

①MSC

a. 수치해석 교과목 리포트

-Python에서 Gauss quadrature을 이용한 유량(Q) 계산

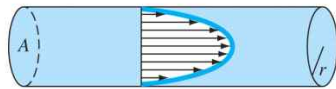
If the velocity distribution of a fluid flowing through a pipe is known, the flow rate Q (that is, the volume of water passing through the pipe per unit time) can be computed by $Q = \int v dA$, where v is the velocity and A is the pipe's cross-sectional area. For a circular pipe, $A = \pi r^2$ and $dA = 2\pi r dr$. Therefore,

$$Q = \int_0^{r_0} v(2\pi r) dr$$

Where r is the radial distance measured outward from the center of the pipe. If the velocity distribution is given by

$$v = 2 \left(1 - \frac{r}{r_0} \right)^{1/6}$$

Where r_0 is the total radius ($r_0=3\text{cm}$), compute Q Gauss quadrature with 3, 4 and 5 points.



```
1 import math
2 import numpy as np
3
4 n = int(input('n = '))
5 c = np.zeros(n)
6 x = np.zeros(n)
7 g = np.zeros(n)
8 i = 0
9 if n == 3:
10     c = [0.5555556, 0.8888889, 0.5555556]
11     x = [-0.77459669, 0, 0.77459669]
12
13 elif n == 4:
14     c = [0.3478548, 0.6521452, 0.6521452, 0.3478548]
15     x = [-0.861136312, -0.339981044, 0.339981044, 0.861136312]
16
17 elif n == 5:
18     c = [0.2369269, 0.4786287, 0.5688889, 0.4786287, 0.2369269]
19     x = [-0.906179846, -0.538469310, 0, 0.538469310, 0.906179846]
20 else:
21     print('n = 3,4,5')
22
23 for j in range(0,n):
24     g[j] = (1-(x[j]+1)/2)**(1/6)*(x[j]+1)*9*math.pi
25     i += c[j]*g[j]
26 print(i)
```

-상미분방적식 풀이 3가지 방법비교

Using (a) Euler method, (b) Heun's method, and (c) 4th order Runge-Kutta method, evaluate x, y for the range of $[0,2]$. Step size t is 0.1.

Compare your results with exact solution.

$$\begin{aligned}x' &= 2x - 3y, \quad x(0) = 8 \\ y' &= y - 2x, \quad y(0) = 3\end{aligned}$$

Exact Solution

$$\begin{aligned}x &= 5e^{-t} + 3e^{4t} \\ y &= 5e^{-t} - 2e^{4t}\end{aligned}$$

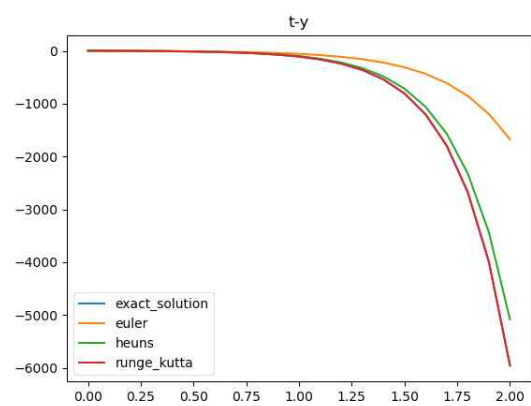
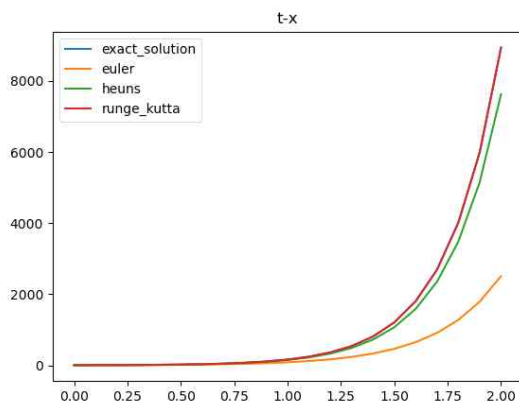
```
1 import math
2 import numpy as np
3
4 t = np.arange(0,2.1,0.1)
5 x = np.zeros(t.size)
6 y = np.zeros(t.size)
7 x[0] = 8
8 y[0] = 3
9
10 def f1(t):
11     return 2*x[t]-3*y[t]
12 def f2(t):
13     return y[t]-2*x[t]
14
15 for i in range(0,t.size-1):
16     x[i+1] = x[i] + f1(i)*0.1
17     y[i+1] = y[i] + f2(i)*0.1
18
19 print(x,y)
```

```
1 import math
2 import numpy as np
3
4 t = np.arange(0,2.1,0.1)
5 xp = np.zeros(t.size)
6 yp = np.zeros(t.size)
7 x = np.zeros(t.size)
8 y = np.zeros(t.size)
9 x[0] = 8
10 y[0] = 3
11
12 def f1(x,y):
13     return 2*x-3*y
14 def f2(x,y):
15     return y-2*x
16
17 for i in range(0,t.size-1):
18     xp[i+1] = x[i] + f1(x[i],y[i])*0.1
19     yp[i+1] = y[i] + f2(x[i],y[i])*0.1
20     x[i+1] = x[i] +(f1(x[i],y[i])+f1(xp[i+1],yp[i+1]))*0.1/2
21     y[i+1] = y[i] +(f2(x[i],y[i])+f2(xp[i+1],yp[i+1]))*0.1/2
22
23 print(x,y)
```

```

1 import math
2 import numpy as np
3 import matplotlib.pyplot as plt
4 x = np.arange(0,10.1,0.1)
5 y = np.zeros(x.size)
6 z = np.zeros(x.size)
7 y[0] = 40
8 z[0] = 12.6907
9
10 def f1(x,y,z):
11     return z
12 def f2(x,y,z):
13     return 0.01*(y-20)
14 def runge(x,y,z):
15     for i in range(0,x.size-1):
16         kx1 = f1(x[i],y[i],z[i])
17         ky1 = f2(x[i],y[i],z[i])
18         kx2 = f1(x[i]+0.1/2,y[i]+kx1*0.1/2,z[i]+ky1*0.1/2)
19         ky2 = f2(x[i]+0.1/2,y[i]+kx1*0.1/2,z[i]+ky1*0.1/2)
20         kx3 = f1(x[i]+0.1/2,y[i]+kx2*0.1/2,z[i]+ky2*0.1/2)
21         ky3 = f2(x[i]+0.1/2,y[i]+kx2*0.1/2,z[i]+ky2*0.1/2)
22         kx4 = f1(x[i]+0.1/2,y[i]+kx3*0.1,z[i]+ky3*0.1)
23         ky4 = f2(x[i]+0.1/2,y[i]+kx3*0.1,z[i]+ky3*0.1)
24         y[i+1] = y[i] + (kx1+2*kx2+2*kx3+kx4)*0.1/6
25         z[i+1] = z[i] + (ky1+2*ky2+2*ky3+ky4)*0.1/6
26         print(x,y,z)
27 tt = []
28 for i in x:
29     ttt = 73.4523*math.exp(0.1*i)-53.4523*math.exp(-0.1*i)+20
30     tt.append(ttt)
31 runge(x,y,z)
32 plt.plot(x,y,label='runekutta')
33 plt.plot(x,tt,label='analytical')
34 plt.title('HW12')
35 plt.legend()
36 plt.show()

```



②공학주제(일반)

a.인공지능응용기계공학

-Regularized Linear Regression

Programming assignment 5 : Regularized Linear Regression and Bias v.s. Variance

기계공학과
16011504 임진욱

1. Regularized Linear Regression

이번 프로그래밍은 저수지의 수위 변화로 댐에서 흘러나오는 물의 양을 예측하는 선형회귀를 시행한다.

```
function [J, grad] = linearRegCostFunction(X, y, theta, lambda)
    m = length(y); % number of training examples

    h = X * theta; % 12 X 1
    J = (sum((h-y).^2) + lambda*sum(theta(2:end).^2))/(2*m);
    grad = (X'*(h-y) + lambda*[0;theta(2:end)])/m;
    grad = grad(:);
end
```

Cost at theta = [1 ; 1]: 303.993192
(this value should be about 303.993192)

Gradient at theta = [1 ; 1]: [-15.303016; 598.250744]
(this value should be about [-15.303016; 598.250744])

위와 같이 linear regression의 costfunction, gradient를 구하는 linearRegCostFunction 함수를 작성하고 주어진 값과 비교했을 때 일치함.

```

function [theta] = trainLinearReg(X, y, lambda)

% Initialize Theta
initial_theta = zeros(size(X, 2), 1);

% Create "short hand" for the cost function to be minimized
costFunction = @(t) linearRegCostFunction(X, y, t, lambda);

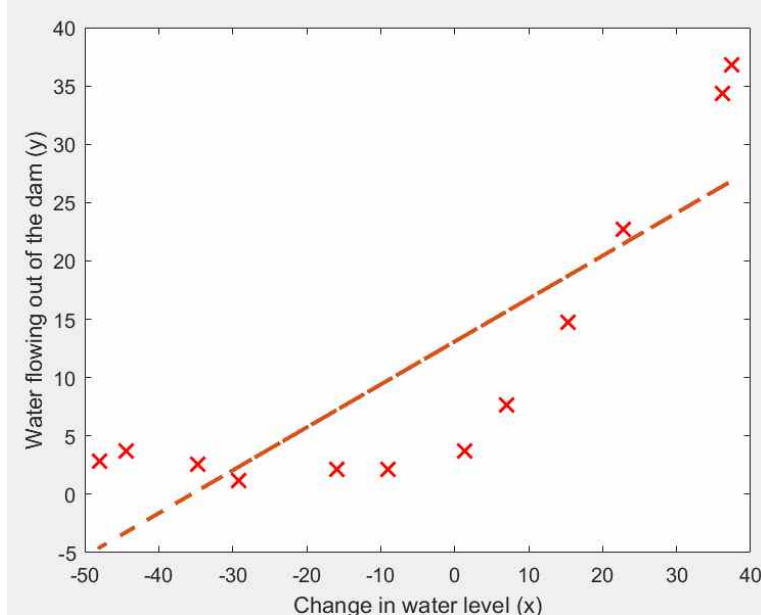
% Now, costFunction is a function that takes in only one argument
options = optimset('MaxIter', 200, 'GradObj', 'on');

% Minimize using fmincg
theta = fmincg(costFunction, initial_theta, options);

end

```

fmincg 함수를 사용해 최적화된 theta를 찾는 trainLinearReg 함수.



그렇게 찾은 theta로 그래프에 나타내면 위 그래프와 같다.

2. Bias-variance

Training example의 수와 error의 관계를 나타내고자 한다.

Train error와 validation error를 구하는 함수 learningCurve를 아래와 같이 작성.

```

function [error_train, error_val] = ...
    learningCurve(X, y, Xval, yval, lambda)

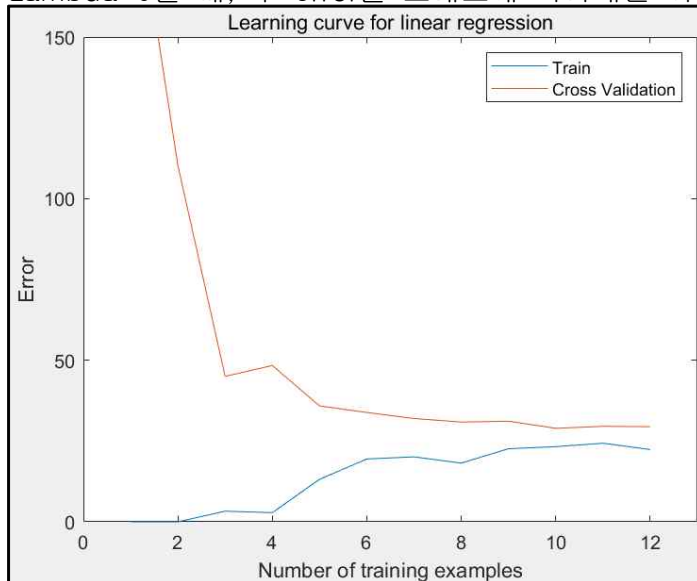
m = size(X, 1);

error_train = zeros(m, 1);
error_val    = zeros(m, 1);

for i = 1:m
    theta = trainLinearReg(X(1:i,:), y(1:i), lambda);
    error_train(i) = linearRegCostFunction(X(1:i,:), y(1:i), theta, lambda);
    error_val(i) = linearRegCostFunction(Xval, yval, theta, lambda);
end
end

```

Lambda=0일 때, 두 error를 그래프에 나타내면 아래와 같다.



3. Polynomial regression

Feature의 수를 늘리기 위한 함수 polyFeatures 함수

```

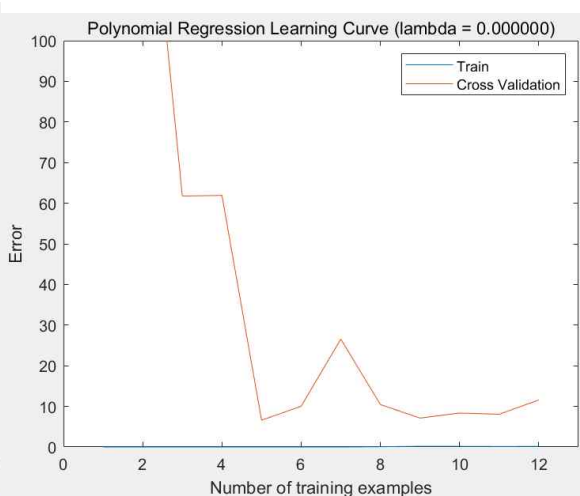
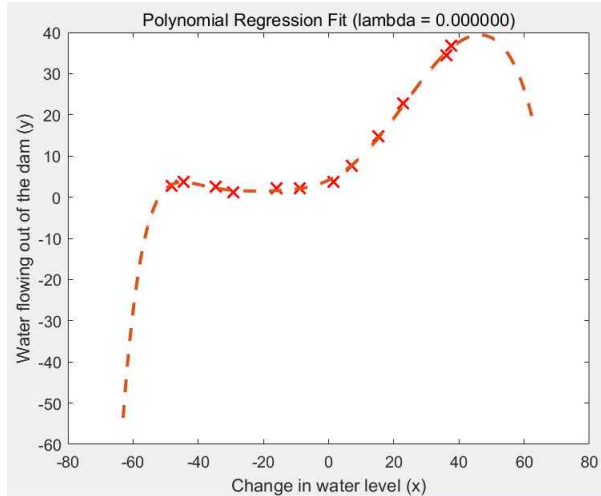
function [X_poly] = polyFeatures(X, p)

X_poly = zeros(numel(X), p);

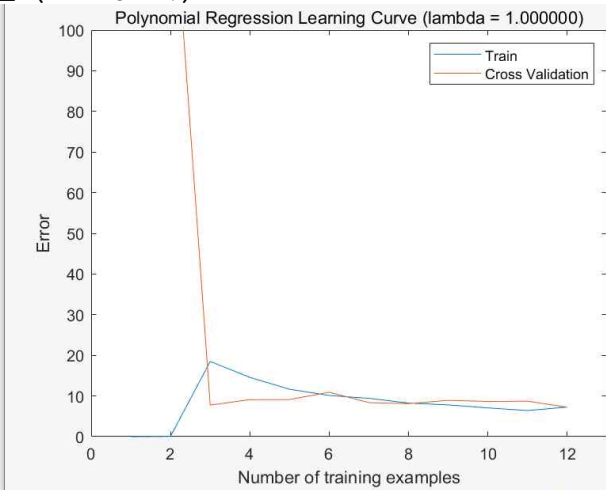
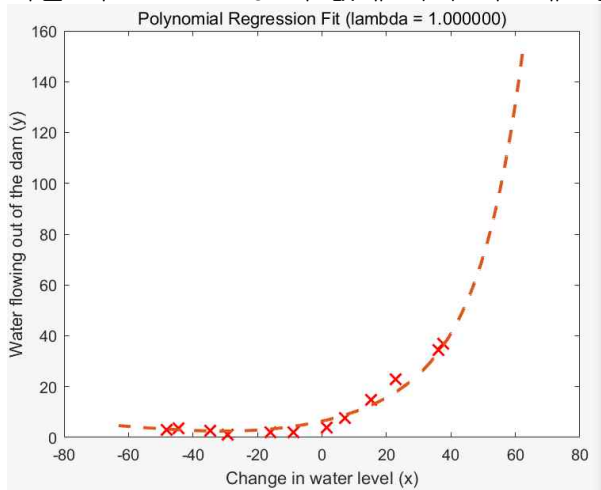
for i = 1:p
    X_poly(:,i) = X.^i;
end
end

```

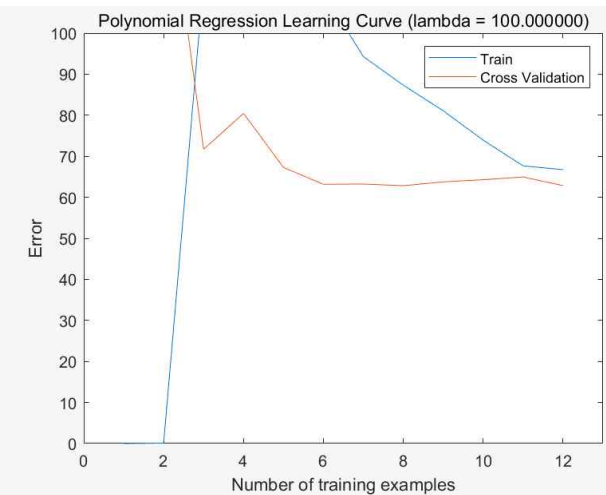
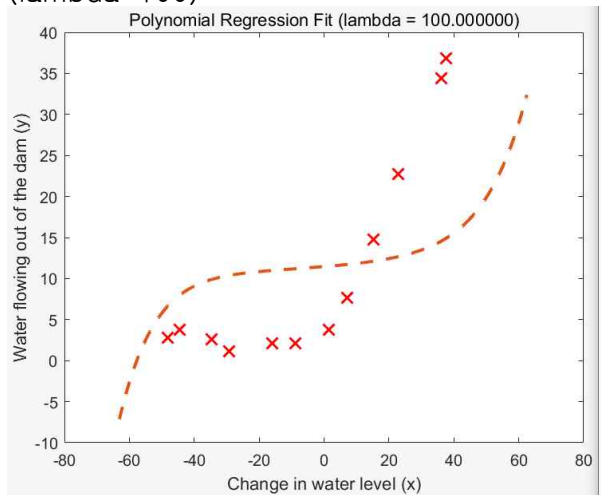
이를 이용해 feature의 수를 8개로 하고 trainLinearReg를 통해 theta값을 구해 그래프에 regression함수와 error를 표시했다. (lambda=0)



이때 overfitting됨을 볼 수 있다.
이런 식으로 λ 의 값에 따라 비교해보면 ($\lambda=1$)



($\lambda=100$)



* λ 의 값에 따라 error의 값을 비교하기 위해 validationCurve함수를 사용한다.

```

function [lambda_vec, error_train, error_val] = ...
    validationCurve(X, y, Xval, yval)

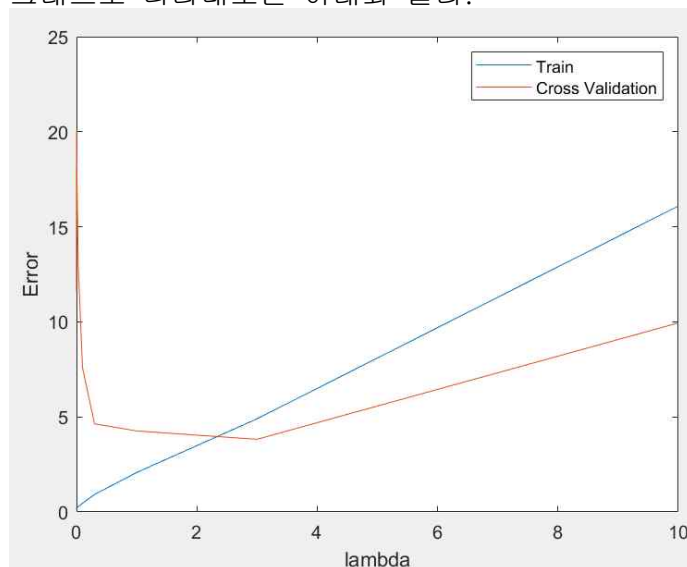
lambda_vec = [0 0.001 0.003 0.01 0.03 0.1 0.3 1 3 10]';

error_train = zeros(length(lambda_vec), 1);
error_val = zeros(length(lambda_vec), 1);

for i = 1:length(lambda_vec)
    lambda = lambda_vec(i);
    theta = trainLinearReg(X,y,lambda);
    error_train(i) = linearRegCostFunction(X, y, theta, 0);
    error_val(i) = linearRegCostFunction(Xval,yval,theta,0);
end
end

```

그래프로 나타내보면 아래와 같다.



TensorFlow assignment 2

기계공학과
16011504 임진욱

- 숫자를 인식하는 문제로 SVHN dataset을 사용한다. 이미지는 32x32 pixel로 RGB로 되어있는 이미지이다.

1. Data load

주어진 dataset은 train_32x32.mat/test_32x32.mat 두가지로 다운받아 구글드라이브에 저장하고 mat파일을 불러오는 코드 작성함

```
[2] # Run this cell to connect to your Drive folder

from google.colab import drive
drive.mount('/content/gdrive')

Mounted at /content/gdrive

[3] # Load the dataset from your Drive folder

train = loadmat('/content/gdrive/My Drive/train_32x32.mat')
test = loadmat('/content/gdrive/My Drive/test_32x32.mat')
```

2. Inspect and preprocess the dataset

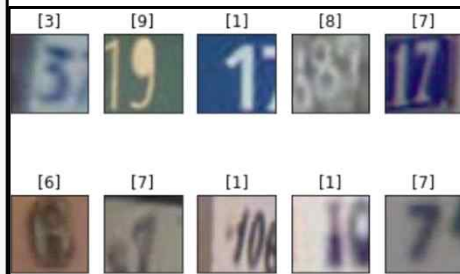
- mat파일 데이터로부터 X_train, y_train, X_test, y_test를 받아오고 크기를 확인함. Train data 73257개와 Test data 26032개 그리고 32x32 pixel, 3channel임을 확인할 수 있다.

```
X_train=train['X']
y_train=train['y']
X_test=test['X']
y_test=test['y']
#####
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)

(32, 32, 3, 73257) (73257, 1)
(32, 32, 3, 26032) (26032, 1)
```

- train data에서 랜덤으로 골라 color이미지로 나타내 보았다.

```
def plot_images(img, labels, nrows, ncols):
    fig, axes = plt.subplots(nrows, ncols)
    for i, ax in enumerate(axes.flat):
        j = random.randint(0,X_train.shape[3])
        if img[:, :, :, j].shape == (32, 32, 3):
            ax.imshow(img[:, :, :, j])
        else:
            ax.imshow(img[:, :, 0, j], cmap='gray')
        ax.set_xticks([]); ax.set_yticks([])
        ax.set_title(labels[j])
    plot_images(X_train, y_train, 2, 5)
```

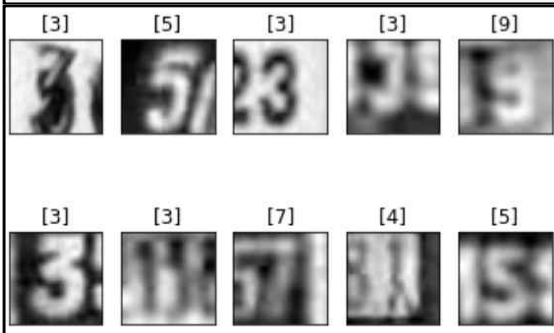


- color이미지를 흑백으로 바꿔 주기위해 channel 3개를 평균 내어 저장한 뒤 흑백이미지를 plot함

```
def rgb2gray(images):
    return np.expand_dims(np.mean(images/255,axis=2),axis=2)

X_train = rgb2gray(X_train)
X_test = rgb2gray(X_test)

plot_images(X_train,y_train,2,5)
```



3. MLP neural network classifier

- 우선 X data를 transpose시킴

```
X_train= X_train.transpose((3,0,1,2))
X_test= X_test.transpose((3,0,1,2))
#####
X_train.shape, X_test.shape
```

- MLP model에 flat, dense layer만을 사용하고 activation은 relu를 사용하였으며 마지막 layer만 softmax를 사용하여 구성하였다. Input shape은 32x32x1로 주었다.

```
def get_model(input_shape):
    model = Sequential([
        Flatten(input_shape=input_shape),
        Dense(512, activation = "relu"),
        Dense(128, activation = "relu"),
        Dense(64, activation = "relu"),
        Dense(10, activation = "softmax"),
    ])
    return model

model = get_model(X_train[0].shape)
```

Model을 summary하고 확인해보았다. (parameter가 599370개)

```
model.summary()
```

```
Model: "sequential_13"
```

Layer (type)	Output Shape	Param #
flatten_13 (Flatten)	(None, 1024)	0
dense_54 (Dense)	(None, 512)	524800
dense_55 (Dense)	(None, 128)	65664
dense_56 (Dense)	(None, 64)	8256
dense_57 (Dense)	(None, 10)	650

```
Total params: 599,370
```

```
Trainable params: 599,370
```

```
Non-trainable params: 0
```

- y를 one vs all방법으로 바꾸어 준다.

```
with tf.device('/device:GPU:0'):  
    # Softmax를 타고 나온 output과의 shape를 맞추기위한 One-hot encoding  
    y_train_one_hot = tf.keras.utils.to_categorical(np.array(y_train))  
    y_test_one_hot = tf.keras.utils.to_categorical(np.array(y_test))  
  
    # 인코딩된 값에서 무의미한 값인 첫번째 열 삭제  
    y_train_one_hot = np.delete(y_train_one_hot, 0, axis = 1)  
    y_test_one_hot = np.delete(y_test_one_hot, 0, axis = 1)
```

- callback을 사용하는데 ModelCheckpoint와 EarlyStopping으로 정한 뒤 적용하였다.

```
def get_checkpoint_every_epoch():  
  
    checkpoint_path = 'model_checkpoint/checkpoint'  
    checkpoint = ModelCheckpoint(filepath = checkpoint_path,  
                                frequency='epoch',  
                                save_weights_only=True,  
                                verbose=1)  
  
    return checkpoint  
  
def get_early_stopping():  
    early_stopping = EarlyStopping(monitor='val_accuracy', patience=3)  
    return early_stopping  
  
checkpoint = get_checkpoint_every_epoch()  
early_stopping = get_early_stopping()
```

- optimizer = adam, loss = categorical_crossentropy, metrics = accuracy를

사용하여 컴파일 하고 epochs=30, validation=0.15, batchsize=64로 fit시킴

```
def get_early_stopping():
    early_stopping = EarlyStopping(monitor='val_accuracy', patience=3)
    return early_stopping

checkpoint = get_checkpoint_every_epoch()
early_stopping = get_early_stopping()

def compile_model(model):
    # optimizer Adam사용
    optimizer = optimizers.Adam()
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
    compile_model(model)
```

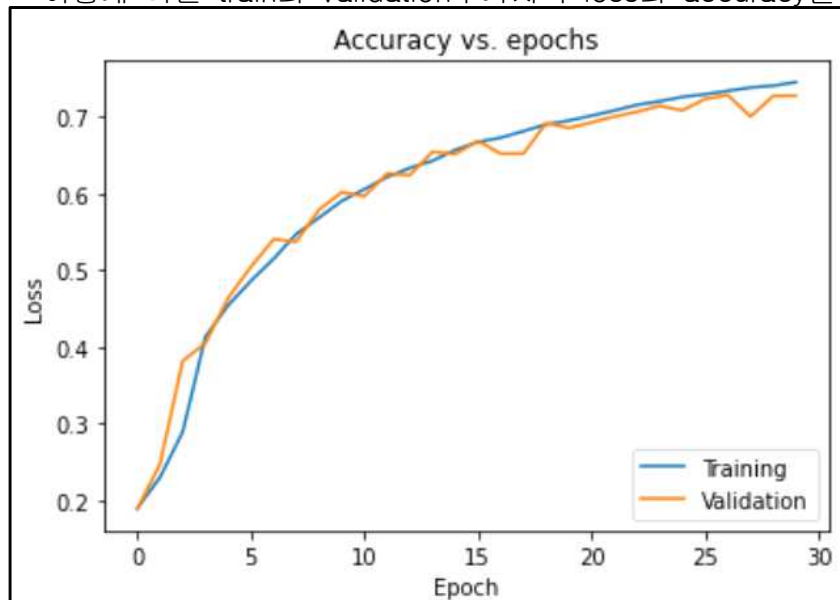
- epochs를 30으로 하고 마지막을 확인했을 때 train data의 loss는 0.8129 로 1보다는 작았으나 충분하지 않다고 생각하고 accuracy는 0.7455로 정확도도 부족하다.

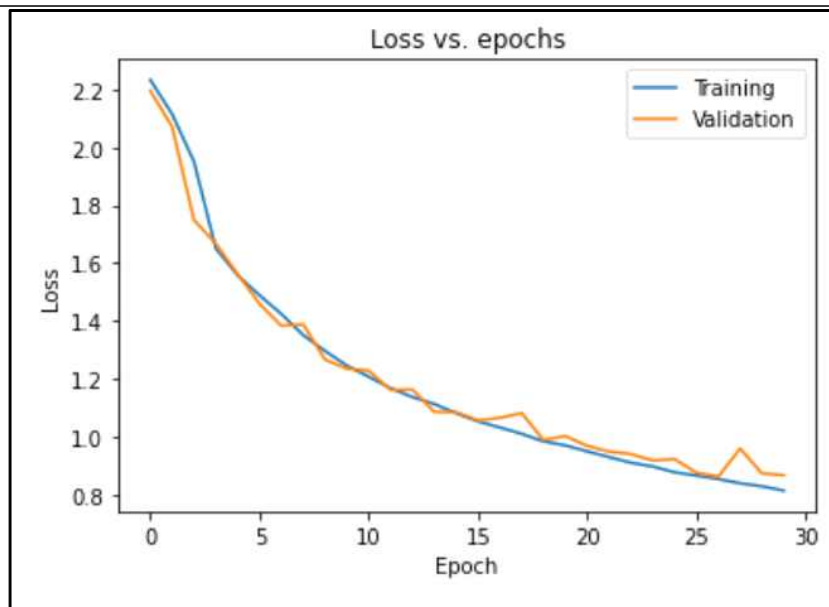
Epoch 30/30

Epoch 00030: saving model to model_checkpoint/checkpoint

973/973 - 8s - loss: 0.8129 - accuracy: 0.7455 - val_loss: 0.8659 - val_accuracy: 0.7279

- 이렇게 나온 train과 validation두가지의 loss와 accuracy를 서로 비교해본다.





4. CNN neural network classifier

- CNN model에 Conv2D, MaxPool2D, Dropout, BatchNormalization, Dense layer를 적용하여 작성하였다.

```
def get_model(input_shape):
    model = Sequential([
        Conv2D(8, (3,3), activation='relu', input_shape=input_shape, ),
        BatchNormalization(),
        Conv2D(16, (3,3), activation='relu'),
        MaxPool2D((2, 2)),
        Dropout(0.3),
        BatchNormalization(),
        Conv2D(32, (3,3), activation='relu'),
        MaxPool2D((2, 2)),
        Dropout(0.3),
        BatchNormalization(),
        Flatten(),
        Dense(64, activation='relu'),
        Dense(32, activation='relu'),
        Dense(10, activation='softmax')
    ])
    return model
model = get_model(X_train[0].shape)
```

Model: "sequential_26"		
Layer (type)	Output Shape	Param #
=====		
conv2d_25 (Conv2D)	(None, 30, 30, 8)	80
batch_normalization_14 (Batch Normalization)	(None, 30, 30, 8)	32
conv2d_26 (Conv2D)	(None, 28, 28, 16)	1168
max_pooling2d_22 (MaxPooling2D)	(None, 14, 14, 16)	0
dropout_17 (Dropout)	(None, 14, 14, 16)	0
batch_normalization_15 (Batch Normalization)	(None, 14, 14, 16)	64
conv2d_27 (Conv2D)	(None, 12, 12, 32)	4640
max_pooling2d_23 (MaxPooling2D)	(None, 6, 6, 32)	0
dropout_18 (Dropout)	(None, 6, 6, 32)	0
batch_normalization_16 (Batch Normalization)	(None, 6, 6, 32)	128
flatten_26 (Flatten)	(None, 1152)	0
dense_91 (Dense)	(None, 64)	73792
dense_92 (Dense)	(None, 32)	2080
dense_93 (Dense)	(None, 10)	330
=====		
Total params: 82,314		
Trainable params: 82,202		
Non-trainable params: 112		

- CNN model의 parameter개수는 82000개 정도로 앞선 MLP model의 parameter개수보다 적은 수이다.
- optimizer, loss, metrics는 MLP에 사용된 것과 같은 것을 사용하였다.

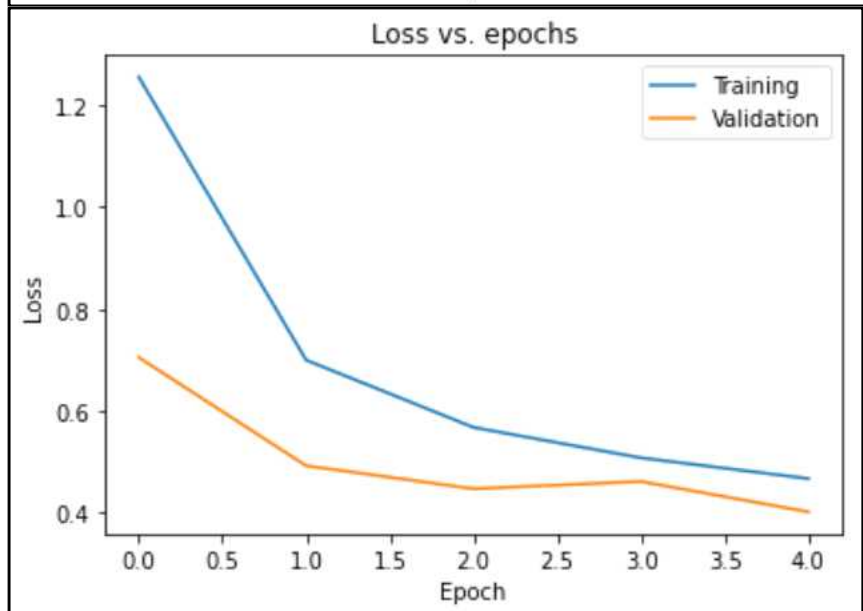
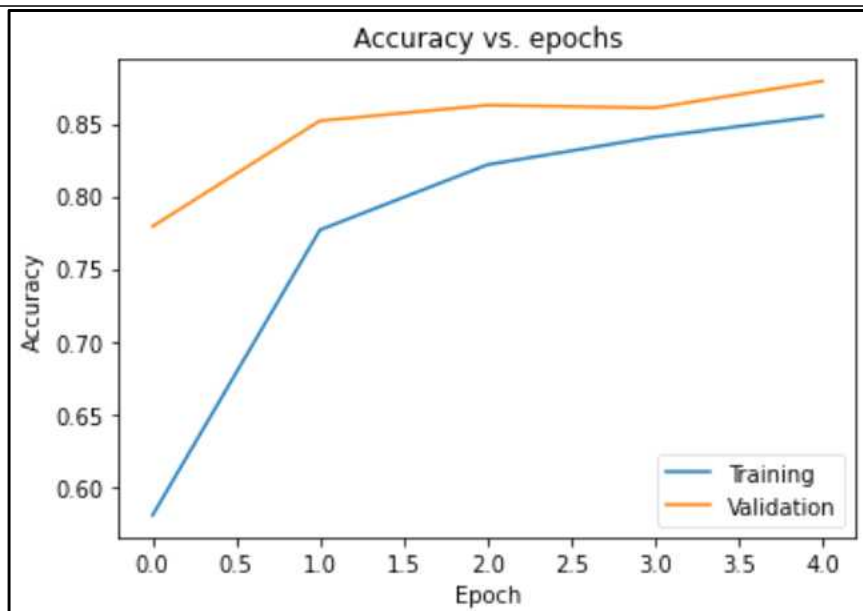
```
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

history=model.fit(X_train, y_train_one_hot, epochs=5, validation_split=0.15, verbose=2)
```

Epochs=5로 설정하고 실행한 뒤 loss와 accuracy를 확인해 보았다.

```
Epoch 1/5
1946/1946 - 76s - loss: 1.2554 - accuracy: 0.5801 - val_loss: 0.7051 - val_accuracy: 0.7791
Epoch 2/5
1946/1946 - 76s - loss: 0.6998 - accuracy: 0.7768 - val_loss: 0.4924 - val_accuracy: 0.8519
Epoch 3/5
1946/1946 - 79s - loss: 0.5676 - accuracy: 0.8217 - val_loss: 0.4473 - val_accuracy: 0.8627
Epoch 4/5
1946/1946 - 76s - loss: 0.5079 - accuracy: 0.8408 - val_loss: 0.4616 - val_accuracy: 0.8609
Epoch 5/5
1946/1946 - 79s - loss: 0.4672 - accuracy: 0.8554 - val_loss: 0.4019 - val_accuracy: 0.8792
```

- Train data의 loss는 0.4672, accuracy는 0.8554로 MLP의 loss, accuracy보다 조금 증가한 것을 볼 수 있다.
- 이렇게 나온 train과 validation두가지의 loss와 accuracy를 서로 비교해보았다.



5. Get model predictions

-MLP와 CNN의 prediction하는 코드 작성

```
def model_prediction(model):
    num_test_images = X_test.shape[0]

    random_inx = np.random.choice(num_test_images, 5)
    random_test_images = X_test[random_inx, ...]
    random_test_labels = y_test[random_inx, ...]

    predictions = model.predict(random_test_images)

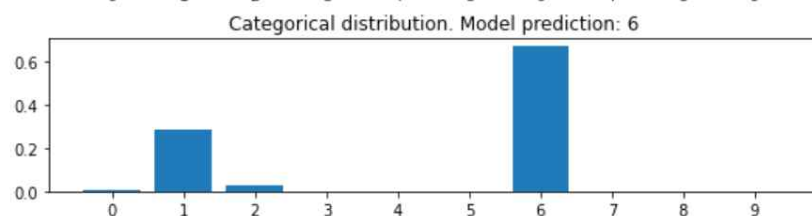
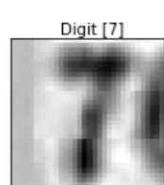
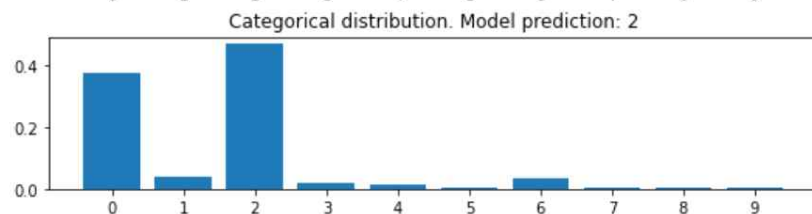
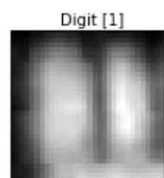
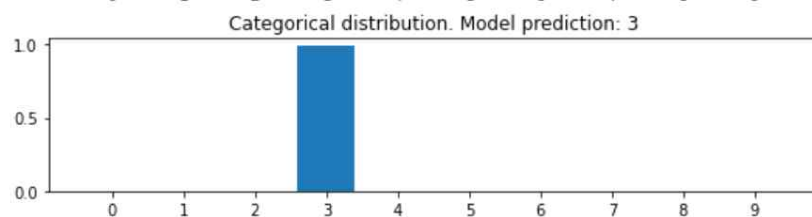
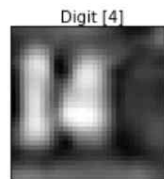
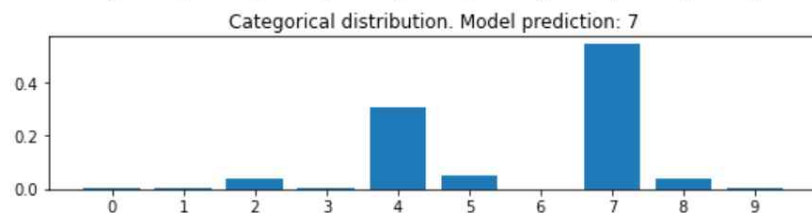
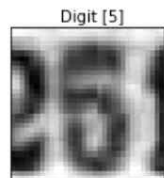
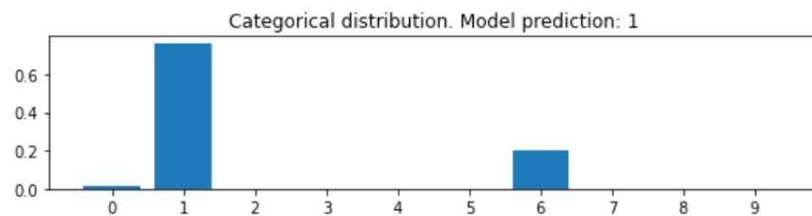
    fig, axes = plt.subplots(5, 2, figsize=(16, 12))
    fig.subplots_adjust(hspace=0.4, wspace=-0.2)

    for i, (prediction, image, label) in enumerate(zip(predictions, random_test_images, random_test_labels)):
        axes[i, 0].imshow(np.squeeze(image), cmap='gray')
        axes[i, 0].get_xaxis().set_visible(False)
        axes[i, 0].get_yaxis().set_visible(False)
        axes[i, 0].text(10., -1.5, f'Digit {label}')
        axes[i, 1].bar(np.arange(len(prediction)), prediction)
        axes[i, 1].set_xticks(np.arange(len(prediction)))
        axes[i, 1].set_title(f"Categorical distribution. Model prediction: {np.argmax(prediction)}")

    plt.show()
```

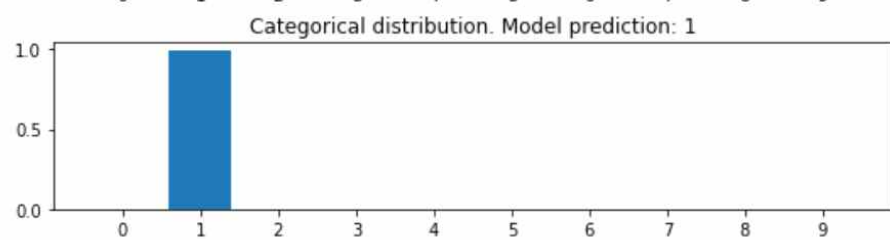
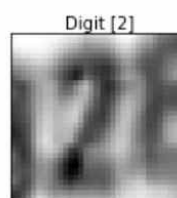
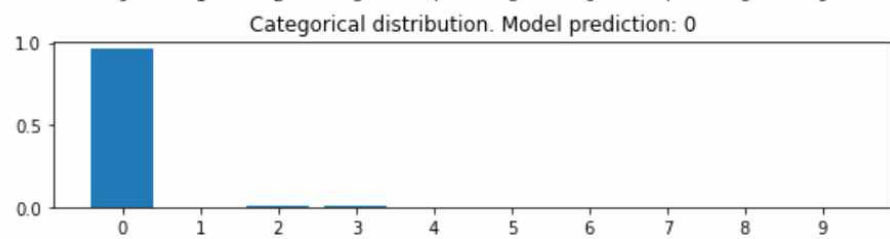
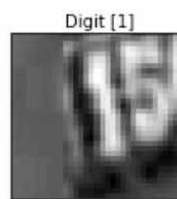
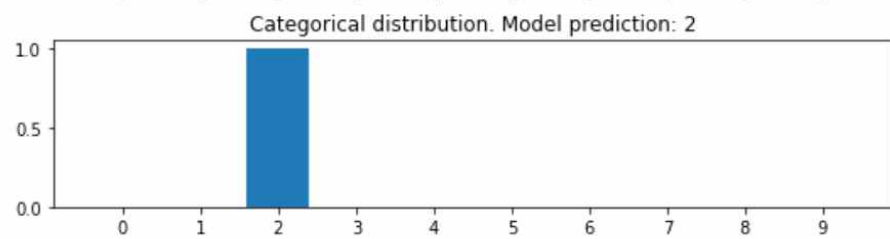
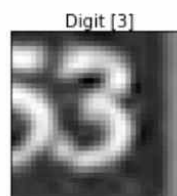
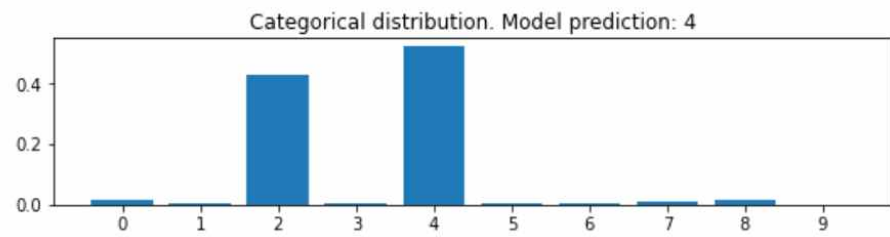
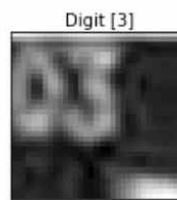
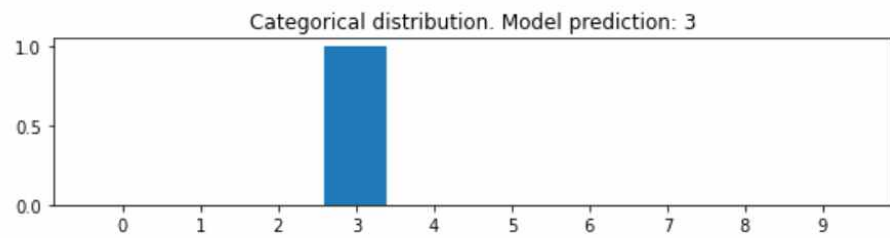
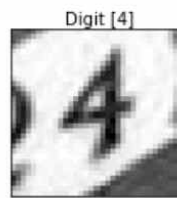
Test data에서 랜덤으로 5개를 골라 plot시키고 옆에 예측분포를 나타냈다.
-MLP model

model_prediction(MLP)



-CNN model

model_prediction(CNN)



인공지능응용 기계공학 텀프로젝트

세종 랜드마크 챌린지
(Sejong Landmark Challenge)

4조 14011541 김 형준
16011504 임 진욱
18011230 배 대원

2020. 12. 17 최종발표

목차

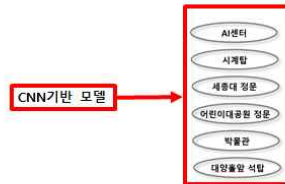
- I Sejong Landmark Challenge (SLC)
- II SLC Dataset
- III 작업 환경
- IV 파이프라인
- V 모델학습
- VI Neptune 시각화
- VII 모델 평가
- VIII 개선 방안
- IX 참고 자료

I. Sejong Landmark Challenge (SLC)



Sejong Landmark Challenge란?

- 세종대학교 건물 이미지를 분류하는 알고리즘 제작



이미지를 6개의 라벨로 분류

3

II. SLC Dataset



4

II. SLC Dataset

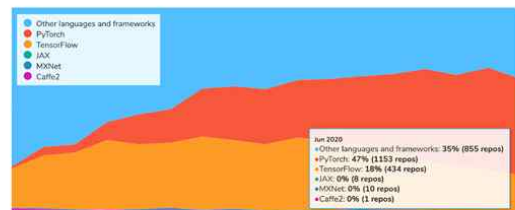


Test set에 회전, 광도, 스케일, 방해물 등 노이즈를 넣어 Challenging하게 구성

5

III. 작업 환경

텐서플로우 vs 파이토치

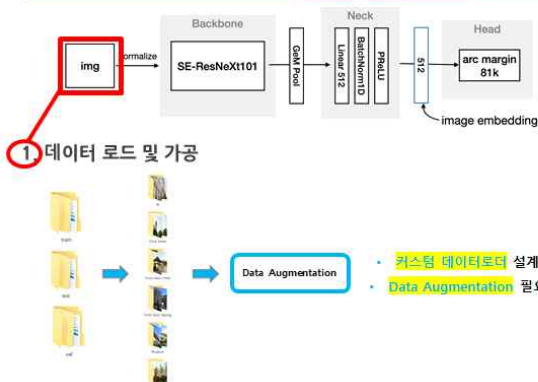


Github상 논문 구현언어 파이토치 점유율 [11]

→ 이와 더불어 디버깅상의 이점으로 파이토치 사용

6

IV. 파이프라인



7

IV. 파이프라인

(1) 커스텀 데이터로더

```
class Custom_dataloader(Dataset):
    def __init__(self, mode, datapath, img_w, img_h, transform=None):
        → 클래스 호출시 실행되는 Init함수

    def full_load(self):
        → 이미지가 저장된 폴더로부터 전체이미지 불러오기

    def csv_exist(self):
        → 이미지가 저장된 csv파일이 있으면 호출

    def __len__(self):
        return len(self.images)
        → 저장된 이미지의 길이 반환

    def __getitem__(self, idx):
        data = { 'image': img, 'label': label }
        return data
        → index값을 받아 해당 index의 이미지와 라벨을 딕셔너리 형태로 반환
```

8

IV. 파이프라인

(2) Data Augmentation

	A 0.4.2	Imgaug 0.3.0	Torchvision 0.4.1	Keras 2.3.1	Augmentor 0.2.6	Solt 0.1.8
HorizontalFlip	2183	1403	1757	1068	1779	1031
VerticalFlip	4217	2334	1538	4196	1541	3820
Rotate	456	368	163	32	60	116
ShiftScaleRotate	800	549	146	34	-	-
Brightness	2209	1288	405	211	403	2070
Contrast	2215	1387	338	-	337	2073
BrightnessContrast	2208	740	193	-	193	1060
ShiftRGB	2214	1303	-	407	-	-
ShiftHSV	468	443	61	-	-	144
Gamma	2281	-	730	-	-	925
Grayscale	5019	436	788	-	1451	4191
RandomCrop64	173,877	3340	43,792	-	36,869	36,178
PadToSize512	2906	-	553	-	-	2711
Resize512	663	506	968	-	954	673
RandomSizeCrop64_512	2565	933	1395	-	1353	2360
Equalize	759	457	-	-	684	-

[4] 논문 참고 ImageNet의 validation set 2000장에 대한
→ 이미지 처리속도가 가장 빠른 **Albumentations** 사용

9

IV. 파이프라인

(2) Data Augmentation

Torchvision 코드

```
torchvision_transform = transforms.Compose([
    transforms.Resize(256, 256),
    transforms.RandomCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
])
```



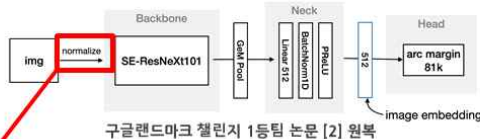
Albumentations 코드

```
albumentations_transform = albumentations.Compose([
    albumentations.Resize(256, 256),
    albumentations.RandomCrop(224, 224),
    albumentations.HorizontalFlip(),
    albumentations.pytorch.transforms.ToTensor()
])
```

→ **torchvision** 라이브러랑 호환성이 좋음

10

IV. 파이프라인



구글랜즈마크 챌린지 1등 팀 논문 [2] 원복

2. Standardization으로 데이터 표준화

$$\frac{x - \mu}{\sigma} \quad (\mu: \text{평균}, \sigma: \text{표준편차})$$

Train + validation 데이터 CSV파일로 가공

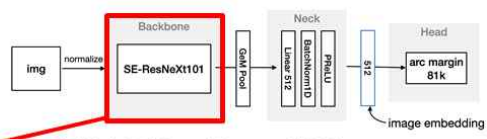
numpy사용 평균과 표준편차

→ 평균: [124.5, 125.58, 118.67]

→ 표준편차: [70.97, 70.39, 78.2]

11

IV. 파이프라인



3. Timm 라이브러리 사용 Backbone 교체실험

참고사항

① 구글랜즈마크 1등 팀 논문 [2]

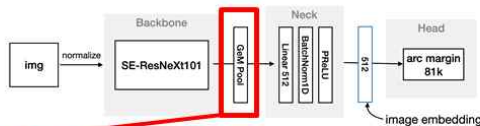
② ImageNet 기준 모델 순위 [5]

③ 코랩 작업환경 고려 실험모델 선정

→ [5]참고 ImageNet 기준 상위에 랭크된 모델 중 너무 무겁지않은 모델들 사용

12

IV. 파이프라인



4. GeM Pooling 사용

$$\mathbf{f}^{(g)} = [f_1^{(g)} \dots f_k^{(g)} \dots f_K^{(g)}]^\top, \quad f_k^{(g)} = \left(\frac{1}{|\mathcal{X}_k|} \sum_{x \in \mathcal{X}_k} x^{p_k} \right)^{\frac{1}{p_k}}$$

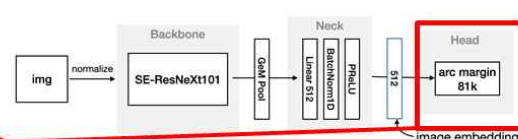
$p_k \rightarrow \infty$: Max pooling

$p_k = 1$: Average pooling

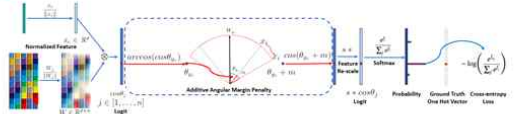
미분가능 → p_k 학습가능

13

IV. 파이프라인



5. Arc margin 사용

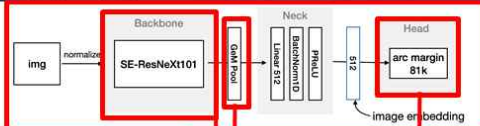


[9] 논문 참고 Arc margin 부분적 구현

State of the Art(SOTA) 논문들에 주로 등장

14

IV. 파이프라인



class ArcMarginProduct(nn.Module):
Arc margin

class GeM(nn.Module):
Trainable GeM Pooling

class BackboneNet(nn.Module):
Backbone network 선언

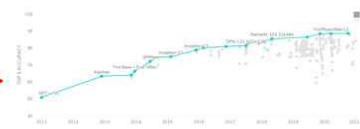
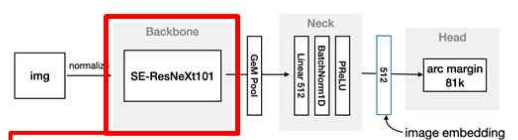
class Net(nn.Module):
def __init__(self, ...):
위 클래스를 불러오기
Neck 설계 (sequential 이용)
def forward(self, image, ...):
이미지 불러온 클래스를 통해
return x # 통과한 결과 반환

총 4개의 class로 분할 (객체지향)
↓
Net class에서 앞의 3개의 class를 불러옴
Feedforward 연산

15

V. 모델 학습

[5] 모델 순위, [12] 논문 기반 Backbone 선정



ImageNet 기준 모델 성능 순위[5]

16

V. 모델 학습

Freezing을 이용한 학습기법



17

V. 모델 학습

Freezing을 이용한 학습기법 (코드)

1. 모델의 파라미터를 확인

```
names_param=[] (파라미터들의 이름을 저장할 리스트)
for name_param, _ in model.named_parameters():
    names_param.append(name_param)
print(names_param)
```
2. names_param에 저장된 파라미터 중 맨 뒤에서 9개만 빼고 나머지 Freeze

```
for name_param, param in model.named_parameters():
    param.requires_grad = False
    if name_param in names_param[-9:]:
        param.requires_grad = True
```

global_pool.p, 'neck.0.weight', 'neck.0.bias', 'neck.1.weight', 'neck.1.bias', 'neck.2.weight', 'neck.3.weight', 'neck.3.bias', 'head.weight'

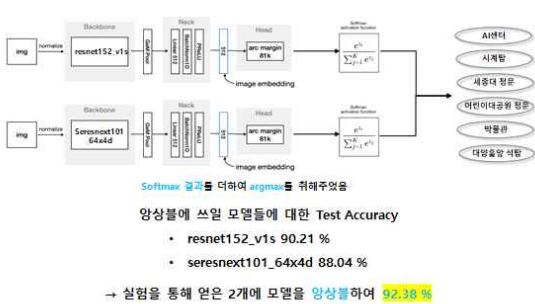
해당 파라미터 제외 Freeze



18

V. 모델 학습

양상불



19

VI. Neptune 시각화



Neptune의 장점

- 실험조건, 매트릭 백업 및 트래킹
- 코드 백업
- GPU, CPU 사용량 기록
- 에러메시지 기록
- 원하는 변수로 실시간 Plotting
- 다양한 visualization tool과 연동가능
- 코랩환경에서 코업자와 교류가능



20

VI. Neptune 시각화

사용방법 소개

1. 초기화

```
import neptune
# Neptune parameters
api_token="ANONYMOUS",
project_qualified_name='사용자명/저장소이름'
api_token='API 토큰 번호'
upload_source_files = '파일명.py' #백업할 파일 지정
# Neptune initialize
neptune.init(
    api_token=api_token,
    project_qualified_name=project_qualified_name,
    name,
)
```

넵톤 공식사이트에서 회원가입 및 토큰발급 필요



21

VI. Neptune 시각화

사용방법 소개

2. 실험생성 및 실험변수 업데이트

```
def create_exp(name, params, upload_source_files):
    neptune.create_experiment(
        name=name,
        params=params,
        upload_source_files = upload_source_files
    )
```

3. 태그생성 및 매트릭 트래킹

```
def create_tag(tag):
    return neptune.append_tags(tag)

def create_log_metric(name, val):
    return neptune.log_metric(name, val)
```

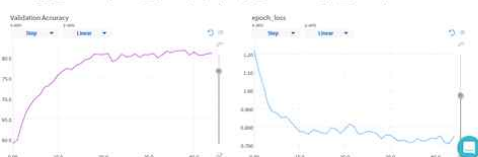
[1] <https://neptune.ai> 에서 튜토리얼 제공 (동영상도 있음)

22

VI. Neptune 시각화



실험 날짜, 조건, 변수, 주석 등 넵톤에 업데이트



원하는 변수 Tracing 및 실시간 plot

23

VII. 모델평가

정량적 평가

실험No.	모델명	주요 변동사항	Test Acc
1	gluon_resnext101_64x4d	Standardization 적용	90.22 %
2	gluon_resnext101_64x4d	Data augmentation 적용	67.39 %



지나친 Data Augmentation으로 성능 감소 발생
확률값 p 낮춤

실험No.	모델명	주요 변동사항	Test Acc
3	gluon_resnext101_64x4d	Data augmentation 수정	84.78 %
4	gluon_resnext101_64x4d	Data augmentation 수정 Normalize로 변경	81.52 %
5	gluon_seresnext101_64x4d	Standardization로 다시 변경 백본 모델 변경	88.04 %
6	gluon_seresnext101_64x4d	Augmentation 수정	86.96 %

24

VII. 모델평가

정량적 평가

실험No.	모델명	주요 변동사항	Test Acc
-------	-----	---------	----------

Efficientnet(8)은 Resolution과 Depth가 커야 성능이 좋음
But 코랩에서 GPU 메모리부족으로 성능↓

실험No.	모델명	주요 변동사항	Test Acc
8	gluon_resnet152_v1s	모델 변경, batch size 128	90.21 %
9	gluon_resnet152_v1s	[12]근거 Test set 사이즈 2배	73.91 %
10	gluon_resnet152_v1s + gluon_seresnext101_64x4d	확률합 기반 앙상블 (exp5+exp8)	92.39 %
11	gluon_resnet152_v1s + gluon_seresnext101_64x4d + gluon_resnext101_64x4d	확률합 기반 앙상블 (exp1+exp5+exp8)	92.39 %

25

VII. 모델평가

정량적 평가

실험No.	모델명	주요 변동사항	Test Acc
-------	-----	---------	----------



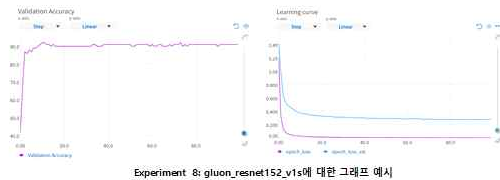
앙상블을 했음에도 성능이 더 떨어짐
∴ 성능이 낮은 모델과 높은 모델의 조합

실험결과: 가장 좋은 성능을 낸 모델은 "실험12"로 95.65% 기록

26

VII. 모델평가

정량적 평가



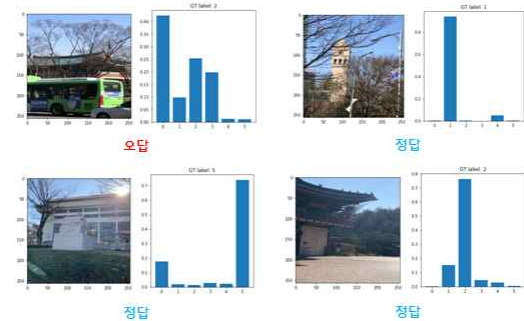
Experiment 8: gluon_resnet152_v1s에 대한 그래프 예시

- Validation loss를 기준으로 Best model을 저장
- 모든 실험에 대하여 파라미터, 그래프, 체크포인트, CSV파일을 저장
- Learning curve를 참고하여 추가학습여부를 결정

27

VII. 모델평가

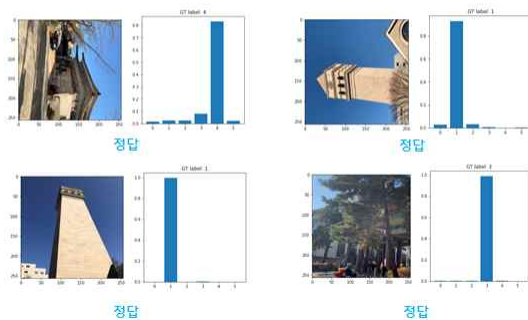
정성적 평가



28

VII. 모델평가

정성적 평가



29

VII. 모델평가

정성적 평가



30

VIII. 개선방안

개선 방안

- 다른 앙상블기법을 이용
 - 각 모델마다 나온 feature를 L2 norm 적용 후 concatenate하여 새로운 feature로 사용
 - 다른 앙상블기법 (Voting, Bagging, Boosting) 사용
- Softmax를 통한 분류가 아닌 SVM 등 분류기를 통한 feature 분류
- Cos similarity를 이용한 이미지 유사도 비교
 - Trainset에서 Non-landmark 이미지 제거
- Multi-GPU, 메모리 사용
 - 이미지 해상도, Depth, Batch size를 키워서 efficientnet 학습
 - Metric learning 기반 loss 사용시 batch size 영향도 있을거라 예측
- 데이터로더 설계시 파일경로만을 CSV로 가공하여 메모리 절약
 - 메모리 확보로 좀 더 다양한 실험가능
 - 데이터로더 선언부가 훨씬 빠르게 돌아감

31

참고 자료

앱인벤터로 개발한 안드로이드 어플 데모영상 (라이트모텔 오픈팅 컴퓨팅)



32

IX. 참고 자료

- [1] <https://nndune.ai> (논문 공식사이트)
- [2] Henkel, Christof and Philipp Singer. "Supporting large-scale image recognition with out-of-domain samples." arXiv preprint arXiv:2010.01650 (2020). (구글랜드마크 챌린지 2020 1등 논문)
- [3] <https://albumentations.readthedocs.io/en/latest/> (albumentation 라이브러리 다크언트)
- [4] Buslaev, Alexander, et al. "Albumentations: fast and flexible image augmentations." Information 11.2 (2020): 125. (data augmentation 관련 논문)
- [5] <https://paperswithcode.com/sota/image-classification-on-imagenet> (ImageNet 기준 모델 성능 순위)
- [6] <https://amaarera.github.io/2020/08/30/gem-pool.html#gem-pooling> (GeM pooling 관련 설명)
- [7] <https://realibacklog.github.io/2020/03/29/normalization-standardization-regularization.html> (Normalize 관련 설명)
- [8] Tan, Mingxing, and Quoc V. Le. "EfficientNet: Rethinking model scaling for convolutional neural networks." arXiv preprint arXiv:1905.11946 (2019). (EfficientNet 논문)
- [9] Deng, Jifeng, et al. "Arcface: Additive angular margin loss for deep face recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019. (Arcface 관련 논문)
- [10] <http://www.aifaces.com/news/article/awh.html?doc=182756> (페이스 관련 기사)
- [11] https://docs.google.com/presentation/d/1ZUimafyXC85Lqbad6-a-dpQ7Vvcl17JhICBUVTA/edit#slide=id.g8b560a0a6_0_43 (라이프지 관련 자료)
- [12] Touvron, Hugo, et al. "Fixing the train-test resolution discrepancy: FixEfficientNet." arXiv preprint arXiv:2003.08237 (2020). (Test set의 resolution 변화에 따른 성능에 관한 논문)
- [13] <https://spinnventormit.edu/> (MIT에서 제공하는 오픈번터 공식사이트)

33

감사합니다

해당 깃허브에 코드공개예정, 데이터는 이메일문의

<https://github.com/rgw117/Sejong-Landmark-Challeng-ME-termproject-rgw117@naver.com>

4조

14011541 김 형준

16011504 임 진욱

18011230 배 대위

34

③공학주제(설계)

a.기초설계 교과목 리포트

-탑 구조물 제작

○ 목차

1. 조원소개 및 역할 안내
2. 프로젝트 설명
3. 브레인스토밍
4. 문제 정의
5. 작업 일지
6. 시행착오
7. 조원들의 소감

1. 조원소개 및 역할 안내

- 안홍규 (122552) - 조장 / 최종 수정
- 장효령 (13011450) - ppt 책임자 / 발표 준비
- 김도연 (16011553)- ppt 준비 / 발표 준비
- 김지광 (16011622) - ppt 준비 / 자료 준비
- 박대원 (16011574) - 기록자 / 자료 준비
- 임진욱 (16011504) - 보고서 책임자 / 자료준비
- 김태민 (16011529) - 보고서 준비 / 자료준비

2. 프로젝트 설명

○ 탑 구조물의 기능을 염두에 둔 설계

- 우주선 발사대 - 우주선 발사 시 준비과정이나 발사과정에서 우주선의 방향을 지지하는 역할.

○ 스파게티 면발 만을 이용하여 탑 구조형태 아이디어 구현

- 테이프와 목공용 풀을 이용해 접착.
- 스파게티 면은 보(beam)와 같은 형태라는 점 인식.

○ 구조적 안정성을 고려한 설계

- 우주선을 지지 및 발사할 때 발생하는 하중들을 버텨야하므로 이를 효과적으로 버티기 위해 트러스구조를 선택.

3. 브레인스토밍

○ 초기 탑의 설정에 대한 아이디어

- 안홍규 : 등대 (ex 팔미도 등대, 파로스 등대, 화암추 등대, 항공등대)
- 장효령 : 미국의 광범위한 농사지역에 물을 뿌릴 수 있게 해주는 스프링클러 역할의 탑
- 박대원 : 타워 크레인, 풍력발전기
- 김태민 : 관제탑
- 임진옥 : 전파탑
- 김지광 : 관제탑
- 김도연 : 전망대

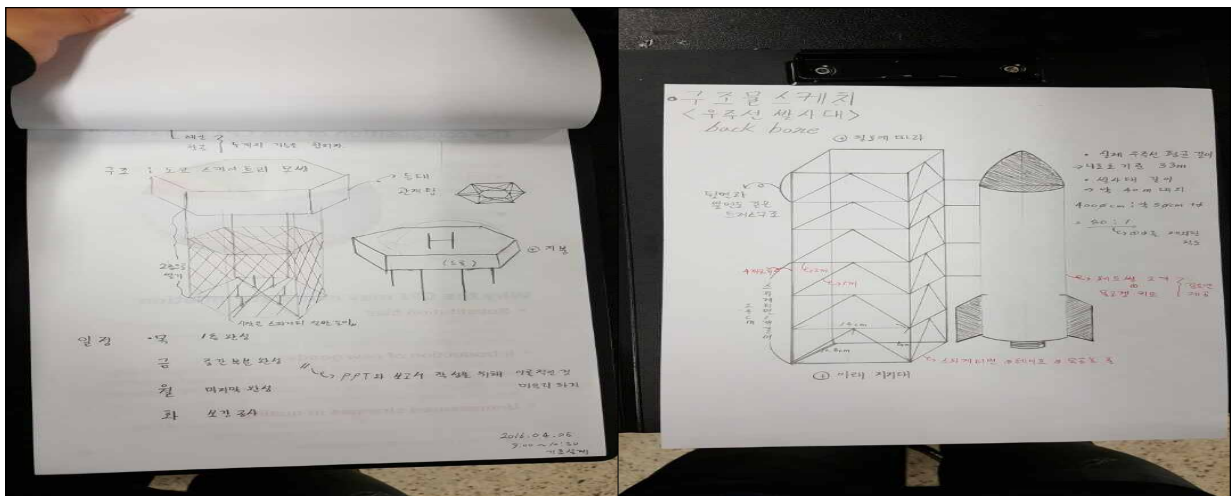


< 초기의 아이디어 >

○ 초기 탑의 기능 : 등대+드론

이 과정에서 탑의 목적에 대해 생각을 해 보았을 때 ‘드론이 왜 관측소가 필요한가?’ 라는 의문을 가졌다. 그 의문에 산간지방에서 이/착륙을 할 때 안전을 위한 목적을 생각했다. 이 의견에 교수님은 “그렇다면 드론에 탑과 서로 정보를 교환할 수 있는 무엇인가가 필요할 것이다.”라는 의문점을 제시하셨다. 그 결과 좀 더 나은 기능을 고려하게 되었고 “풍력발전을 더한다면 어떨까?(김지광)”라는 생각을 하게 되어 드론의 기능을 제거하게 되었다. 이 의견에 대한 추가의견으로 “둘 다 높으면 좋는데 하나를 탑의 중심 쪽으로 낮춘다면 비효율적이지 않은가?(장효령)”라는 의견과 “등대의 자급자족만 생각하면 전력은 충분히 보급 가능하다(박대원)”는 의견이 나왔다.

또 다른 아이디어를 생각하던 중 “작은 섬을 위한 다목적 탑을 만들자(안홍규)”는 의견이 나왔다. 그에 대한 멀티 탑의 기능으로 작은 마을에 전력을 보급할 수 있도록 태양광과 풍력발전기, 등대의 기능을 하는 탑을 생각을 하였다. 이 의견에 대한 추가 의견으로 태양광을 없애자는 의견이 나왔다. 이렇게 의견이 오가다가 아예 기존의 등대를 포기하고, 우주선 발사대를 만들자는 의견(안홍규)이 나왔다. 우주선 발사대는 1)기준에 탑하면 쉽게 떠오르지 않는 독창적 아이디어라 판단되었고 2)탑의 기능이 명확하며 3)탑의 안정성도 고려하기에 적합한 구조물이라 최종적인 아이디어로 채택되었다.



< 초기 구조 스케치 >

< 최종 구조 스케치 >

4. 문제 정의

1) 우주선 발사대의 역할 및 기능

지구 탈출 속도는 11.2km/s , 즉 4만 km/h 가 된다. 이 어마어마한 속력을 내기 위해서는 방법뿐만 아니라, 비용 적으로도 최대한 이득을 내야 한다. 우주선 탈출속도에 대한 이득을 얻기 위해 선속도가 가장 빠른 적도에서 쏘는 게 효율적이다. 그러나 이 부분은 이 프로젝트의 목적에 부합하지 않으므로 장소에 대한 자세한 내용은 생략을 한다. 지구의 자전과 지구면과 지구를 벗어나는 가상고도면의 최단거리로 이동, 공기와의 마찰 같은 외력도 최대한으로 줄이는 것을 고려해 우주선은 수직으로 쏘아야 한다. 우주선을 수직으로 세우기 위해서 발사대가 필요하며, 이것이 발사대의 핵심 기능이다. 발사대는 발사 전 우주선을 지탱해주는 역할뿐만 아니라 작용반작용의 원리로 공중으로 솟는 우주선의 발사 직후의 방향까지 지지해주는 역할을 수행을 해야 한다. 또한, 발사대는 우주선과 의사소통을 하는 중간 단계의 장소이다.

2) 높이와 폭을 구체적으로 어떻게 설정해야 하는지

탐의 기능을 정한 우리는 당장의 작업 시작에 앞서 try & error 즉, 시행착오를 최대한 줄이기 위해 이론적인 부분을 탄탄히 하는 것으로 결정하였다. 먼저 틀을 생각하였는데 실제 모델을 물 모델로 하여 축척을 정해 틀을 잡기로 하였다. 그 실제 모델은 '아폴로 우주선 새턴 V2로켓'이며 높이는 약 40m 이다. 우리가 사용할 스파게티 면발의 길이는 약 24cm 이므로, 스파게티 면발 2개로 기둥을 세운다고 가정하면 높이는 48cm 이다. 이는 실제 높이(40m)와 $1:83.333$. 즉, 약 $1:80$ 의 축척을 이룬다. 하지만, 폭은 인터넷에서 정보를 구하기가 어려워 실제 폭의 길이를 구하지 못하였다. 따라서 폭의 길이를 설정하기 위해 참고할만한 모델이 필요하였다. 따라서 모형 우주선인 물 로켓 참고모델로 하여 폭을 설정하였고 길이는 14cm 가 되어 $14 \times 14 \times 48(\text{cm})$ 인 정육면체 모양의 탐의 틀이 완성되었다. 왜 정육면체인지는 3) 구조적 설계란 에서 설명을 하겠다.

※ 물 로켓을 사용해야 하는 이유

1. 실제 발사대의 역할인 지지대의 모습을 구현하기 위해.

-우주선 발사대의 핵심기능인 우주선 지지대의 역할을 실현하기 위해 물 로켓이 필요했다.

2. 우주선 발사대의 사이즈 설정.

-실제 발사대의 높이와 폭을 알았다면 그에 맞게 축척을 하여 제작하면 된다. 높이는 알 수 있었지만, 폭은 알 수 없었기 때문에 물 로켓을 참고삼아 폭을 설정하고 그 폭에 맞게 적절한 폭을 결정해야 했다.

3) 구조적 설계

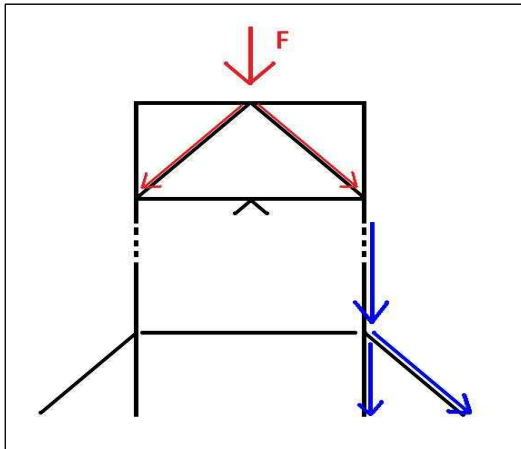
가. 다양한 역할을 수행할 수 있는 구조.

우주선 발사대는 앞서 설명했듯이 우주선의 방향을 지지해줄 뿐만 아니라 발사 전 우주선과의 통신 등 다양한 공학적 장치들이 들어가 있는 중간 단계의 장소이다. 우리는 장소가 협소한 것보다 넓은 게 적합하다고 판단하여, 높이가 높아져도 공간이 넓은 구조를 생각하게 되었다. 따라서, 높이가 높아질수록 좁아지는 모양이 아닌 정육면체의 구조로 설계하였다.

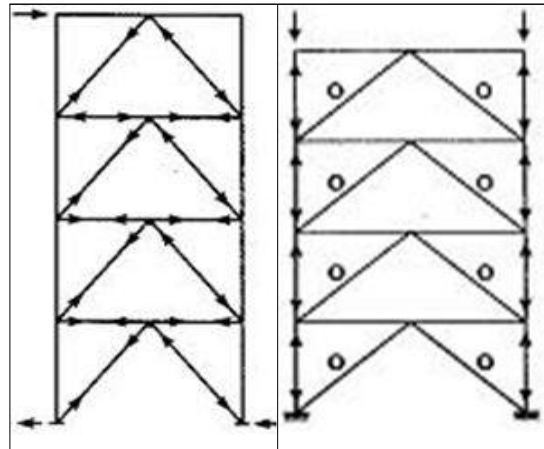
나. 안전한 구조

틀은 갖춰진 가운데 강한 하중을 어떻게 안정적으로 버틸지를 생각했었다. 발사대는 우주선을 지탱할 때 우주선이 기울 때의 무게를 버티는 것뿐만 아니라 추진 체에서 나오는 어마어마한 가스가 땅을 미는 힘을 버텨야 한다. 발사 직후 발사대 바로 옆에서 발사되는 우주선은 당연히 발사대도 밀면서 위로 나아갈 것이므로 강한 하중이 발생되고 이는 발사대를 위에서 아래로 내리는 방향이다. 이를 효과적으로 버티기 위해 트러스 구조를 선택하였다.

<그림 1>과 같이 트러스 구조를 선택하였고 그 이유는 벡터 합법칙에 따라 나뉘진 방향에 따라 힘이 반반씩 나뉘므로 제일 많이 힘을 버티는 부담을 줄여준다고 생각했기 때문이다. 그리고 양끝 기둥은 <그림 2>를 보면 알 수 있듯이 기둥자체가 힘을 버티므로 아래 지지대를 트러스 구조의 개념으로 강화해줌으로써 단단하게 하였다.

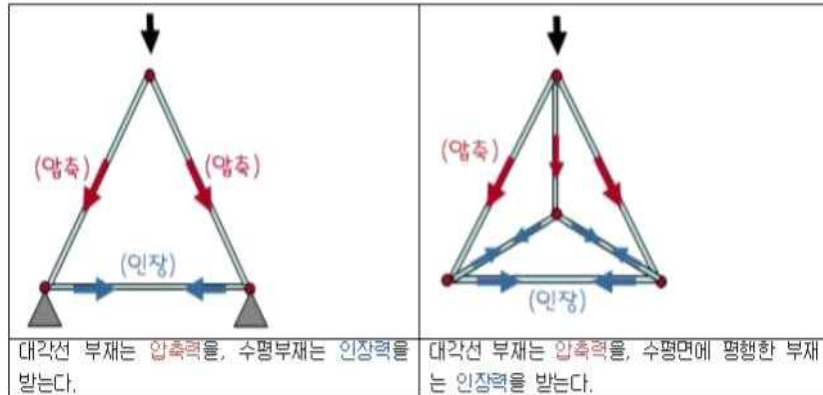


< 그림 1 >



< 그림 2 >

※ 트러스(Truss) 구조 : 몇 개의 직선 부재를 한 평면 내에서 연속된 삼각형의 뼈대구조로 조립한 것



출처 : <http://blog.naver.com/smartcst/30180400598>

6. 작업 시 시행착오 및 에러 사항

- 스파게티 면을 접착할 때 무엇으로 접착을 해야 할지가 문제였다. 목공용 풀로 스파게티 면을 붙였을 때는 접착하는데 걸리는 시간이 길고 접착하는 시간동안 스파게티 면이 고정되지 않아 어려움이 있었다. 그래서 테이프로 먼저 스파게티 면을 고정시킨 후 목공용 풀로 붙였는데 시간은 걸리지만 손으로 잡아 고정시키는 수고는 덜었다. 목공용 풀을 말리는 데에도 일반적으로 말리기에는 오랜 시간을 기다려야 하므로 헤어드라이기를 사용하였다.
- 지급되어진 재료만을 가지고 프로젝트를 진행해야 되기 때문에 물 로켓을 사용해야하는지 말아야하는지 고민이 많았다. 제작을 하는 과정에서 실제처럼 비율을 맞추기 위해 물 로켓 사용하였지만 프로젝트에서의 제한적인 상황이었고 이 상황을 어떻게 해야 할지 팀원들과 의논하였다. 의논한 결과 프로젝트 제작 과정에서 필요했고 기능에 대한 설명에 필요했기 때문에 사용하기로 결정했다.
- 우주선과 발사대가 연결되는 것을 스파게티 면으로 제작을 하니 표현하기 어려운부분이 있었지만 최대한 표현해보려고 노력했다. 우주선 발사대에서 튀어나온 구조물과 우주선을 연결하기 위해 물 로켓 구멍을 뚫기로 했다. 물 로켓이 생각보다 단단한 플라스틱 재질이라 송곳으로 구멍을 뚫는데 구멍의 모양과 크기를 정확히 맞추기가 힘들었다. 그래서 사용하지 않는 페트병에 여러 가지 방법을 시도해보고 찾은 방법이 송곳과 볼펜, 칼 등 온갖 동원할 수 있는 것을 다 동원하여 크기를 맞추었다.

○ 목차

1. 조원소개 및 역할 안내
2. 프로젝트 설명
3. 브레인스토밍
4. 문제 정의
5. 작업 일지
6. 시행착오

2. 프로젝트 설명

○구조물의 무게는 계란의 무게(약 55g)를 넘어서는 안된다.

○계란을 테이프나 끈으로 감싸서 계란의 표면 강도를 높이는 행위는 안 된다.

○사용재료로는 종이, 빨대, 소독저, 풀, 고무줄, 접착테이프 등 정해진 재료만을 사용한다.

○구조물이 계란을 넣을 수 있는 개폐형 구조여야 한다.

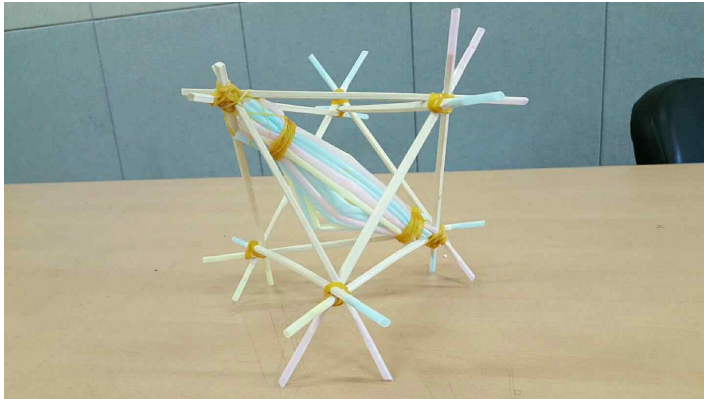
3. 브레인스토밍

○ 구조물에 대한 아이디어

- 안홍규 : 빨대의 접히는 부분을 사용하여 자전거 완충 장치와 같은 역할을 하는 아이디어, 낙하산 부착
- 장효령 : 빨대를 작게 자르거나 고무줄들을 모아 스펀지와 같이 충격을 흡수할 수 있는 아이디어, 정팔면체 형태의 틀 고안
- 박대원 : 정사면체 형태의 틀, 날개를 만들어 프로펠러를 적용하는 아이디어 고안
- 김태민 : 비닐봉지를 사용하여 계란을 넣을 수 있는 구조 고안
- 임진욱 : 빨대를 사용하여 상자를 만들고 상자에 계란을 넣는 구조 고안
- 김지광 : 비닐봉지를 사용하여 계란을 넣을 수 있는 구조 고안

○초기 계란 구조물

먼저 계란이 낙하되어 충돌될 때 안전하게 보존될 수 있는 구조물을 생각해보았고 정사면체 형태가 구조적으로 안정되어 있으니 정사면체의 틀을 사용하자는 아이디어가 나왔다. 이후 정사면체 틀에서 충돌 시 계란에 작용하는 충격량을 줄이도록 하는 여러 아이디어가 나왔다. 빨대의 특성을 적용하여 완충장치와 같은 구조를 만들자는 아이디어, 비닐봉지와 빨대로 만든 상자에 계란을 넣어 정사면체의 꼭짓점에 고무줄로 연결하자는 아이디어 등이 나왔고 초기 구조물로 채택된 것은 정사면체의 빨대 틀에 비닐봉지를 고무줄로 연결한 구조물이다.

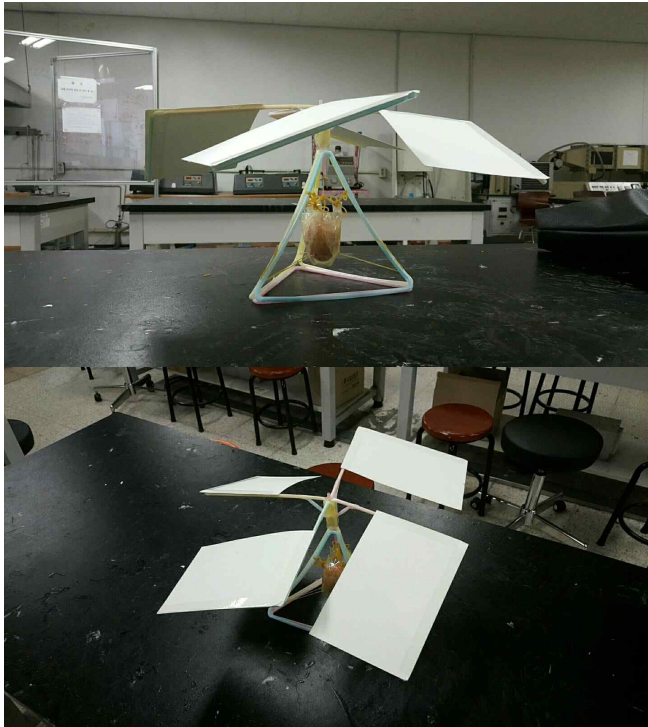


○보완한 계란 구조물

초기 구조물은 낙하속도도 빠르고 충돌 시 충격을 흡수해줄 장치가 부족했다. 그리고 교수님이 지적하신 '사면체가 지면에 충돌할 때에 충돌되는 부분이 꼭짓점이 될 수도 있고 면이 될 수도 있다.' 이 부분의 보완점을 찾으며 생각해 낸 아이디어가 정팔면체이다. 정팔면체는 대칭성이 있어 어느 부분으로 떨어져도 안전하다는 생각이었다. 그래서 정팔면체 나무틀을 만들었고 옛날에 계란을 짚으로 묶어 보관하던 것과 같이 빨대로 계란을 감쌌다. 또, 지면에 착지할 때에 충격흡수에 도움을 주도록 나무틀 끝부분에 빨대로 다리를 만들었다.

○최종 계란 구조물

두 번째 구조물에서 속도를 줄일 수 있는 방법이 없을까 생각하다가 낙하산과 프로펠러를 사용하자는 아이디어가 나왔다. 하지만 낙하산을 만들기엔 필요한 재료가 없어서 프로펠러를 제작하기로 했다. 최종 구조물의 틀은 초기 구조물과 같은 정사면체 빨대 틀로 정했고 스케치북과 빨대로 프로펠러를 만들고 계란을 보관하는 통을 테이프로 제작했다.



4.문제 정의

◆계란이 깨지지 않는 구조

1)충격량 감소

계란이 깨지지 않기 위해서는 계란이 지면과 충돌할 때 받는 충격이 작아야 한다.

$$\text{충격량}(I) = \text{충격력}(F) \times \text{충격시간}(t)$$

충격량은 위의 식과 같은데 여기서 충격력을 줄이기 위해서는 충격시간을 늘여야 하는 것을 알 수 있다.

또, 충격량은 운동량의 변화량으로 나타낼 수 있다.

$$\text{충격량}(I) = \text{운동량의 변화량}(\Delta P)$$

여기서 운동량(P)=질량(m)×속도(v) 이므로

$$\text{충격량}(I) = \text{운동량의 변화량}(\Delta P) = \text{질량}(m) \times \text{속도의 변화량}(\Delta v) \text{ 으로 나타낼 수 있다.}$$

위 식에서는 충격량을 줄이기 위해서는 운동량의 변화량을 줄여야 하는 것을 알 수 있는데 이는 속도의 변화량을 줄이는 것과 같다.

때문에 계란이 깨지지 않는 구조에서 두 가지를 고려했다.

첫 번째, 충격 시간을 늘리기 위해 고무줄의 탄성력을 사용 하였다.

두 번째, 우리가 계란을 낙하시킬 때 처음속도가 0 이므로 지면에 닿기 직전의 속도를 최소로 하면 속도의 변화량이 작아진다. 속도를 줄이기 위해 프로펠러를 만들어 최대한 공기저항을 받도록 했다.

2)구조물의 형태

정사면체인 이유;;이거 지평이가 자료 가지고 있을텐데 물어봐서 써주라 밑에 는 왜 빨대를 썼는지 도움이 될까해서 써봤어

물질	고무	나무	빨대(폴리에틸렌)
영률(10^9Pa)	0.002-0.008	10-15	5.9

계란구조물이 충격을 최대한 흡수를 하여 계란을 보호해야하는데 충격을 최대한 흡수하려면 변형력에 의해 쉽게 변형이 일어나야한다. 그럴 경우 동일한 변형력에 대해서 영률(물체의 늘어나는 정도와 변형되는 정도를 나타내는 탄성률)이 작을수록 변형이 쉽게 일어난다. 위에 표를 보면 고무가 영률이 가장 작고 빨대, 나무 순서로 커지는 것을 알 수 있다.

따라서 변형이 너무 일어나는 고무나 변형이 잘 일어나지 않아 충격흡수가 안 되는 나무 보다 빨대를 채택했다.

출처:<http://blog.naver.com/s00s1016/220600111102>

6.시행착오

시행착오1

1. 의도한 방향으로 떨어지지 않음. → 계란이 땅에 직접 닿음
2. 고무줄이 충격을 버티지 못하고 끊어짐.

시행착오2

1. 낙하시간이 짧아 충격량이 큼.

○ 목차

1. 조원소개 및 역할
2. 프로젝트 이해
3. 문제 정의
4. 브레인 스토밍
5. 결과 분석
6. 작업 일지
7. 느낀점

1. 조원소개 및 역할

- 안홍규 (122552) – PPT 준비 / 발표 준비 / 최종수정
- 장효령 (13011450) – 보고서 준비 / PPT 준비 / 설계도면
- 김도연 (16011553) – PPT 준비 / 자료 준비
- 김지광 (16011622) – 기록자 / 자료 준비
- 박대원 (16011574) – 종이 다리 제작 / 조장
- 임진욱 (16011504) – 종이 다리 제작 / 자료 준비
- 김태민 (16011529) – 보고서 준비 / 발표 준비

2. 프로젝트 이해

○ 제한조건

- 25mm X 25mm X 300mm 규격제한
- 주어진 하드보드지로 제작
- 뚫려있는 구조물 내부
- 접착제(딱풀, 목공용 풀) 사용

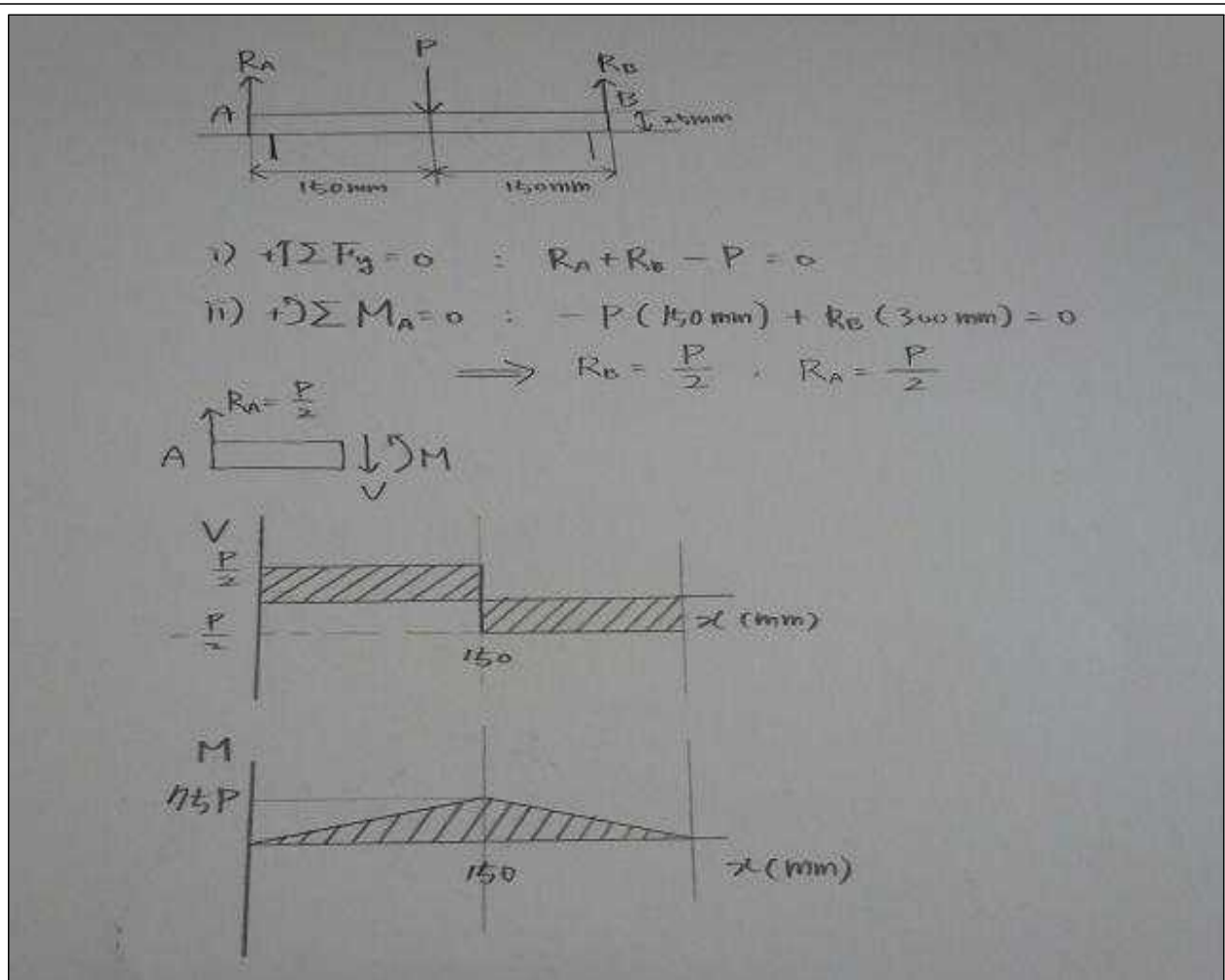
○ 평가항목

- 다리 중간의 14kg 중량을 가한 후 견디는지
- 가벼운 무게
-

3. 문제 정의

이번 실험에서 고려해야하는 요소는 전단과 굽힘모멘트, 그 힘으로 발생 하는 응력(Stress)들이다. 그리고 단면에 대해 힘 정도를 예측해줄 수 있는 단서인 2차모멘트 값이 핵심이다.

○ 전단과 굽힘모멘트



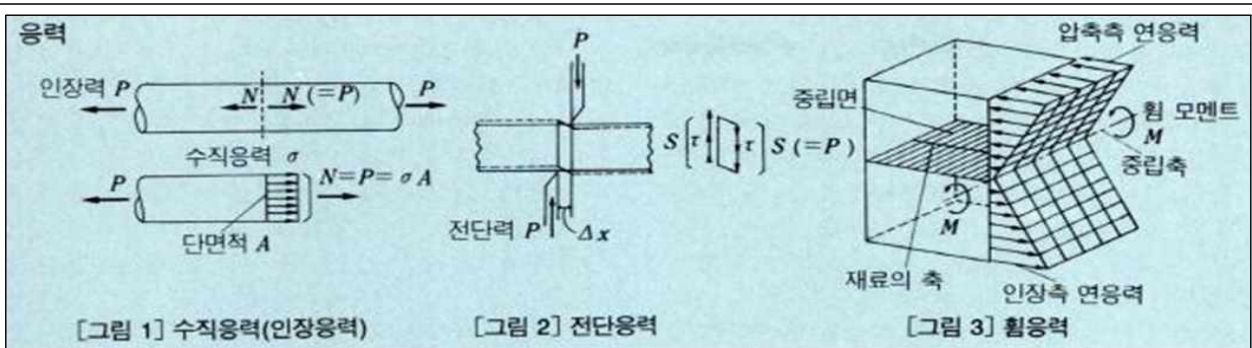
< 부재에 가해지는 하중 및 모멘트 >

위 그림에서 2차원적으로 생각하면 자유물체도는 그림의 맨 위와 같고 각 힘들의 크기는 아래 표와 같다. 즉, 하중이 가해지는 부재의 중앙에서 $P/2$ 크기인 34.3N의 힘과 가장 큰 굽힘 모멘트를 받는 것을 알 수 있고 그 크기는 5.125Nm이다.

구 분	크 기 (중력가속도=9.8m/s ²)
P	$mg = (7\text{kg})(9.8\text{m/s}^2) = \mathbf{68.6\text{N}}$
M_{\max}	$r(\text{거리}) \times P = (75\text{mm})(68.6\text{N}) = \mathbf{5.125\text{Nm}}$

○ 응력(Stress)의 종류

부재는 응력(stress)이 생기는데 크게 수직응력과 전단응력, 그리고 휨응력으로 나눌 수 있고 아래 그림에 나와 있다. 수직응력은 인장력에 의해 발생하는 응력이고 전단응력은 전단력에 의해 발생하는 응력이다. 그리고 휨응력을 모멘트에 의해 발생하는 응력이다. 이번 실험에서는 인장력이 없으므로 전단응력과 휨응력만 존재한다.

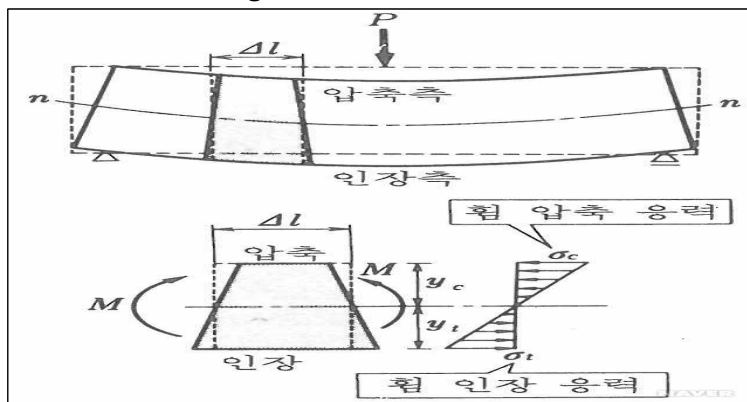


< 발생하는 응력의 종류 >

○ 전단응력 (Shearing Stress)

전단응력은 부재를 끊어버리는 응력이다. 하지만 이번 프로젝트에서 추에 매달은 실이 하드보드지로 만든 종이 다리를 끊는 것보다, 휨응력에 의해서 구부러지는 것이 더 큰 관건이었기 때문에 전단응력을 크게 신경 쓰지 않았다.

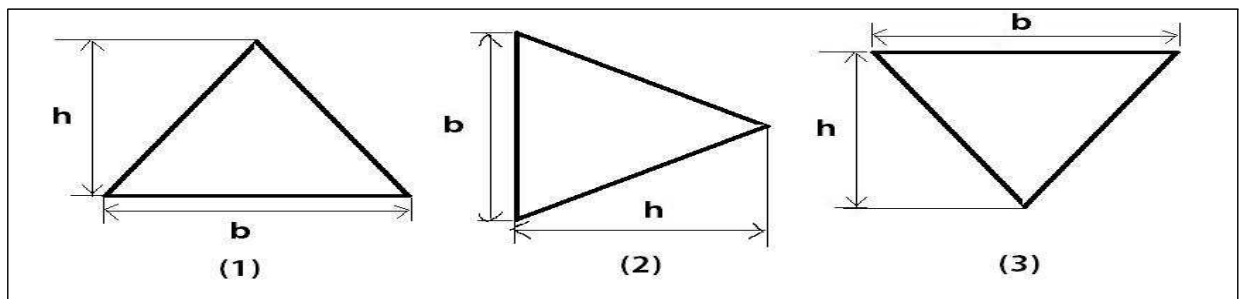
○ 휨응력 (Bending Stress)



휨응력은 굽힘 모멘트(Bending moment)에 의해 발생하고 공식은 $\sigma = \frac{Mc}{I} = \frac{M}{S}$ (M =모멘트, c =중심축으로부터의 거리, I =단면에 대한 2차모멘트)이다. 굽힘 모멘트란 보(beam)에 어떤 힘이 가해질 때 작용하는 보를 굽히려는 힘이다. 일반적으로 위 그림에서 보듯이 중심축에서 멀리 떨어질수록 휨응력의 크기가 커진다.

○ I 값과 휨응력과의 관계

구조물의 모양을 정하기 위해 어떤 모델이 굽힘에 대해 유리한지 알아보려 I값을 고려했다.



$\sigma = \frac{Mc}{I}$ 에서 단면 2차 모멘트 I 는 휨 또는 처짐에 대한 저항을 예측하는 데 사용되는 단면의 성질을 뜻한다.

쉽게 생각하기 위해 이 I값만을 고려해서 모델을 정했다. 위 그림에서 일단 2번은 좌우 대칭이 아니기에 한쪽으로 힘이 더 집중이 되어 다른 1, 3번 모델보다 불리할 것임을 쉽게 알 수 있다. I의 값은 평행축 정리에 의해

$I = \frac{1}{12}bh^3 + Ay^2$ 이며 y 값은 하중이 가해지는 지점을 기준으로 도심까지의 거리이다. 여기서 $b=h=25\text{mm}$ 이고 A (면적)은 모두 같다. 따라서 y 값만 고려해주면 y 값이 클수록 I 값이 크다는 것을 알 수 있다. 일반적으로 이 등변삼각형의 경우 도심은 밑변에서부터 $(1/3)h$ 인 지점이므로 y 값을 구해주면 아래 표와 같다.

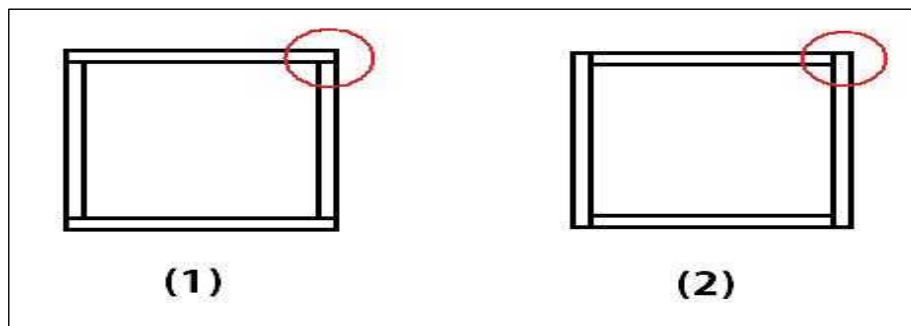
구 분	(1)	(2)	(3)
y (mm)	$\frac{2}{3}h$	$\frac{1}{2}h$	$\frac{1}{3}h$

따라서 1번 모델이 가장 I 값이 크다는 것을 알 수 있으며 힘에 대한 저항력이 가장 크다는 것을 예측할 수 있다.

4. 브레인스토밍

1) 사각형의 구조

부재의 외벽을 이루고 있는 사각형 구조에서는 아래 그림에서 (2)에서는 하중을 폴의 접착력으로만 버티는 꼴이므로 (1)처럼 기둥이 받혀주고 있는 형태를 선택하였다.

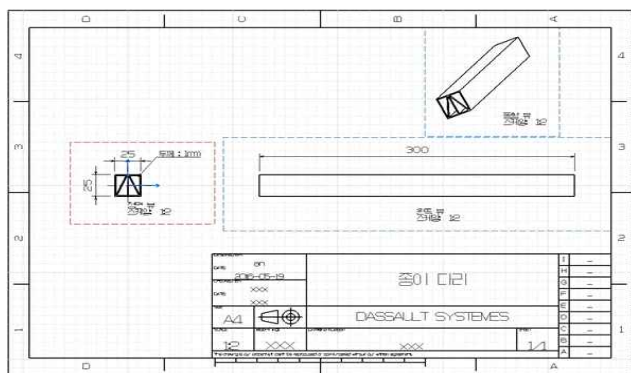


2) 굽힘에 잘 버티는 구조

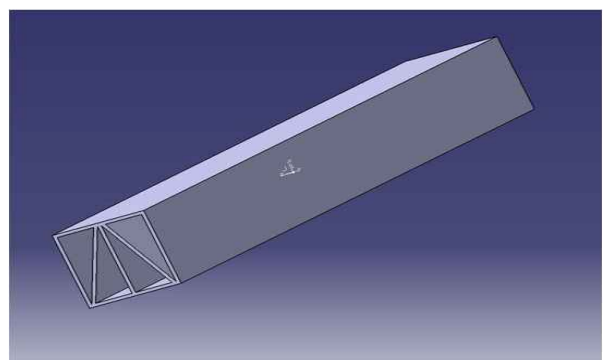
굽힘에 대해 최대한 버티기 위해 휨응력이 최소인 밑변이 아래로 향하는 삼각형 구조를 선택하였다. 즉, 면적에 대한 2차모멘트가 최대인 구조를 선택하였고, 구조적뿐만 아니라 직접적으로도 굽힘에 대해 저항을 갖도록 구조물 가운데에 기둥도 추가하였다. 이번 실험의 핵심은 굽힘을 견디는 것이다. 단면에서의 트러스 구조는 하중을 직접적으로 버티는 것이 핵심이며, 이는 이번 프로젝트의 핵심인 굽힘과는 다소 거리가 있다. 또한 트러스 구조는 앞선 프로젝트들에서 여러 차례 이미 언급 되었다. 따라서 프로젝트에서 트러스 구조에 대한 자세한 설명은 생략하였다.

3) 최종 모습

앞서 설계한 구조대로 CATIA 로 설계를 한 후에 구조물을 만들었다.



< 설계 도면 >



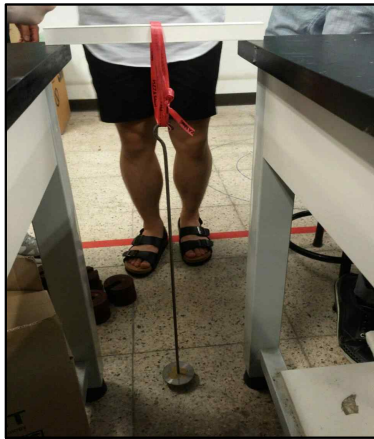
< 최종 모습 >

5. 결과분석

○ 5/19 실험 분석

1) 실험

5/17 화요일 오후에 만들었던 샘플 종이다리를 약 36시간을 건조시킨 후 5/19 목요일 수업시간에 양쪽 테이블에 25mm 씩 걸치고 종이 다리 가운데에 실로 추를 매달아서 2kg 씩 5초 간격으로 14kg 을 매달았다.



< 실험 모습 >



< 성공한 모습 >



< 무게 >

2) 결과 및 분석

종이 다리에 14kg 의 추를 매달은 결과, 매우 잘 버티었고 변형이나 훼손이 거의 일어나지 않았다. 이렇게 잘 버틴 이유를 추론하면 이론적으로도 하중을 잘 분산시키는 이유도 있지만 종이다리를 제작하는 과정에서 조원들이 매우 세심하고 정교히 만들었기 때문이다. 우리가 만든 샘플 모델은 설계도와 비교하였을 때, 오차도 거의 없고 균형이 매우 잘 맞는 등 이론에서 크게 벗어나지 않는 이상적인 모델이었다. 실험에 성공한 후, 남은 과제는 어떻게 하면 무게를 좀 더 효과적으로 줄일 것인가였다. 조원들과 토의 후, 이론상 이 구조가 제일 안전하게 버티는 구조라고 생각하였고 안전한 구조 그대로 가기로 하였다. 따라서 기존의 모델을 더욱 정교하게 하는 것으로 최종 결정하였다.

○ 시연 분석

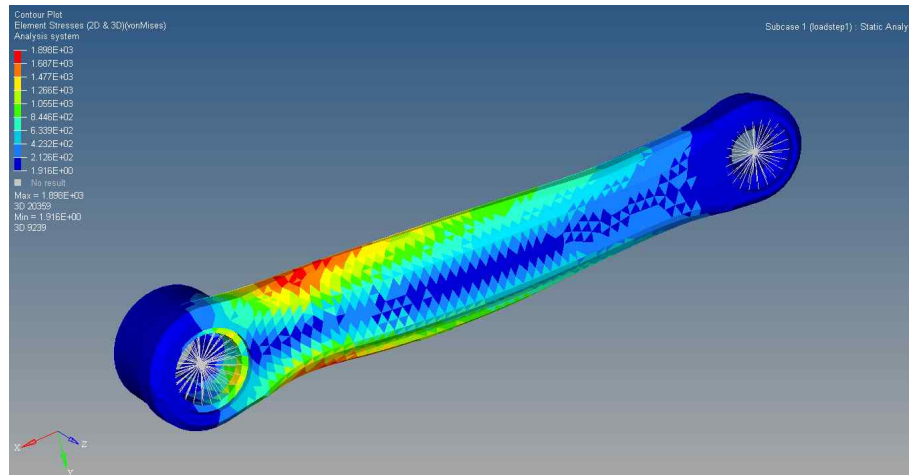
1) 시연 및 결과

시연 결과, 실험 때와 마찬가지로 별다른 굽힘없이 성공하였다. 우리의 구조물은 시연 후 거의 변형이 일어나지 않았다. 하지만 이번 프로젝트의 목표는 어느 정도의 굽힘(변형)이 발생하더라도, 추가 바닥에 닿지 않는 선에서 버티는 것이 목표였다. 그래서 우리 조의 구조물보다 더 많은 변형이 발생하였지만, 더 무게가 작은 조들에 밀려 4등을 하게 되었다. 이번 프로젝트에서 상위권을 못한 가장 큰 이유는 무게 제한이라는 조건을 가볍게 생각한 데에서 비롯되었다. 시연 때 굽힘이 발생하여 실패할 것을 지나치게 염려하였고, 샘플 모델은 교수님께서 칭찬하실 정도로 정교하게 만들어졌기 때문에 이 모델을 끝까지 고수하게 되었다. 좀 더 시간적 여유를 가지고 다른 모델을 만들어 보며 측정을 해봤으면 더 적은 무게의 모델로 성공할 수도 있었을 것이다. 그러나 우리는 현실에 안주하였고, 이것이 상위권에 들어가지 못한 이유라고 생각된다.

b.응용기계설계 교과목 리포트 및 발표자료

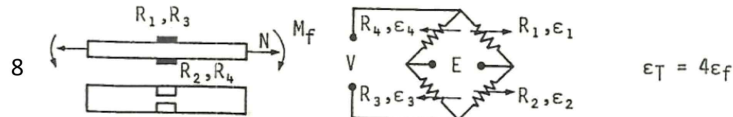
-스트레인게이지를 활용한 파워미터 제작

Weekly Report			
Date	2020.05.04	Team	14조
Report writer	18011273 김용민		
Members (Participants only)	16011504 임진욱 18013256 박현우 18011276 강전욱 18011230 배대위		
Topic	1. 크랭크암에 작용하는 힘과 응력 분석 (이론과 Hyperworks) 2. Strain gauge 부착 위치 선정 및 방법		
Discussions	<p>본 설계의 목적은 strain gauge를 통해 크랭크암의 변형률을 알아내고 변형률을 통해서 힘을 구하는 것이다. 이번 만남에서는 strain gauge의 부착 위치를 선정하고 어떻게 부착할지를 정했다.</p> <p>Strain gauge를 부착하는 위치는 크랭크암이 하중을 받을 때 응력이 가장 크게 나타나는 부분이다. 왜냐하면 strain gauge는 변형률을 측정하는 장치인데 응력과 변형률은 비례 관계()이므로 만약 응력이 작은 부분에 strain gauge를 부착하게 된다면 변형률이 너무 작게 나오고, 유의미한 힘의 크기를 구할 수 없기 때문에 응력이 큰 부분에 strain gauge를 부착해야 한다.</p> <p>부착하는 위치를 이론적으로 구한다면 (크랭크암을 단면이 사각형인 beam이라고 가정)</p> <p>if axial force \Rightarrow</p> <p>$\sigma_x = \frac{P}{A} \quad \epsilon_x = \frac{P}{E \cdot A}$</p> <p>$\therefore$ 위치에 관계없이 σ, ϵ 값 일정</p> <p>이와 같은 방법으로 구할 수 있다. 따라서 strain gauge의 부착 위치는 최대한 고정 축과 가깝고, neutral surface로부터 멀리 떨어진 부분에 부착해야 한다. 이것이 실제로 맞을지 Hyperworks를 통해 확인해 본다</p>		



빨간색으로 된 부분이 응력이 가장 크게 걸린다는 것을 알 수 있다. 이론과 비교했을 때 매우 비슷한 위치라는 것을 알 수 있다.

따라서 이와 같은 이유로 strain gauge를 위 사진에서 붉은 부분에 설치를 해주어야 한다. Strain gauge를 설치하는 방법은



이와 같은 방법으로 설치를 해야 응력이 가장 강하게 작용하는 부분의 bending에 의한 변형률만을 측정할 수 있다. 하지만 실험에 쓰일 크랭크암의 형상은 다음과 같은 방법으로 strain gauge를 설치할 수 없기 때문에

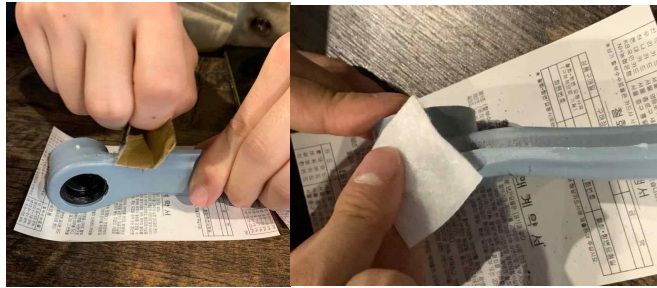


이와 같은 일자 형태로 설치하고 식을 변형시키고 두 부분의 변형률의 평균값을 이용한다. 두 부분에 각각 Strain gauge를 최대한 가깝게 붙이면 두 센서에서 측정되는 변형률의 차이가 크게 나지 않을 것이다.

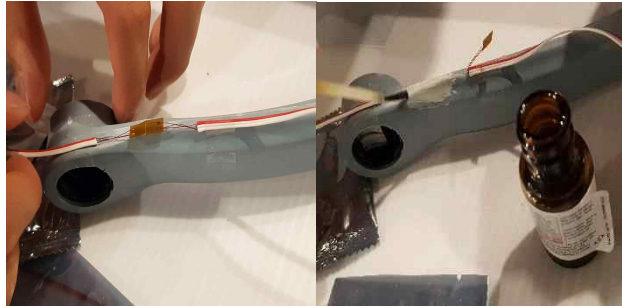
Grade (A/B/C/F)		Delayed	
--------------------	--	---------	--

Weekly Report

Date	2020.05.16	Team	14조
Report writer	18011276 강전욱		
Members (Participants only)	16011504 임진욱 18013256 박현우 18011273 김용민 18011230 배대위		
Topic	1. 크랭크 암에 스트레인게이지 부착 2. 휘트스톤 브릿지 구성		
Discussions	크랭크 암에 스트레인게이지 부착 1. 크랭크 암의 어떤 부위에 스트레인게이지를 부착할지 이전 모임에서 토의를 진행했었기 때문에 바로 해당 위치에 사포질을 했다. 그 후 알코올솜으로 세척을 했다. (철판과 스트레인게이지의 접촉면을 늘리는 효과 외에도 혹시 모를 이물질 제거도 기대할 수 있다.)		



2. 토의로 선택했던 부위에 테이프로 임시고정을 한 뒤 록타이트384를 퍼 발라주고 록타이트7387을 위에 덧칠했다.



3. 테이프로 위치가 틀어지지 않도록 하며 밀착시킨 뒤 굳어 고정되기를 기다렸다. 하지만 잘 고정이 되지 않아 다른 방법들을 많이 써봤지만 마찬가지로 결국 실패했다. 그래서 무엇이 문제이고 어떻게 해결할지 다시 논의를 했다.

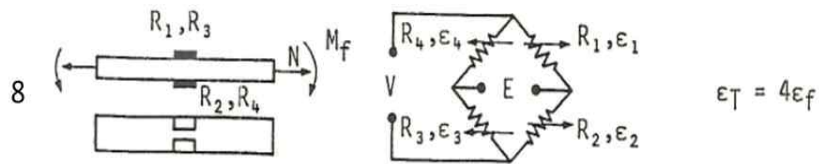


4. 토의 끝에 스트레인게이지의 끝 부분이 크랭크 암과 밀착되지 못하고 뜨는이유를 스트레인게이지의 부착면적이 너무 좁아서 온전히 붙지 못하는 것으로 결론을 내렸고 모든 조원의 만장일치로 이를 해결하기 위해 약간 비스듬하지만 어쩔 수 없이 면적이 더 넓은 비탈면을 이용하기로 했다. 그리고 스트레인게이지 부착을 성공했다.



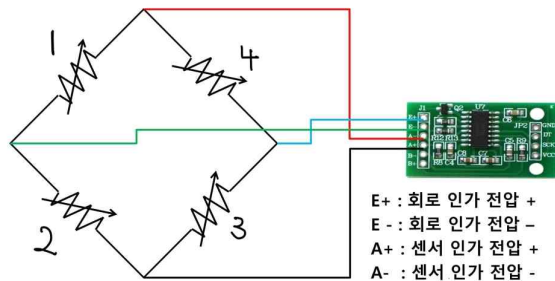
휘트스톤 브릿지 구성

1. 팀원들과 지난번 회의를 할 때 아래의 그림과 같은 구성이 가장 적합할 것이라고 예상하고 해당 브릿지를 설계하기로 했었다. 그런데 크랭크 암이 스트레인게이지를 양 옆으로 부착하기에는 면적이 좁은 관계로 앞뒤로 부착한 뒤 평균값을 도출하는 방식으로 하기로 했었다.

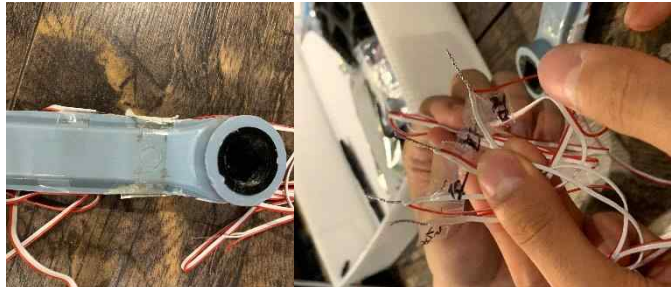


$$\begin{aligned}\epsilon_T &= \epsilon_1 - \epsilon_2 + \epsilon_3 - \epsilon_4 \\ &= (\epsilon_n + \epsilon_f + \epsilon_a) - (\epsilon_n - \epsilon_f + \epsilon_a) \\ &\quad + (\epsilon_n + \epsilon_f + \epsilon_a) - (\epsilon_n - \epsilon_f + \epsilon_a) \\ &= 4\epsilon_f\end{aligned}$$

2. 그래서 스트레인게이지를 앞에서 설명한 방식으로 부착했고 휘트스톤 브릿지 연결은 스트레인게이지의 +선과 -선을 묶어서 만들 수 있었다.



3. R1, R2, R3, R4에 해당하는 선을 회로의 모양과 일치하게 만들고 선이 너무 복잡해 보여서 구별하기 쉽도록 테이프로 글씨를 써 부착했다.



Grade (A/B/C/F)		Delayed	
--------------------	--	---------	--

Weekly Report

Date	2020-05-20 2020-05-22	Team	14조
Report writer	16011504 임진욱		
Members (Participants only)	18011276 강전욱 18013256 박현우 18011273 김용민 18011230 배대위		
Topic	1)strain gauge실험 2)arduino 회로 연결 3)Arduino coding		

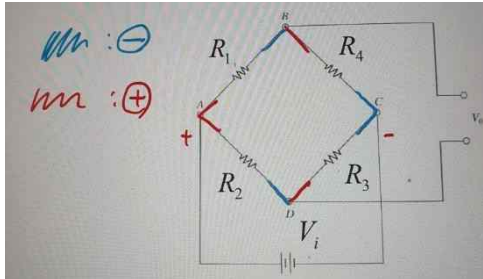
Discussions

2020-05-20
•strain gauge 실험

- strain gauge 부착에 있어서 문제가 있어 새로운 strain gauge로 크랭크암에 다시 부착하였다.
- 가능한 같은 면에 붙어있는 strain gauge가 가깝고 일직선 상에 놓이도록 부착하였다. 그래야 load cell의 오차가 커지지 않기 때문이다.



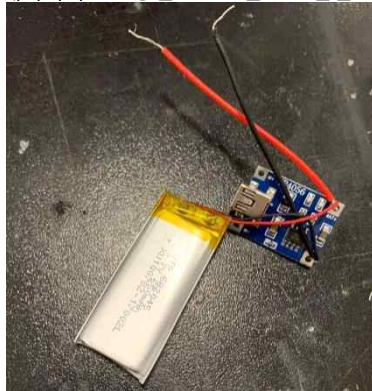
- 기존 휘트스톤브리지의 연결방식은 strain gauge가 직렬로 연결되어 있었기에 다시 토의한 후 병렬로 연결하였다.



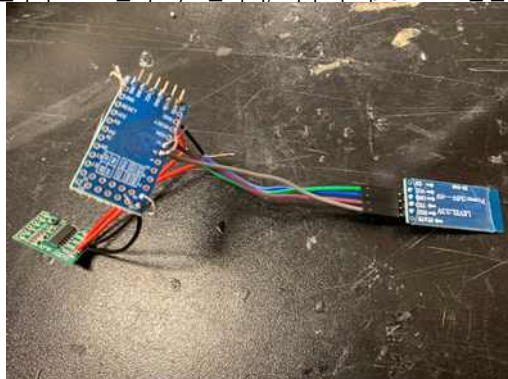
이후 진행한 실험에서 컴퓨터에서 출력되는 strain gauge값을 확인할 수 있었다.

•arduino 회로 연결

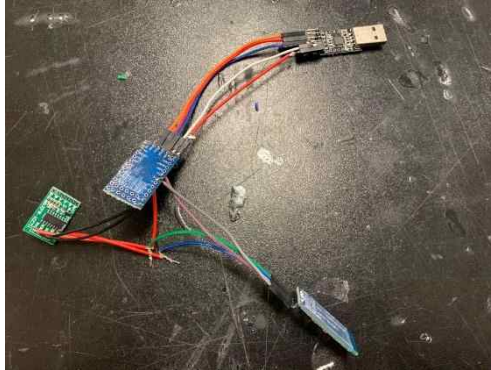
- 1) 배터리와 충전 모듈 연결 (승압 컨버터와 스위치는 연결하지 않음)



- 2) 블루투스 모듈과 A/D컨버터, 아두이노pro mini 연결



3) 아두이노pro mini와 USB UART 연결



4) A/D컨버터와 휘트스톤브리지 연결



-회로 연결과정에서 점퍼케이블을 잘라서 사용함, 이 때문에 핀에 꽃을
점퍼케이블이 부족해 일반케이블로 납땜하여 마무리했다. 다소 지저분하고 불안해
보이기도 함

-노트북에 USB연결하여 전원이 들어오는지 확인한 뒤 작업 마침

2020-05-22

•arduino coding

- 1) 아두이노 코드에 대해 학습한 뒤
ide에서 코딩함
-오른쪽 사진이 코딩한
스케치이다.

sketch_may21a

```
1 #include <HX711.h>
2
3 #define DOUT 8
4 #define CLK 9
5 #define calibration_factor -7050
6
7 HX711 scale(DOUT, CLK);
8
9 void setup() {
10   Serial.begin(9600);
11   scale.set_scale(calibration_factor);
12   scale.tare();
13 }
14
15 void loop() {
16   Serial.print(scale.get_units(), 1);
17   Serial.println();
18
19   delay(100);
20 }
```

- 2) 노트북에 USB연결하여 아두이노에 업로드

-노트북에 연결했을 때 아두이노에 전원이 들어오는 것을 확인함

-업로드하기위해 업로드를 누르고 아두이노의 리셋 버튼을 누르고 난 뒤 아래와 같은 오류가 발생함


```

avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 1 of 10: not in sync: resp=0x00
avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 2 of 10: not in sync: resp=0x00
avrdude: stk500_rcv(): programmer is not responding
avrdude: stk500_getsync() attempt 3 of 10: not in sync: resp=0x00
avrdude: stk500_rcv(): programmer is not responding

```

-아두이노에 업로딩이 되지 않음

-오류에 대해 알아보고 조사한 뒤 코드를 다시 만들어보고 연결된 선들을 확인하였는데 문제가 해결되지 않음

-아두이노pro mini에 문제(USB UART와 연결된 부분에 접촉불량과 합선같은 문제가 있음)가 있음을 확인하고 새로운 아두이노pro mini에 연결함

-다시 업로드 한 뒤 아래와 같이 제대로 업로드 되었음을 확인함

```

avrdude: 4926 bytes of flash written
avrdude: verifying flash memory against C:\Users\JINWOOK\AppData\Local\Temp\arduino_build_976075/4
avrdude: load data flash data from input file C:\Users\JINWOOK\AppData\Local\Temp\arduino_build_976075/sketch_may21a.ino.hex
avrdude: input file C:\Users\JINWOOK\AppData\Local\Temp\arduino_build_976075/sketch_may21a.ino.hex
avrdude: reading on-chip flash data:

Reading | ##### | 100% 1.01s

avrdude: verifying ...
avrdude: 4926 bytes of flash verified

avrdude done. Thank you.

```

3) 시리얼 모니터에 나오는 값을 확인후 알맞게 calibration factor 조정

-업로드후 시리얼 모니터에 출력되는 값 확인

```

COM3

20:29:25.394 -> 0.1
20:29:25.465 -> 0.1
20:29:25.573 -> 0.1
20:29:25.680 -> 0.1
20:29:25.790 -> 0.1
20:29:25.902 -> 0.1
20:29:25.995 -> 0.1
20:29:26.093 -> 0.2
20:29:26.197 -> 0.2
20:29:26.305 -> 0.2
20:29:26.410 -> 0.2
20:29:26.482 -> 0.2
20:29:26.590 -> 0.2

```

-조원들과 토의에서 출력되는 값과 측정하는 무게가 일대일 대응이 되도록 calibration factor를 정하고자 했다.

-정확한 영점이 맞추어 지지 않았다. 크랭크암에 힘을 가하지 않고 가만히 놔두어도 출력 값이 계속 변한다.

-Calibration factor를 1로 정하면 출력 값이 다섯 자리 수 이상으로 크게 나와서 7000, 10000 정도로 했더니 값이 충분히 작아졌다.

-휘트스톤브리지를 제작하면서 생각했던 banding에 의한 변형만 출력하는 것을 확인할 수 있었다. 크랭크암을 잡아당겨 인장시키면 출력 값에 변화가 크게 나타나지 않지만 굽힘을 가하면 값이 충분히 변하는 것을 확인할 수 있었다.

-힘을 가한 후 다시 가만히 뒤도 출력 값이 처음 값과 가까운 지점까지 돌아가지 않는다. 측정이 제대로 되지 않는 이유는 strain gauge부착이 제대로 되지 않거나 회로 구성과정에서 문제가 있었을 수도 있다

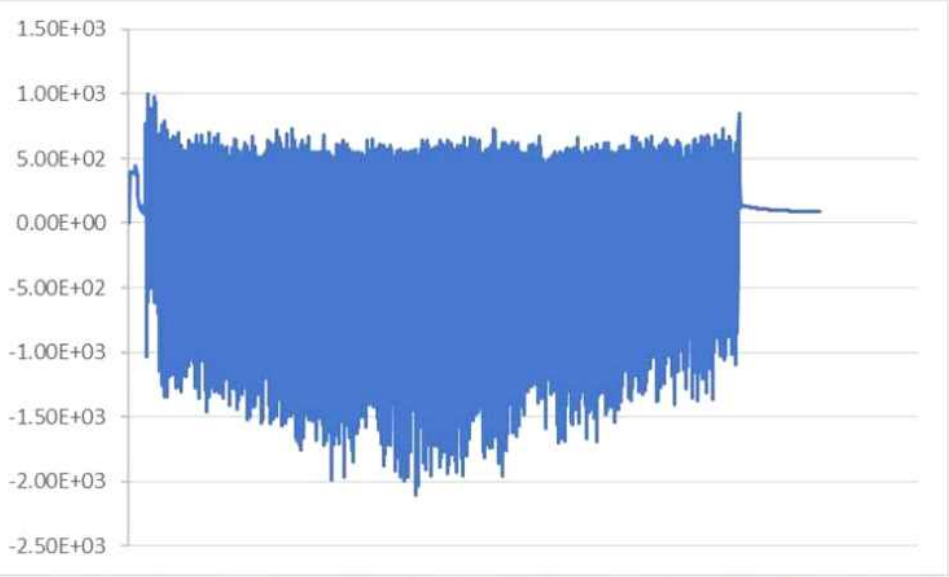
-크랭크암을 일단 고정하게 고정할 수 없었다. 적당한 calibration factor를 찾기 위해서는 크랭크암을 단단히 고정하지 못해 정확한 측정이 힘들다. 적당한 calibration factor를 찾기 위해서는 크랭크암을 단단히 고정시킨 후 실험을 해야 하는데 고정시킬 도구가 없어 실습실 사용이 가능한데로 다시 한번 실험이 필요하다고 생각된다.

•추후 논의할 내용

- 1) 적당한 calibration factor를 정하기위해 여러 무게를 달아보고, 모니터에 출력되는 값을 비교한 뒤 calibration factor 지정하기
- 2) 블루투스 모듈 연결한 것으로 블루투스가 잘 작동하는지 값을 확인해 봐야함
- 3) 시리얼통신이나 블루투스로 받은 값을 매트랩과 연동
- 4) 매트랩 코드작성에 대해 논의
- 5) 연결된 케이블들이 길이 정리가 필요함

Grade (A/B/C/F)		Delayed	
--------------------	--	---------	--

Weekly Report

Date	2020-06-12 2020-06-14	Team	14조
Report writer	18011230 배대위		
Members (Participants only)	18011276 강전욱 18013256 박현우 18011273 김용민 16011504 임진욱		
Topic	1. 데이터 변환 2. Torque 구하기 3. Torque 비교		
Discussions	<p>6/8 : 다양한 데이터를 얻기 위해 2번째 실험을 함.</p> <div style="text-align: center;">  </div> <p><그림1> 받은 strain 데이터를 Matlab을 통해 plot 그래프</p> <p>해석: Strain 값을 +값과 -값을 보면 값이 뒤집혔음을 볼 수 있음. 우리는 힘이 들어간 부분이 -값으로 나옴.</p> <p>문제제기 : 데이터를 조정해야함.</p>		

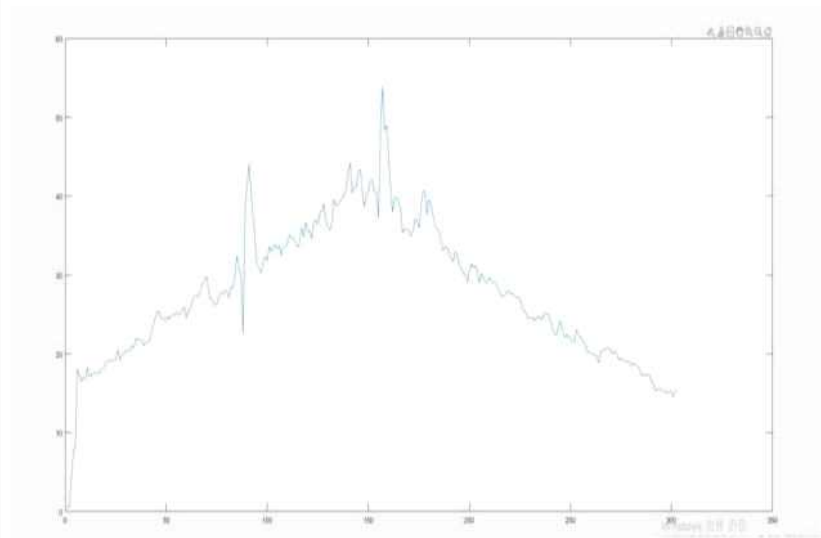
목표

: 와우키커의 Torque 값과 바뀐 데이터 Torque 값을 비교.

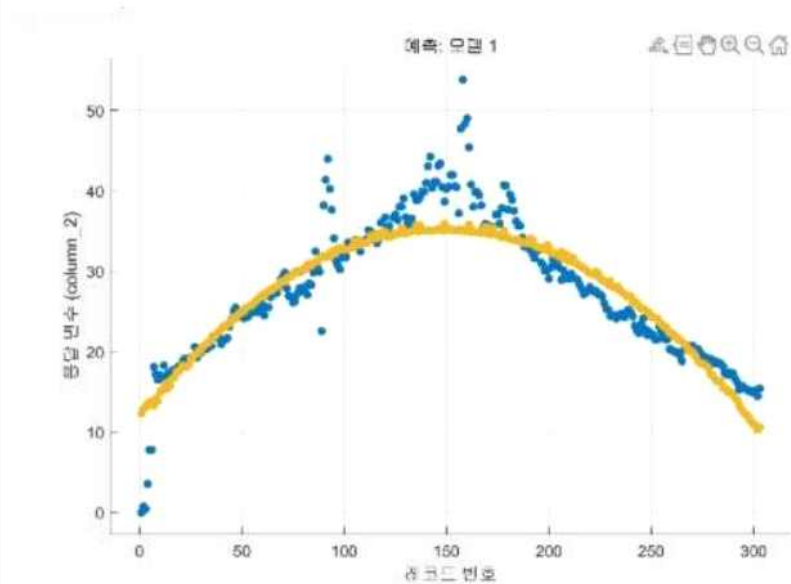
1. 와우키커의 토크

(1) 와우키커의 cadence는 rpm이므로 rad/s로 변경함.

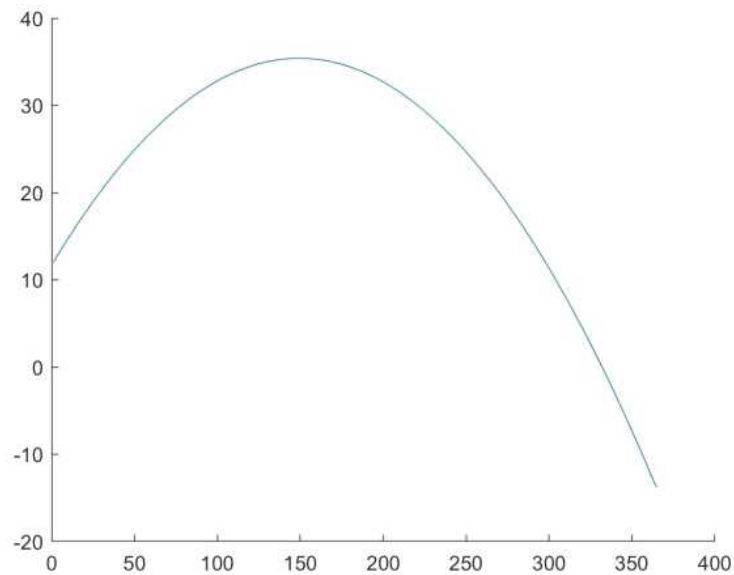
(2) 와우키커의 Power와 cadence를 이용해 $\text{Power} / \omega = \text{Torque}$ 를 구함.



<그림2> 와우키커 Torque



<그림3> 와우키커 Torque regression Model(2차 SVM)



<그림4> 와우키커 Torque regression Model(2차SVM)

2. 우리의 T(토크)

(1). Matlab 코드를 활용하여 Strain과 time을 각각 5개씩 데이터를 저장함.
Strain 값은 최대값(Strain_max), time은 평균값(Time_avg)으로 저장함.

(1.1) 데이터를 5개씩 묶어 저장한 이유

와우키커의 Power는 1초에 1번씩 데이터 입력되고 파워 미터의 strain은 0.2초씩 데이터 입력되기 때문에 Strain 신호는 5개씩 묶어 1초가 되기 위함.

```
clc;
clear;

a = xlsread('data.xlsx');
[x,y] = size(a);

for i = 1 : 1 : x
    a(i,2) = - a(i,2);
end
e = 1;
b=[];
for i = 1 : 5 : x
```

```

w = 0;
q = 0;
r = 1;
t = 0;
m = [];
for k = i : 1 : i + 4      #데이터를 5개씩 묶어 저장하는 과정.
    if k > x
        break
    end
    m(1,r) = a(k,2);
    w = w + a(k,1);
    r = r + 1;
    t = t + 1;
end
q = max(m);
w = w/t;
b(e,1) = w;
b(e,2) = q;
e = e + 1;
end
b(:,2) = b(:,2).*0.02;

d = xlsread('dat.xlsx');
dat = [d(:,1),d(:,2)];

save dat.txt dat -ascii
c = load('dat.txt');

[trainedModel, validationRMSE] = trainRegressionModel(c);
yfit = trainedModel.predictFcn(b(:,1));
hold on
plot(b(:,1),b(:,2))
plot(yfit)

```

(2). 와우키커의 cadence는 RPM이므로 rad/s로 변경함.

(3) 계산된 와우키커의 Torque와 조정시킨 Strain_max 값을 나눔.

이는 $T = rAE\epsilon$ 에서 (rAE)값과 같음 . (rAE = Constant)

$Torque / Strain_max = Constant$

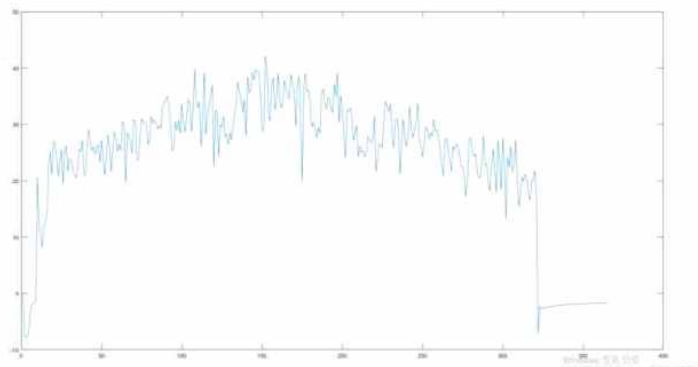
(4) Constant 값의 평균을 구함.

이 때 값이 Constant_avg = 0.02

(5) 5개씩 묶어서 나온 최댓값 Strain_max 값과 Constant_avg 값을 각각 곱해서 Torque 값을 도출해냄.

$Strain_max * Constant_avg = Torque$

(6) 이 때 나온 Torque 그래프.



<그림5> 파워미터 Torque

c.종합설계 교과목 발표자료

-우산털이 프로젝트



Background

- ◆ 우산 비닐커버 연간 사용량 1억장
- > CO2 발생량 698톤
- > 부패기간 100년 이상

구분	단위	2019년	2020년	2021년	2022년
우산 비닐커버 사용량	1억장	1.0	1.0	1.0	1.0
CO2 발생량	톤	698	698	698	698
부패기간	년	100	100	100	100

- ◆ 우산 비닐커버 점진적 금지

종합설계, Sejong University

Team 2

Background

현재 사용중인 우산 비닐 대체용품



- 수동으로 사용하기 때문에 개인마다 탈수 정도가 다름

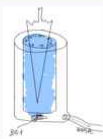
종합설계, Sejong University

Team 2

Problem Definition

목적

자동으로 우산을 탈수 시키는 장치 제작



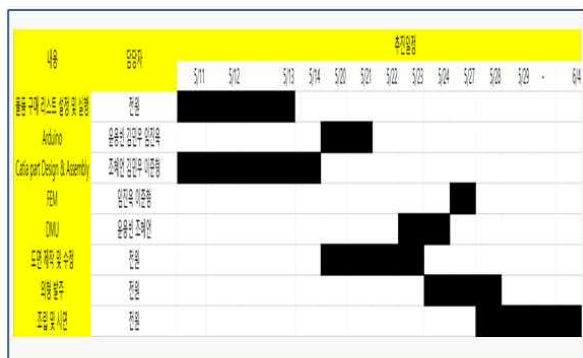
목표 및 기대효과

- ◆ 우산에서 물이 떨어지지 않을 정도의 탈수
- ◆ 사용시간을 5~6초로 설정
- ▶ 바닥의 물로 인한 미끄러짐 사고 예방
- ▶ 비닐 사용 감소

종합설계, Sejong University

Team 2

Gantt Chart



종합설계, Sejong University

Team 2

QFD

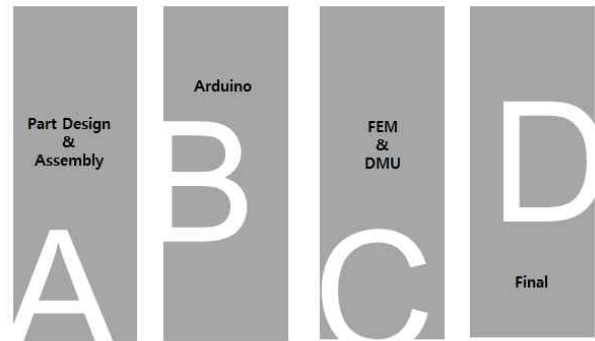
우산 탈수기의 QFD					
	무게	재질	구조	VOC 중요도	VOC 점수
탈수의 정도	3	9	9	1	21
탈수의 속도	1	9	9	0.75	14.25
제품의 내구성	1	9	9	0.75	14.25
제품 사용시 안전성	3	3	9	1	15
에너지 경제성	3	3	3	1	9
공학사양 중요도	11	33	39		

탈수 정도를 우선시하고 제품 사용시 안전하게 설계하는 것이 목표
따라서 제품의 구조가 가장 중요하다고 판단

중합설계, Sejong University

Team 2

Progress

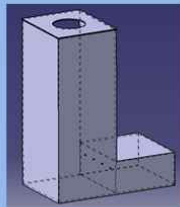


중합설계, Sejong University

Team 2

작동 원리

-제품 구성-



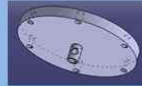
하우징(아크릴)



내부원통(아크릴)



중간지지대(아크릴)



브라켓(AL)



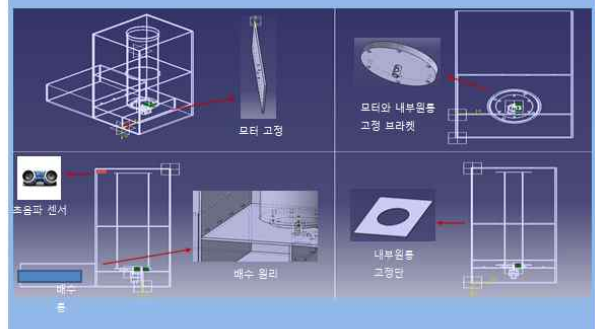
모터

중합설계, Sejong University

Team 2

작동원리

-부품별 용도-

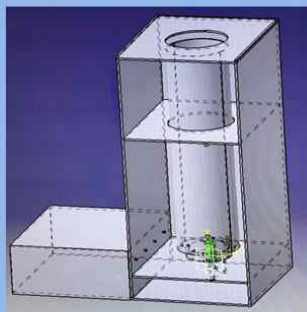


중합설계, Sejong University

Team 2

DMU

-작동 시뮬레이션-

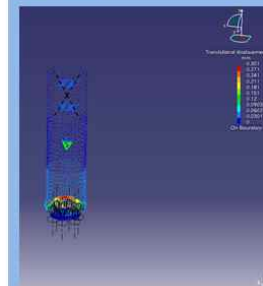


중합설계, Sejong University

Team 2

FEM

-내부 원통-



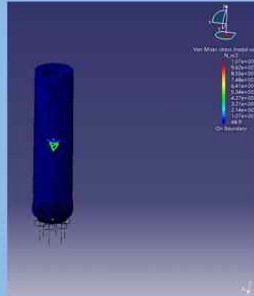
- ◆ 내부 원통과 알루미늄 브라켓에 자중이 부여되었을 때 변위 분석
- ▶ 원통의 아래부분에서 변형이 가장 큼

중합설계, Sejong University

Team 2

FEM

-내부 원통-



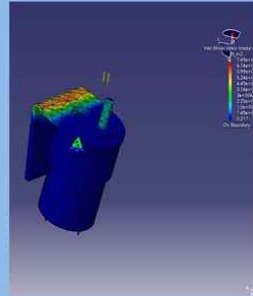
- ◆ 내부 원통과 알루미늄 브라켓에 자중이 부여되었을 때 응력 분석
- ▶ 내부의 응력이 거의 일정하기 때문에 원통과 브라켓이 결합된 상태에서 구조적으로 큰 문제가 없을 것이라 판단

중합설계, Sejong University

Team 2

FEM

-모터-

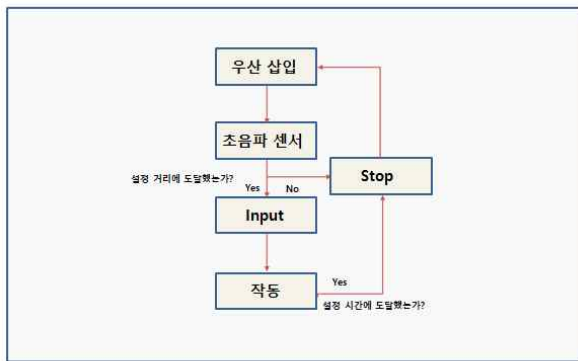


- ◆ 모터 축에 내부 원통과 브라켓의 무게가 주어졌을 때 응력 분석
- ▶ 모터 브라켓이 변형되는 것을 방지하기 위해 추가적인 구조 수정 필요

중합설계, Sejong University

Team 2

Arduino 설계안



중합설계, Sejong University

Team 2

Arduino code

```

void loop(){
  digitalWrite(triggerPin,HIGH);
  delay(Microseconds(10));
  digitalWrite(triggerPin,LOW);
  distance = pulseIn(echoPin,HIGH) / 50;
  distance = distance > 14? 14:distance;
  Serial.println("Distance(cm) = " + String(distance));
  if(distance == 14){
    delay(1500);
    motor_forward();
    delay(3500);
    motor_off();
    delay(100);
    motor_back();
    delay(3500);
    motor_off();
    delay(1500);
  }
  else{
    motor_off();
  }
}
  
```

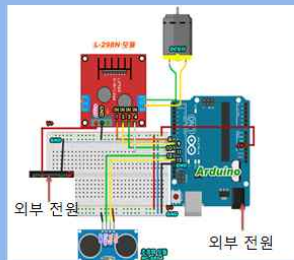
CM 단위로
변경

모터 출력

중합설계, Sejong University

Team 2

Arduino 회로도 및 사용 모터



12V 모터를 사용하기 때문에 외부 전원을 (아두이노 모듈 + 모터 드라이버) 총 2개 나누어 연결



*GGM DC모터 기본 스펙

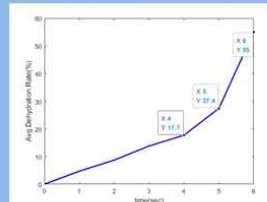
- 12V
- 217 RPM
- 0.9 kgf · cm
- 샤프트규격 6파이 D컷 타입

중합설계, Sejong University

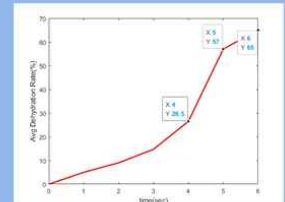
Team 2

탈수 비율 비교

-회전 방향 설정-



Forward Rotation



Forward + Backward Rotation

6초 동안의 탈수율을 10번 실험하여 결과를 측정

정방향 회전만 하는것 보다 정방향과 역방향 회전을 사용하는 것이 탈수 효율 증가

중합설계, Sejong University

Team 2

Discussion



Trial & Error

도면 작성 실수

도면 작성을 완벽히 한 줄 알았지만
가공자의 실수와 조립시 유격이 발생하여 재가공을 진행하였음

-> 추가적인 비용 발생

상세설계의 중요성을 깨닫는 등
은정함이 됨

FEM의 중요성

FEM 분석 결과 저압 모터부분의 브
라켓이 하중을 버티지 못하여 축의
회로가 발생하였음

-> 축을 고정하기 위해 지지대 보완



Problem Occurrence

탈수 비율이 예상보다 낮음

우산의 밑 부분이 매트에 닿지 않아
탈수량이 적음

해결 방안

내부원형을 우산 형상에 맞게 고갈
영으로 제작할 수 있다면 해결 가능

내부원형 직경이 좁아 우산이 접히는 안쪽
면에 탈수가 거의 안되었음

해결 방안

원심력을 이용하여 안쪽에 있는 물기가 바
깅으로 나와 매트로 탈수 할 수 있도록 해야

종합설계, Sejong University

Team 2

Results

우산의 탈수 비율이 (50)% 이면 바닥에 물이 흐르지 않음

제품 사용시 우산의 탈수 비율이 (57)% 달성

'우산에서 물이 떨어지지 않을 정도' 라는 목표 달성

개선 방안:

흡수와 배수가 더 좋은 매트 장착

매트 사용과 동시에 에어 컴프레서를 이용하여 바람으로 탈수를 돕는다면
효율이 증가할 것

종합설계, Sejong University

Team 2

소요 비용

항목	단가	개수	금액
하우징 아크릴가공	308,000	1	308000
알루미늄 브라켓가공	66,000	1	66000
아두이노 관련부품	61,050	1	61050
하우징 수정가공	11,000	1	11000
브라켓 볼트,너트	3,000	1	3000
글루건, 추가 볼트	11,500	1	11500
아두이노 우노 (추가)	10,300	1	10300
매트, 접착제	44,550	1	44550
추가 매트,접착제	25,150	1	25150
합계 금액			540,550

종합설계, Sejong University

Team 2

Reference

- ◆ https://www.breaknews.com/sub_read.html?uid=526376
- ◆ CATIA V5 기초와 응용 (ver 5.9, 과학기술)
- ◆ 세종대 종합설계 FEM, DMU 강의록 (전용태 교수님)
- ◆ <https://rasino.tistory.com/178> (Arduino 코드 및 회로도)

Advanced Mechanical Design, Sejong University

Team 1