self.sizey = 500
        self.b1 = "up"
        self.xold = None
        self.yold = None
        self.drawing_area = tk.Canvas(self.parent, width=self.size, height=self.sizey) #canvas(그림판 설정)
        self.drawing_area.place(x=self.posx, y=self.posy)
        self.drawing_area.bind("<Motion>", self.motion) #bind메소드를 사용하여 마우스 이동시 motion 함수 호출
        self.drawing_area.bind("<ButtonPress-1>", self.b1down) #bind메소드를 사용하여 마우스 클릭 시 b1down 함수 호출
        self.drawing_area.bind("<ButtonRelease-1>", self.b1up) #bind메소드를 사용하여 마우스 클릭 후 b1up 함수 호출

        self.button = tk.Button(self.parent, text="Save", width=10, bg="white", command=self.save)
        self.button.place(x=500, y=self.sizey+20) #save 버튼생성 및 위치 설정

        self.button1 = tk.Button(self.parent, text="Clear!", width=10, bg="white", command=self.clear)
        self.button1.place(x=500+80, y=self.sizey+20) #clear 버튼생성 및 위치 설정

        self.image = Image.new("RGB", (500, 500), (255, 255, 255))
        self.draw = ImageDraw.Draw(self.image)
        self.output_area = tk.Frame(self.parent, width=self.size, height=self.sizey, bg="coral1") #output 출력
        self.output_area.place(x=550, y=self.posy) #output 프레임 위치

    def save(self):
        filename = "input.jpg"
        self.image.save(filename) #input.jpg로 저장

    def clear(self):
        self.drawing_area.delete("all")
        self.image = Image.new("RGB", (500, 500), (255, 255, 255))
        self.draw = ImageDraw.Draw(self.image) #clear버튼 클릭시 canvas에 데이터 제거

    def b1down(self, event):
        self.b1 = "down" #마우스 클릭시 발생할 이벤트

    def b1up(self, event):
        self.b1 = "up"
        self.xold = None
        self.yold = None #마우스 클릭 후 발생할 이벤트

    def motion(self, event):
        if self.b1 == "down": #마우스 클릭 도중 발생할 이벤트
            if self.xold is not None and self.yold is not None:
                event.widget.create_line(self.xold, self.yold, event.x, event.y, smooth="true", width=3, fill="black")
                self.draw.line((self.xold, self.yold, (event.x, event.y)), (0, 0, 0), width=3) #widget메소드를 사용하여 마우스를 따라 line생성하는 event
                self.xold = event.x
                self.yold = event.y

if __name__ == "__main__":
    root = tk.Tk()
    root.wm.geometry("800x800") # (1000, 600, 10, 10)
    root.config(bg="white")
    image_generator = ImageGenerator(root, 10, 10)
    root.mainloop() #위에 설정된 ui를 window에 출력
```

#테블릿 pc, 안드로이드 앱에서 touch pen 을 사용하여서 입력 좌표 값을 얻은 후, Raspberry pi Python prepare\_socket()함수로 UDP 소켓 연결을 통해서 같은 와이파이 망 내에서 및 send\_event()함수로 라즈베리 파이에 정보를 전송을 한다.

```
//소켓을 생성해서 ipv4 socket타입을 제공해주며 통신이 가능하도록 port 번호를 설정
int prepare_socket(){

    int s;
    struct sockaddr_in addr;

    if ((s = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1){
        perror("error: prepare_socket()"); exit(EXIT_FAILURE); }

    bzero(&addr, sizeof(struct sockaddr_in));
    addr.sin_family = AF_INET;
    addr.sin_port = htons(GFXTABLET_PORT);
    addr.sin_addr.s_addr = htonl(INADDR_ANY);

    if (bind(s, (struct sockaddr *)&addr, sizeof(addr)) == -1){
        perror("error: prepare_socket()"); exit(EXIT_FAILURE); }

    return s;
}

void send_event(int device, int type, int code, int value){

    struct input_event ev;
    ev.type = type;
    ev.code = code;
    ev.value = value;
    if (write(device, &ev, sizeof(ev)) < 0){
        perror("error: write()");
        exit(EXIT_FAILURE); }
}
```

#python코드를 기반으로 image 파일을 upload하며 특정 host및 port에서

```
@route('/upload_lineonly', method='POST')
def do_upload():
    print "Got it"
    # lines = request.files.get('lines')
    # colors = request.files.get('colors')
    line_data = request.form.get("lines")
    line_data = re.sub('^data:image/.+;base64,', '', line_data)
    line_s = base64.b64decode(line_data)
    line_img = np.fromstring(line_s, dtype=np.uint8)
    line_img = cv2.imdecode(line_img, -1)

    lines_img = np.array(cv2.resize(line_img, (512,512)))
    lines_img = np.array([lines_img]) / 255.0
    lines_img = lines_img[:, :, 0]
    lines_img = np.expand_dims(lines_img, 3)

    lines_img_sm = np.array(cv2.resize(line_img, (256,256)))
    lines_img_sm = np.array([lines_img_sm]) / 255.0
    lines_img_sm = lines_img_sm[:, :, 0]
    lines_img_sm = np.expand_dims(lines_img_sm, 3)

    random_z = np.random.normal(0, 1, [p.batch_size, p.z_dim])

    color_img = p.sess.run(p.generated_images, feed_dict={p.line_images:
lines_img_sm, p.guessed_z: random_z})
    color_img = np.array([cv2.resize(x, (512,512),
interpolation=cv2.INTER_NEAREST) for x in color_img])[0]

    color_img = color_img * 255.0
    colors_img = imageblur(color_img, True)
    colors_img = np.array([colors_img]) / 255.0
    colors_img = colors_img[:, :, 0:3]
    generated = c.sess.run(c.generated_images, feed_dict={c.line_images:
lines_img, c.color_images: colors_img})
    cnt = cv2.imencode(".png", generated[0]*255)[1]
    return base64.b64encode(cnt)

run(host="0.0.0.0", port=8000)
```



# GAN CNN 에 코드를 통해서 데이터 셋 돌려보기 텐서플로우로 이미 채색된 이미지를 학습하고 사용자가 스케치한 이미지를 작업하면 자동으로 채색이되는 개념으로 프로그램 설계

#loss, batch size 및 train 시간을 변수로 입력된 스케치 그림 데이터를 머신러닝 알고리즘을 통해서 학습을 시킨다.

```

        d_loss, _ = self.sess.run([self.d_loss, self.d_optim],
        feed_dict={self.real_images: batch_normalized, self.line_images:
        batch_edge, self.color_images: batch_colors})
        g_loss, _ = self.sess.run([self.g_loss, self.g_optim],
        feed_dict={self.real_images: batch_normalized, self.line_images:
        batch_edge, self.color_images: batch_colors})
        print "%d: [%d / %d] d_loss %f, g_loss %f" % (e, i,
        (datalen/self.batch_size), d_loss, g_loss)
        if i % 100 == 0:
            recreation = self.sess.run(self.generated_images,
            feed_dict={self.real_images: base_normalized, self.line_images:
            base_edge, self.color_images: base_colors})
            ims("results/"+str(e*100000 +
            i)+".jpg",merge_color(recreation, [self.batch_size_sqrt,
            self.batch_size_sqrt]))
            if i % 500 == 0:
                self.save("./checkpoint", e*100000 + i)
        def loadmodel(self, load_discrim=True):
            self.sess = tf.Session()
            self.sess.run(tf.initialize_all_variables())
            if load_discrim:
                self.saver = tf.train.Saver()
            else:
                self.saver = tf.train.Saver(self.g_vars)

```

#### 4. 기대효과

그래픽적인 요소에 시간이 많이 투자되고 중요한 부분으로 차지하는 게임산업 분야 와 만화분야에서 시간절약과 관련하여 도움이 된다.  
채색 작업에 투입되는 시간 과 인원을 절약함으로써 경제적으로 도움이 된다.  
시각장애인들, 색약인 사람들에게 도움이 되는 프로그램이다. 그림의 외곽선은 그릴 수 있지만 어떠한 색을 채워넣기에는 색에 대한 경험이 낮기때문에 이를 인공 지능이 도와주려 한다.

<p><b>5. 역할 분담</b></p>	<p>김나현 : 안드로이드와 라즈베리 파이 socket 함수를 통한 정보전송, webcrawling 학습시킬 데이터 수집</p> <p>김진환 : 이미지 전처리</p> <p>이송은: WAS 서버 구축 및 프로젝트 총괄</p> <p>이지연: python 언어를 사용한 gui 환경 구성, server 에 전달하기 위한 데이터 가공</p>
<p><b>6. 산학 협력 멘토링</b></p>	<p><b>1 차멘토링</b></p> <p><b>멘토 : 우아한 형제들 임성현 기술사님</b></p> <p><b>일시 : 2017 년 3 월 23 일</b></p> <ul style="list-style-type: none"> <li>- 인공지능에 대해서 기본 지식 습득 및 활용 사례 공유</li> <li>-팀원 개개인의 희망사항 및 프로젝트 방향 협의</li> <li>-프로젝트 완성 결과에 대한 기대사항 공유</li> <li>-업계 동향 및 향후 로드맵을 감안한 범위 조율</li> </ul> <p><b>2 차 멘토링</b></p> <p><b>멘토 : 우아한 형제들 임성현 기술사님</b></p> <p><b>일시 : 2017 년 4 월 17 일</b></p> <ul style="list-style-type: none"> <li>-지금은 타 사례를 감안해서 어떻게 하면 성공적으로 프로젝트를 마무리 할 수 있을지 고민하면서 풀어가는 것이 현명할 것으로 판단됨</li> <li>-인공지능을 활용한 다양한 사례를 수집할 것</li> <li>-먼저는 예제를 따라하면서 어떻게 tensorflow 를 활용할 수 있을지 사용성 관점에서 바라보기를 요청함 학생들에게 전달할 추가 보완요청사항</li> </ul> <p><b>3 차 멘토링</b></p> <p><b>멘토 : 우아한 형제들 임성현 기술사님</b></p> <p><b>일시 : 2017 년 5 월 6 일</b></p> <ul style="list-style-type: none"> <li>- 현재 진행사항에 대한 확인</li> <li>- 프로젝트 주요 이슈 확인 -&gt; 각자의 해결 방안 공유</li> <li>- 잔여 일정을 감안하여 남은 시간 안에 진행할 수 있는 계획 요청 프로젝트 역할 분담에 따라 진행하면서 다른 팀원들과 반드시 공유 요청</li> </ul>