Cartridge Fuses and temperature Fuses status detection after fire

임규영 (Kyu Young Lim)

Contents

01

Images Set(796 images)

Fuses Status

- **Cartridge Fuses Status**

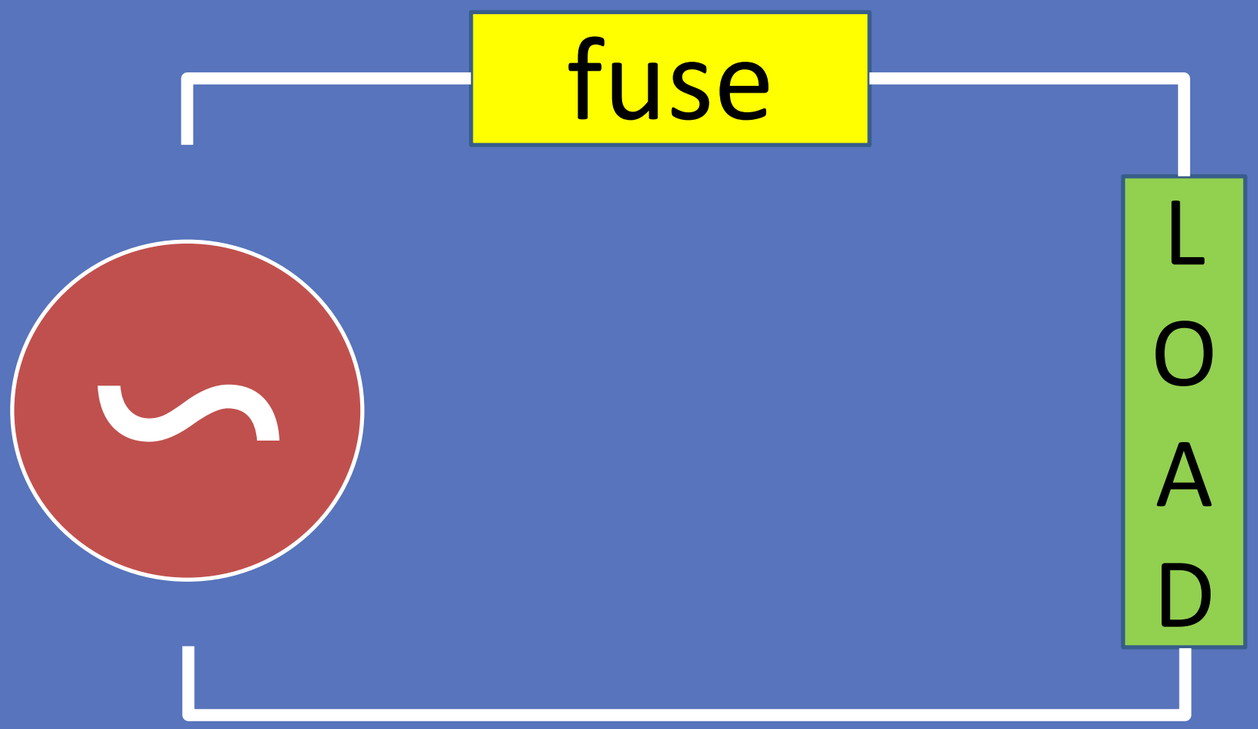
1) Not Cutoff(571)
2) Electrical Cutoff(150)
3) Fire Cutoff(59)

- **Temperature Fuses Status**

1) Not Cutoff(3)
2) Cutoff(13)

< If it exceeds 104 degrees, the fuse blows >

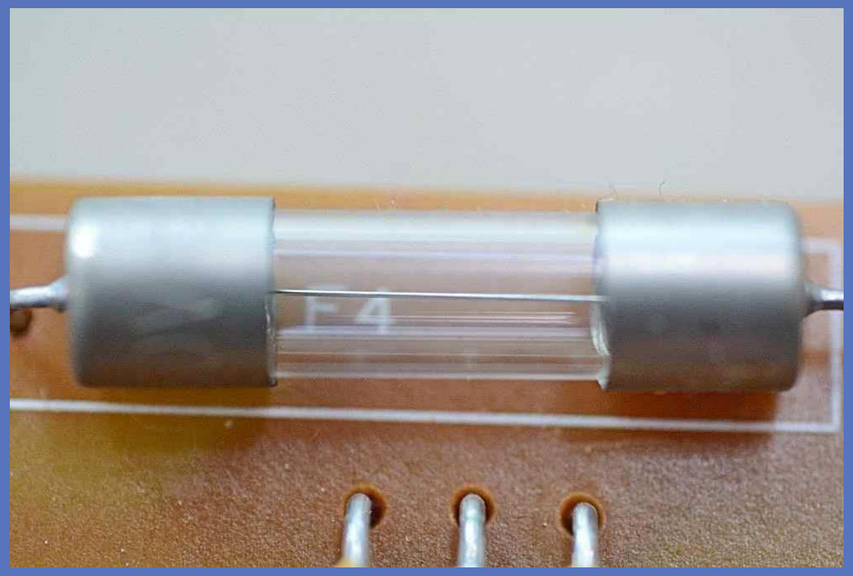
Meaning of fuse status in fire cause investigation

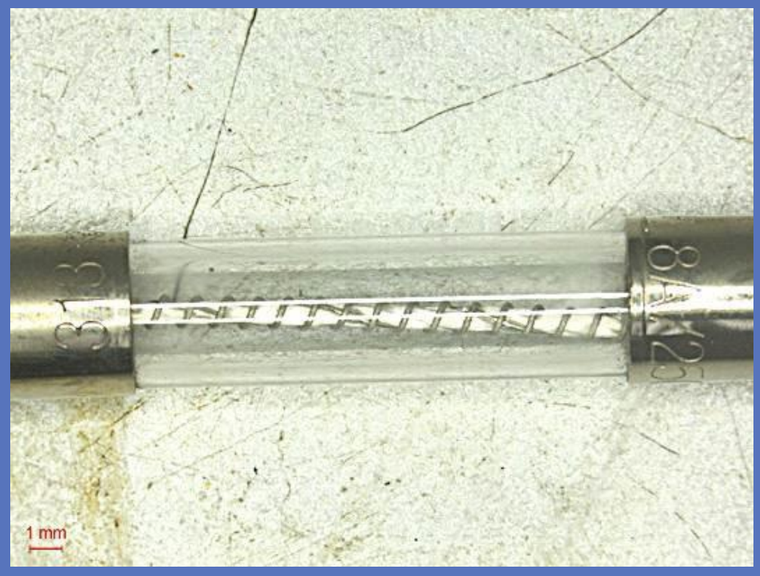


✓ Overcurrent blows the fuse

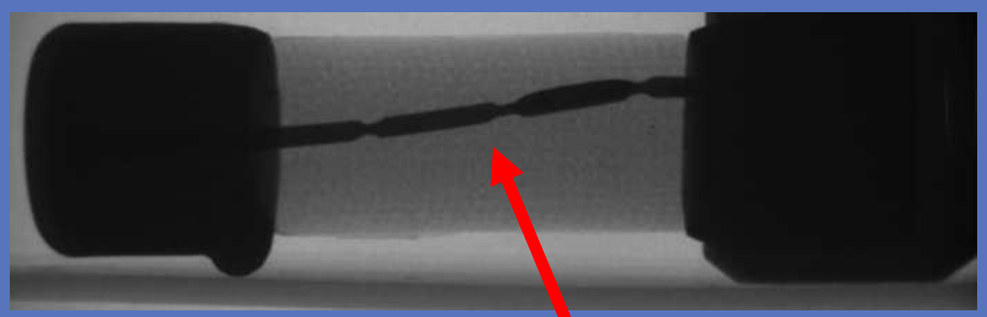
Cartridge Fuses Status

▪ Cartridge Fuses(Not Cutoff)

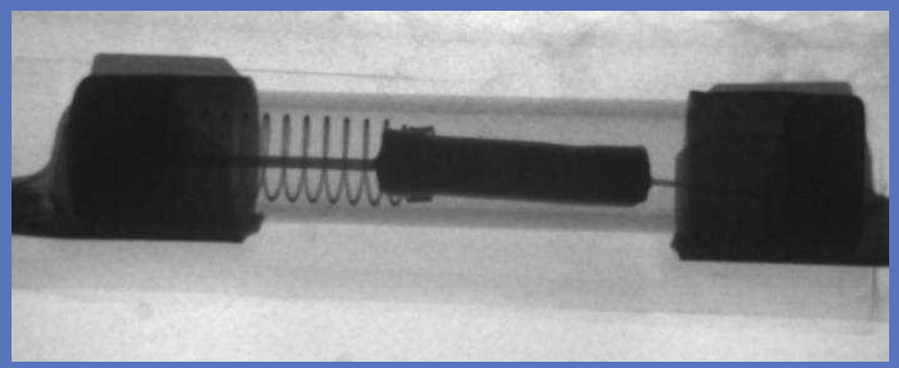


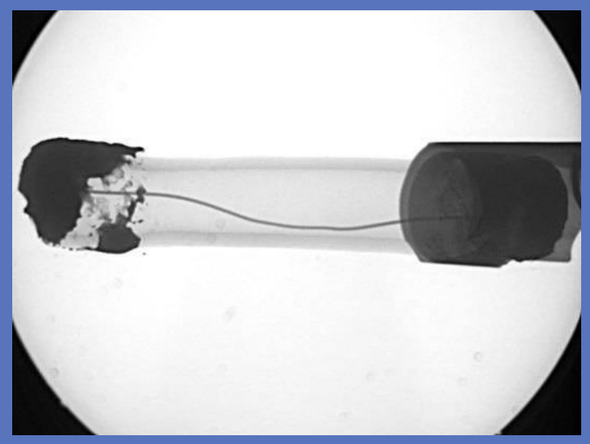


▪ Cartridge Fuses – X ray(Not Cutoff)



Element(가용체)

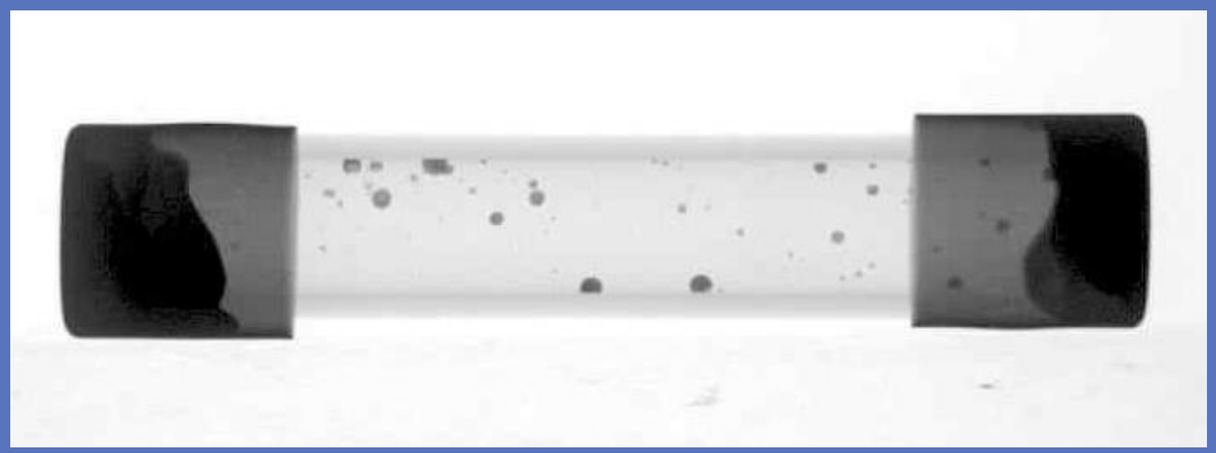


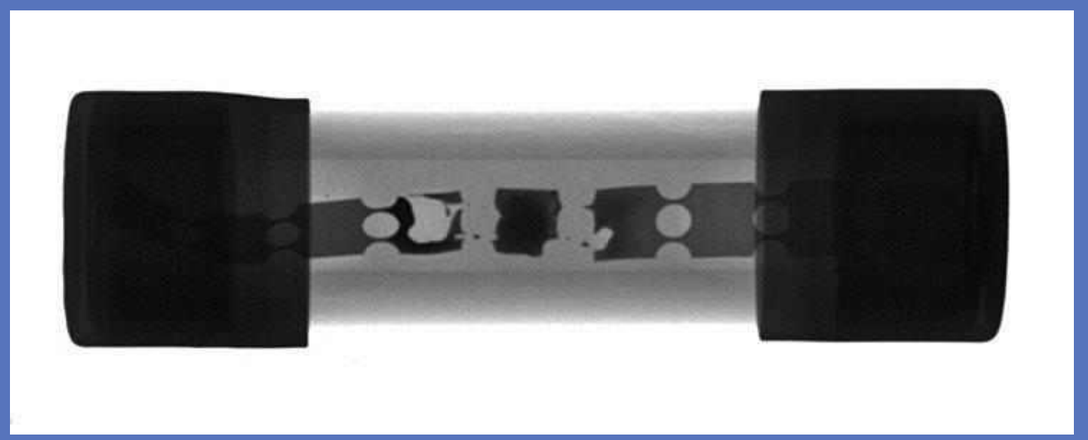


Cartridge Fuses Status

- Cartridge Fuses(Electrical Cutoff)

- Cartridge Fuses – X ray(Electrical Cutoff)

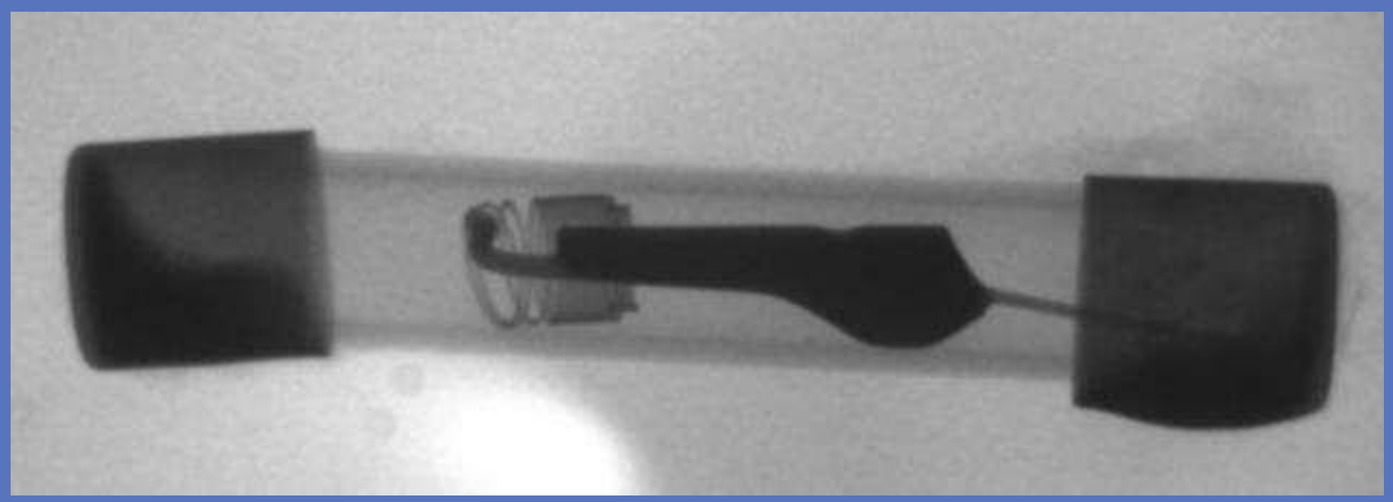


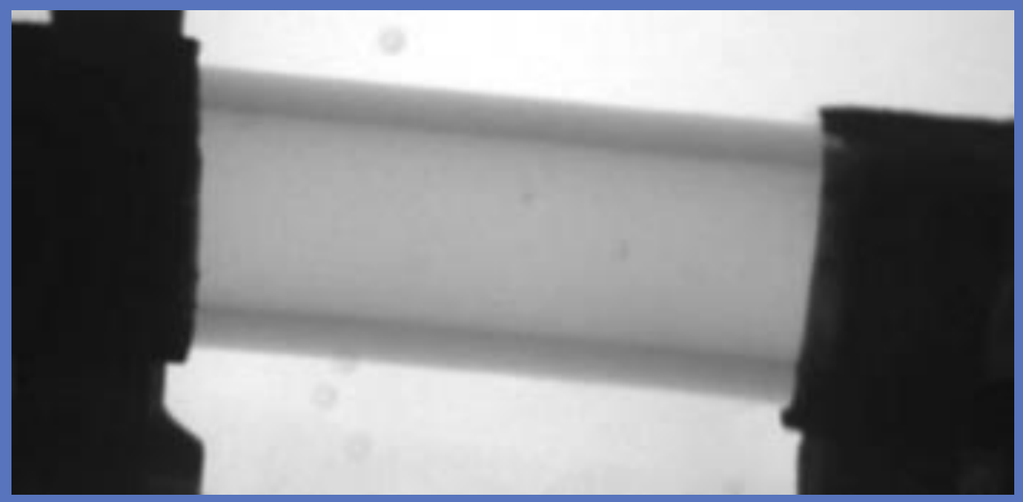


Cartridge Fuses Status

- Cartridge Fuses(Fire Cutoff)

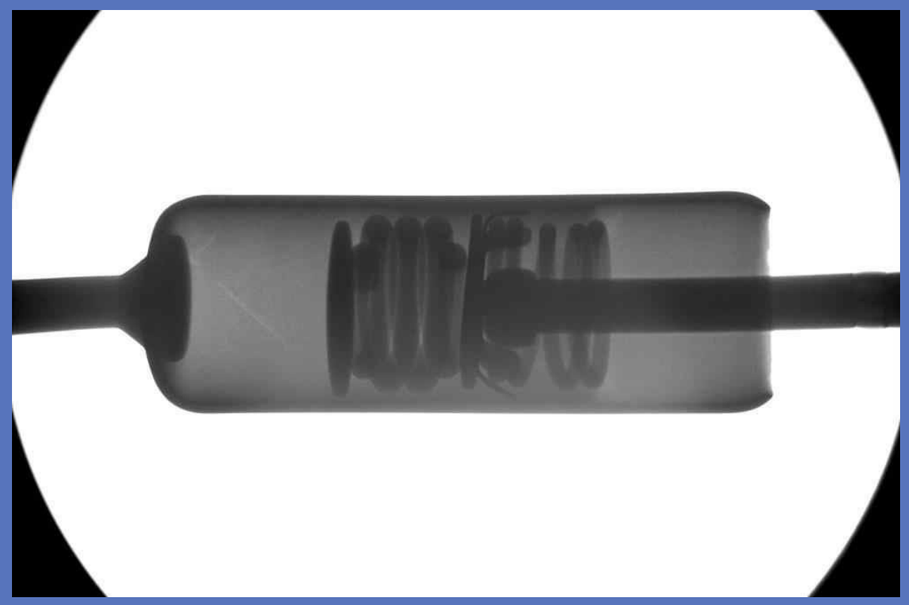
- Cartridge Fuses – X ray(Fire Cutoff)

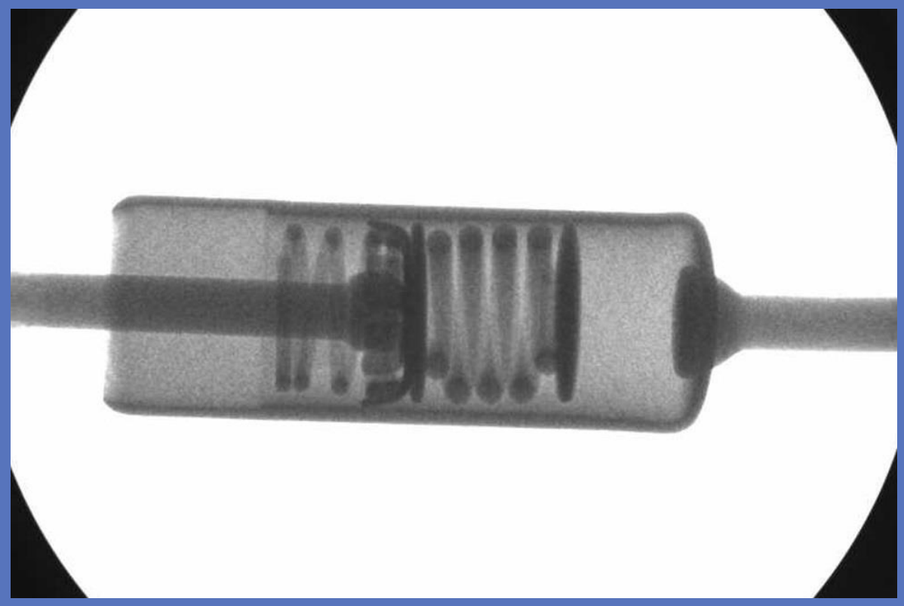




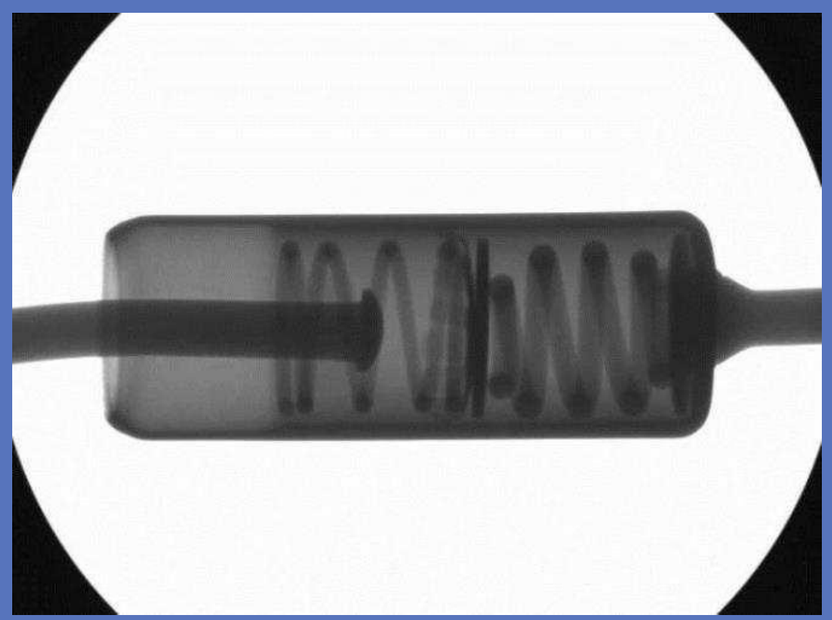
Temperature Fuses Status

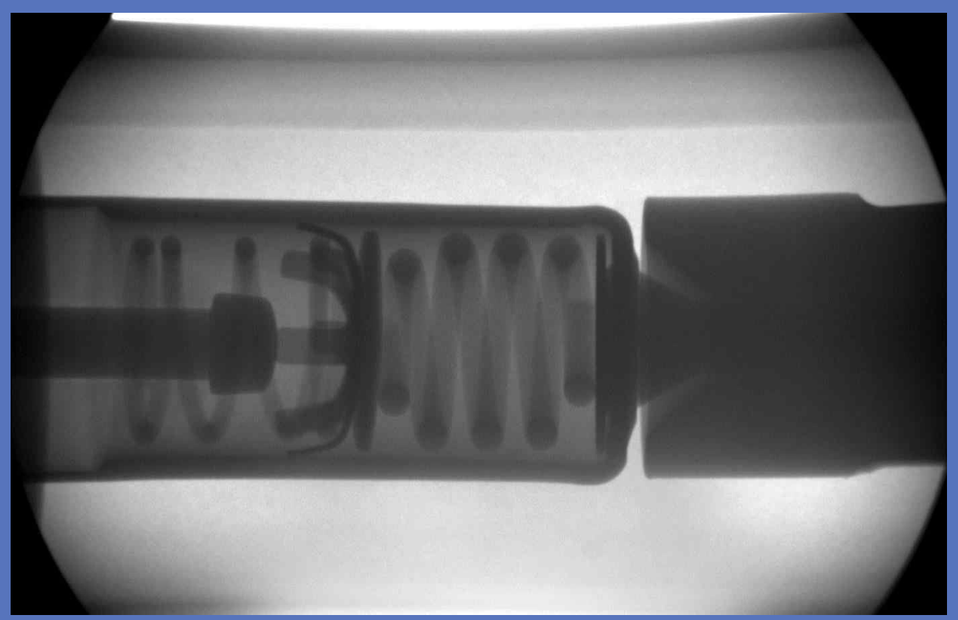
- Temperature Fuses(Not Cutoff)





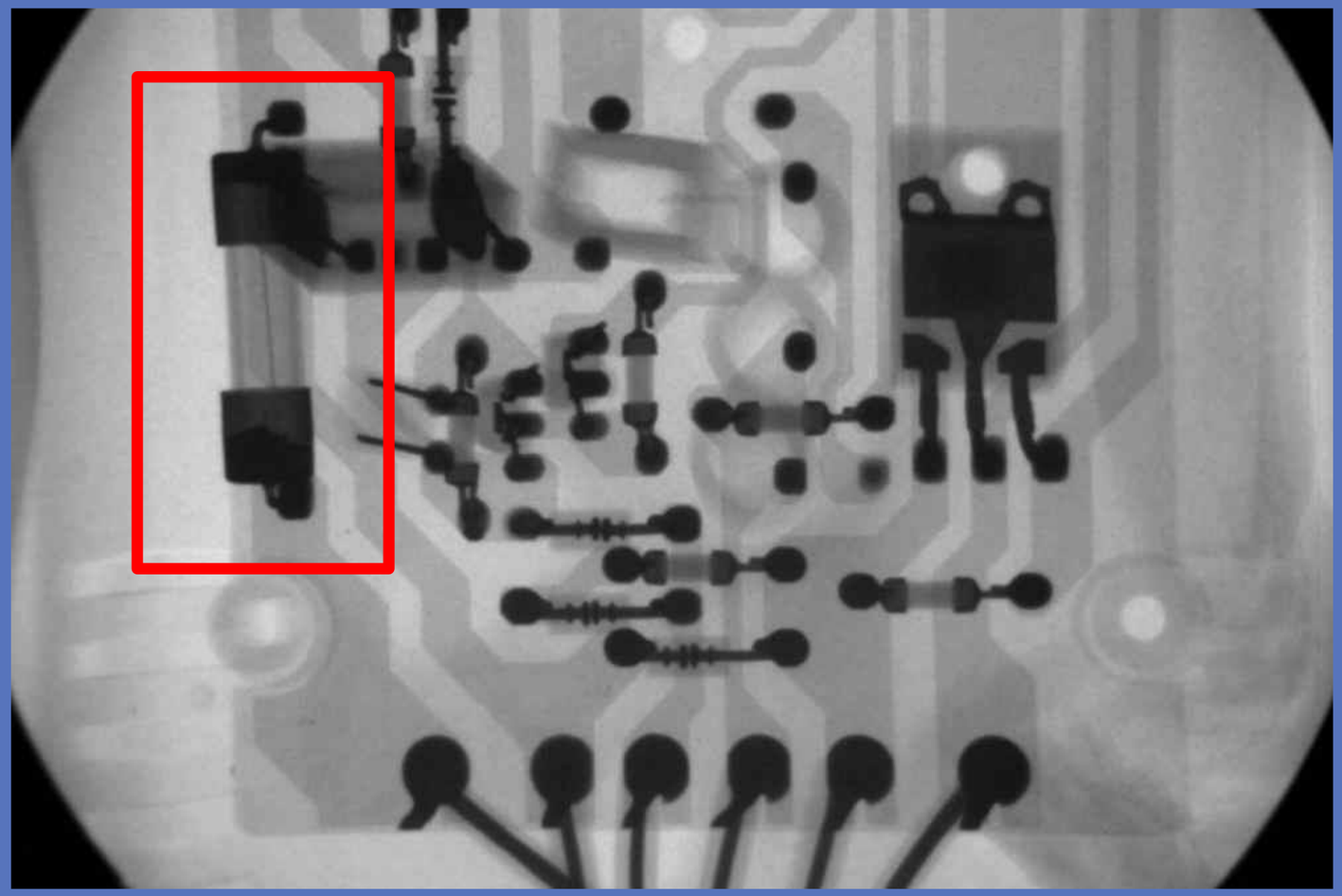
- Temperature Fuses(Cutoff)



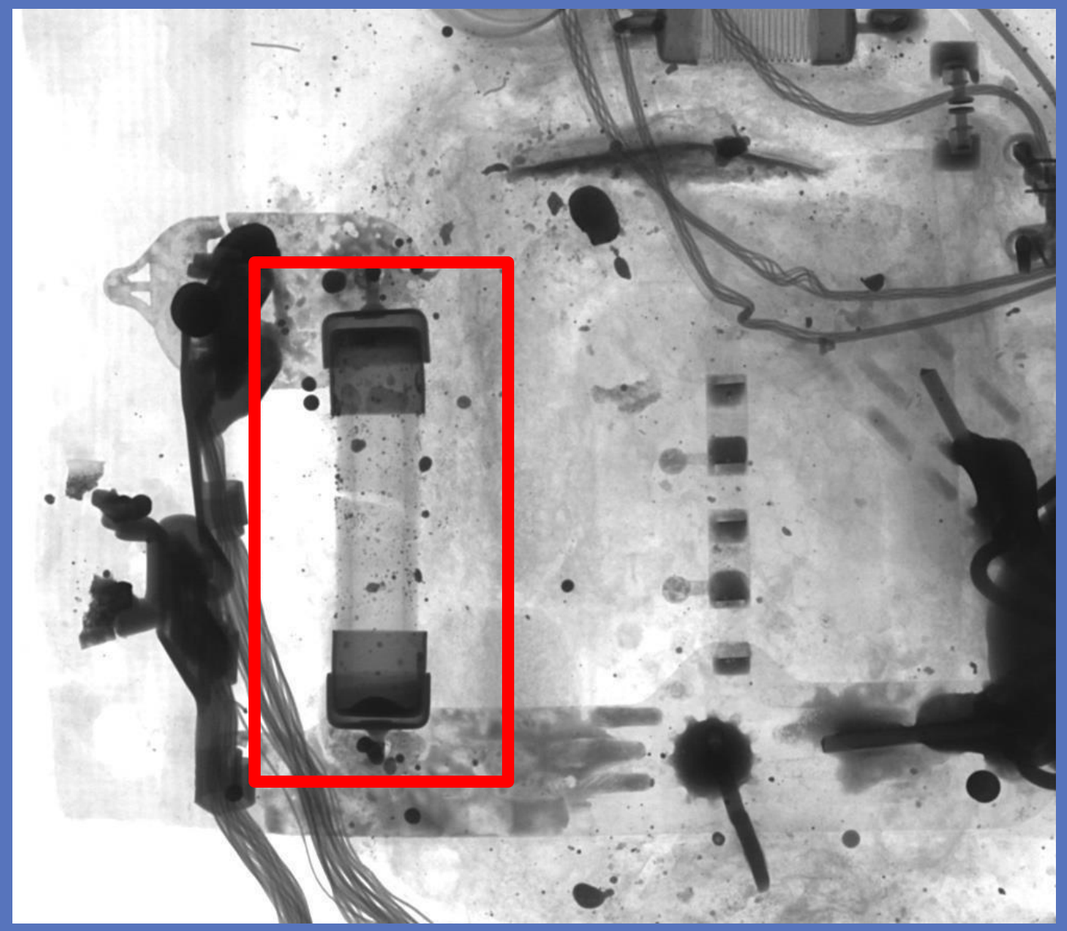


Temperature Fuses Status

- Some images



Cartridge Fuse – Not Cutoff



Cartridge Fuse – Electrical Cutoff

02

Train with Densenet-201

Total Code

```python
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import tensorflow as tf
import cv2
from tensorflow.keras.utils import to_categorical
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, log_loss, accuracy_score
from sklearn.model_selection import train_test_split
from tqdm import tqdm
import time
import datetime

start_time = time.time()

data_dir = "C:/pattern_recognition/fuse_status_detection/images"
Name = os.listdir(data_dir)

#------------ Parameters --------------#

image_resize_width = 500
image_resize_height = 300
pooling_value = 'avg'
batch_size_value = 50
epochs_value = 300
hidden_layer_value = 64

#------------------------------------#

data=[]
data_file = []
data_file_true = []
labels=[]
count=0

for name in tqdm(os.listdir(data_dir)):
    path=os.path.join(data_dir,name)
    for im in os.listdir(path):
        data_file.append(os.path.join(path,im))
        data_file_true.append(name)
        image=cv2.imread(os.path.join(path,im))
        if type(image)==np.ndarray:
            image2=cv2.resize(image,dsize=(image_resize_width,image_resize_height),interpolation=cv2.INTER_LINEAR)
            data+=[image2]
            labels+=[count]
    count=count+1
    print(name)

data=np.array(data)
labels1=np.array(labels)
labels1=to_categorical(labels1)
labels1=np.array(labels1)

train_data,test_data,train_labels,test_labels=train_test_split(data,labels1,test_size=0.2,random_state=44)
```

```python
datagen = ImageDataGenerator(horizontal_flip=True,vertical_flip=True,rotation_range=20,zoom_range=0.2,
                         width_shift_range=0.2,height_shift_range=0.2,shear_range=0.1,fill_mode="nearest")

pretrained_model3 = tf.keras.applications.DenseNet201(input_shape=(image_resize_height,image_resize_width,3),include_top=False,weights='imagenet',pooling=pooling_value)
pretrained_model3.trainable = False

inputs3 = pretrained_model3.input
hidden3 = tf.keras.layers.Dense(hidden_layer_value, activation='relu')(pretrained_model3.output)
outputs3 = tf.keras.layers.Dense(len(Name), activation='softmax')(hidden3)
model = tf.keras.Model(inputs=inputs3, outputs=outputs3)

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

his=model.fit(datagen.flow(train_data,train_labels,batch_size=batch_size_value),validation_data=(test_data,test_labels),epochs=epochs_value)

model.save('C:/pattern_recognition/fuse_status_detection/model.h5')

y_pred=model.predict(test_data)
pred=np.argmax(y_pred,axis=1)
ground = np.argmax(test_labels,axis=1)
print(classification_report(ground,pred))

print(model.summary())

get_acc = his.history['accuracy']
value_acc = his.history['val_accuracy']
get_loss = his.history['loss']
validation_loss = his.history['val_loss']

epochs = range(len(get_acc))
plt.plot(epochs, get_acc, 'r', label='Accuracy of Training data')
plt.plot(epochs, value_acc, 'b', label='Accuracy of Validation data')
plt.title('Training vs validation accuracy')
plt.legend(loc=0)
plt.savefig('C:/pattern_recognition/fuse_status_detection/accuracy_graph.jpg')
plt.close()

epochs = range(len(get_loss))
plt.plot(epochs, get_loss, 'r', label='Loss of Training data')
plt.plot(epochs, validation_loss, 'b', label='Loss of Validation data')
plt.title('Training vs validation loss')
plt.legend(loc=0)
plt.savefig('C:/pattern_recognition/fuse_status_detection/loss_graph.jpg')
plt.close()

pred2=model.predict(data)
print(pred2.shape)
pred3 = pred2

PRED=[]
PRED1=[]
```

Total Code

```python
for item in pred2:
    value2=np.argmax(item)
    PRED += [value2]
    if value2 == 0:
        value2 = 'fuse_electrical_cutoff'
    if value2 == 1:
        value2 = 'fuse_fire_cutoff'
    if value2 == 2:
        value2 = 'fuse_not_cutoff'
    if value2 == 3:
        value2 = 'tempfuse_cutoff'
    if value2 == 4:
        value2 = 'tempfuse_not_cutoff'
    PRED1+=[value2]

data_file = np.array(data_file)
data_file = np.reshape(data_file, (len(data_file), 1))

data_file_true = np.array(data_file_true)
data_file_true = np.reshape(data_file_true, (len(data_file_true), 1))

PRED1 = np.array(PRED1)
PRED1 = np.reshape(PRED1, (len(PRED1), 1))

title = np.array(['파일', '참값', '예측값', 'fuse_electrical_cutoff', 'fuse_fire_cutoff', 'fuse_not_cutoff', 'tempfuse_cutoff','tempfuse_not_cutoff'])
title = np.reshape(title, (1, len(title)))

result = np.hstack((data_file, data_file_true))
result = np.hstack((result, PRED1))
result = np.hstack((result, pred2))
result = np.vstack((title, result))

result_csv = pd.DataFrame(result)
result_csv.to_csv('C:/pattern_recognition/fuse_status_detection/result.csv', mode='w', encoding='euc-kr')

ANS=labels
accuracy=accuracy_score(ANS,PRED)
print('accuracy =', accuracy)

fw = open('C:/pattern_recognition/fuse_status_detection/accuracy.txt', 'w', -1, 'utf-8')
fw.write(str(accuracy))
fw.close()

end_time = time.time()
code_time = (end_time - start_time)
code_time = str(datetime.timedelta(seconds=code_time)).split(".")
```

```python
parameters_value = []
parameters_value.append('image_resize_width')
parameters_value.append(str(image_resize_width))
parameters_value.append('image_resize_height')
parameters_value.append(str(image_resize_height))
parameters_value.append('pooling_value')
parameters_value.append(str(pooling_value))
parameters_value.append('batch_size_value')
parameters_value.append(str(batch_size_value))
parameters_value.append('epochs_value')
parameters_value.append(str(epochs_value))
parameters_value.append('time')
parameters_value.append(str(code_time[0]))

fw = open('C:/pattern_recognition/fuse_status_detection/parameters.txt', 'w', -1, 'utf-8')
for i in range(len(parameters_value)):
    fw.write(parameters_value[i]+'\n')
fw.close()
```

Packages

```python
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
import tensorflow as tf
import cv2
from tensorflow.keras.utils import to_categorical
from tensorflow.python.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, log_loss, accuracy_score
from sklearn.model_selection import train_test_split
from tqdm import tqdm
import time
import datetime
```

- time, datetime : Time spent measuring

Parameters

```
#------------ Parameters ---------------#

image_resize_width = 500
image_resize_height = 300
pooling_value = 'avg'
batch_size_value = 50
epochs_value = 300
hidden_layer_value = 128

#---------------------------------------#
```

- Image resize witgh and height
- Pooling method = average pooling
- Batch size = 50
- Number of Epochs
- Number of Hidden layer

image resize

```
for name in tqdm(os.listdir(data_dir)):
    path=os.path.join(data_dir,name)
    for im in os.listdir(path):
        data_file.append(os.path.join(path,im))
        data_file_true.append(name)
        image=cv2.imread(os.path.join(path,im))
        if type(image)==np.ndarray:
            image2=cv2.resize(image,dsize=(image_resize_width,image_resize_height),interpolation=cv2.INTER_LINEAR)
            data+=[image2]
            labels+=[count]
    count=count+1
    print(name)
```

```
data=np.array(data)
labels1=np.array(labels)
labels1=to_categorical(labels1)
labels1=np.array(labels1)
```

- Image data(resized)
- Label data → Categorical

image assignment and Data augmentation

```
train_data,test_data,train_labels,test_labels=train_test_split(data,labels1,test_size=0.2,random_state=44)
```

- Train images = 637 (80%)
- Test images = 159 (20%)

```
datagen = ImageDataGenerator(horizontal_flip=True,vertical_flip=True,rotation_range=20,zoom_range=0.2,
                             width_shift_range=0.2,height_shift_range=0.2,shear_range=0.1,fill_mode="nearest")
```

- Data augmentation

Set input, hidden, output layer

```
pretrained_model3 = tf.keras.applications.DenseNet201(input_shape=(image_resize_height,image_resize_width,3),include_top=False,weights='imagenet',pooling=pooling_value)
pretrained_model3.trainable = False
```

- Input layer
- Cnn model = DenseNet201
- Weights = imagenet
- Pooling = average pooling

```
inputs3 = pretrained_model3.input
hidden3 = tf.keras.layers.Dense(hidden_layer_value, activation='relu')(pretrained_model3.output)
outputs3 = tf.keras.layers.Dense(len(Name), activation='softmax')(hidden3)
model = tf.keras.Model(inputs=inputs3, outputs=outputs3)
```

- Hidden layer : number of hidden layer, activation function(Relu)
- Output layer : activation function(Softmax)

Model train

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

- Optimizer = adam

```
his=model.fit(datagen.flow(train_data,train_labels,batch_size=batch_size_value),validation_data=(test_data,test_labels),epochs=epochs_value)
```

- Batch size, epochs
- Model train

```
model.save('C:/pattern_recognition/fuse_status_detection/model.h5')
```

- Trained Model save - *.h5

Calculate accuracy and loss

```
y_pred=model.predict(test_data)
pred=np.argmax(y_pred,axis=1)
ground = np.argmax(test_labels,axis=1)
print(classification_report(ground,pred))

print(model.summary())

get_acc = his.history['accuracy']
value_acc = his.history['val_accuracy']
get_loss = his.history['loss']
validation_loss = his.history['val_loss']

epochs = range(len(get_acc))
plt.plot(epochs, get_acc, 'r', label='Accuracy of Training data')
plt.plot(epochs, value_acc, 'b', label='Accuracy of Validation data')
plt.title('Training vs validation accuracy')
plt.legend(loc=0)
plt.savefig('C:/pattern_recognition/fuse_status_detection/accuracy_graph.jpg')
plt.close()

epochs = range(len(get_loss))
plt.plot(epochs, get_loss, 'r', label='Loss of Training data')
plt.plot(epochs, validation_loss, 'b', label='Loss of Validation data')
plt.title('Training vs validation loss')
plt.legend(loc=0)
plt.savefig('C:/pattern_recognition/fuse_status_detection/loss_graph.jpg')
plt.close()
```

- Accuracy calculation
- Loss calculation

Predict fuses status

```
pred2=model.predict(data)
```

- Using trained model, predict 796 images

```
for item in pred2:
    value2=np.argmax(item)
    PRED += [value2]
    if value2 == 0:
        value2 = 'fuse_electrical_cutoff'
    if value2 == 1:
        value2 = 'fuse_fire_cutoff'
    if value2 == 2:
        value2 = 'fuse_not_cutoff'
    if value2 == 3:
        value2 = 'tempfuse_cutoff'
    if value2 == 4:
        value2 = 'tempfuse_not_cutoff'
    PRED1+=[value2]

data_file = np.array(data_file)
data_file = np.reshape(data_file, (len(data_file), 1))

data_file_true = np.array(data_file_true)
data_file_true = np.reshape(data_file_true, (len(data_file_true), 1))

PRED1 = np.array(PRED1)
PRED1 = np.reshape(PRED1, (len(PRED1), 1))

title = np.array(['파일', '참값', '예측값', 'fuse_electrical_cutoff', 'fuse_fire_cutoff', 'fuse_not_cutoff', 'tempfuse_cutoff', 'tempfuse_not_cutoff'])
title = np.reshape(title, (1, len(title)))

result = np.hstack((data_file, data_file_true))
result = np.hstack((result, PRED1))
result = np.hstack((result, pred2))
result = np.vstack((title, result))

result_csv = pd.DataFrame(result)
result_csv.to_csv('C:/pattern_recognition/fuse_status_detection/result.csv', mode='w', encoding='euc-kr')

ANS=labels
accuracy=accuracy_score(ANS,PRED)
print('accuracy =', accuracy)

fw = open('C:/pattern_recognition/fuse_status_detection/accuracy.txt', 'w', -1, 'utf-8')
fw.write(str(accuracy))
fw.close()
```

- Result → save *.csv
- Accuracy → save *.txt

image assignment

```
end_time = time.time()
code_time = (end_time - start_time)
code_time = str(datetime.timedelta(seconds=code_time)).split(".")
```

- Time spent measuring

```
parameters_value = []
parameters_value.append('image_resize_width')
parameters_value.append(str(image_resize_width))
parameters_value.append('image_resize_height')
parameters_value.append(str(image_resize_height))
parameters_value.append('pooling_value')
parameters_value.append(str(pooling_value))
parameters_value.append('batch_size_value')
parameters_value.append(str(batch_size_value))
parameters_value.append('epochs_value')
parameters_value.append(str(epochs_value))
parameters_value.append('time')
parameters_value.append(str(code_time[0]))

fw = open('C:/pattern_recognition/fuse_status_detection/parameters.txt', 'w', -1, 'utf-8')
for i in range(len(parameters_value)):
    fw.write(parameters_value[i]+'\n')
fw.close()
```

- Parameters save *.txt

03

Results analysis

Result (*.csv)

	A	B	C	D	E	F	G	H	I
1		0	1	2	3	4	5	6	7
2	0	파일	참값	예측값	fuse_electrical_cutoff	fuse_fire_cutoff	fuse_not_cutoff	tempfuse_cutoff	tempfuse_not_cutoff
3	1	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0001.jpg	fuse_electrical_cutoff	fuse_not_cutoff	25%	5%	70%	0%	0%
4	2	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0002.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	88%	4%	8%	0%	0%
5	3	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0003.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	100%	0%	0%	0%	0%
6	4	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0004.jpg	fuse_electrical_cutoff	fuse_not_cutoff	4%	1%	95%	0%	0%
7	5	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0005.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	86%	0%	14%	0%	0%
8	6	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0006.jpg	fuse_electrical_cutoff	fuse_not_cutoff	3%	0%	97%	0%	0%
9	7	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0007.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	38%	23%	38%	0%	0%
10	8	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0008.jpg	fuse_electrical_cutoff	fuse_not_cutoff	37%	24%	38%	0%	0%
11	9	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0009.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	94%	0%	5%	0%	0%
12	10	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0010.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	90%	0%	10%	0%	0%
13	11	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0011.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	62%	2%	36%	0%	0%
14	12	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0012.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	96%	1%	3%	0%	0%
15	13	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0013.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	98%	0%	2%	0%	0%
16	14	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0014.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	77%	15%	8%	0%	0%
17	15	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0015.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	83%	11%	5%	0%	0%
18	16	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0016.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	88%	8%	4%	0%	0%
19	17	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0017.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	75%	23%	2%	0%	0%
20	18	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0018.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	83%	3%	14%	0%	0%
21	19	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0019.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	73%	5%	22%	0%	0%
22	20	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0020.jpg	fuse_electrical_cutoff	fuse_not_cutoff	1%	0%	99%	0%	0%
23	21	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0021.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	58%	6%	35%	0%	0%
24	22	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0022.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	72%	6%	22%	0%	0%
25	23	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0023.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	100%	0%	0%	0%	0%
26	24	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0024.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	100%	0%	0%	0%	0%
27	25	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0025.jpg	fuse_electrical_cutoff	fuse_not_cutoff	21%	35%	44%	0%	0%
28	26	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0026.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	86%	2%	12%	0%	0%
29	27	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0027.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	94%	0%	6%	0%	0%
30	28	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0028.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	66%	8%	26%	0%	0%
31	29	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0029.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	80%	0%	20%	0%	0%
32	30	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0030.jpg	fuse_electrical_cutoff	fuse_not_cutoff	7%	9%	84%	0%	0%
33	31	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0031.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	76%	15%	9%	0%	0%
34	32	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0032.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	76%	10%	14%	0%	0%
35	33	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0033.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	53%	12%	34%	0%	0%
36	34	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0034.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	39%	25%	36%	0%	0%
37	35	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0035.jpg	fuse_electrical_cutoff	fuse_not_cutoff	25%	2%	73%	0%	0%
38	36	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0036.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	64%	9%	27%	0%	0%
39	37	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0037.jpg	fuse_electrical_cutoff	fuse_electrical_cutoff	88%	10%	3%	0%	0%
40	38	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0038.jpg	fuse_electrical_cutoff	fuse_not_cutoff	11%	28%	61%	0%	0%

- How accurately did you predict the true value?

  (얼마나 참값을 정확하게 예측하였는가?) → True value

Result (*.csv)

True Value

	fuse_electrical_cutoff	fuse_fire_cutoff	fuse_not_cutoff	tempfuse_cutoff	tempfuse_not_cutoff	average
accuracy	68%	30%	89%	82%	59%	66%
number of images	150	59	571	13	3	

total accuracy	87%
image_resize_width	500
image_resize_height	300
pooling_value	avg
batch_size_value	50
epochs_value	200
time	2:35:33

- Predicts must exceed at least 90%

  (예측값은 최소 90%를 넘어야 증거력을 가질 수 있을 것임)

Result analysis

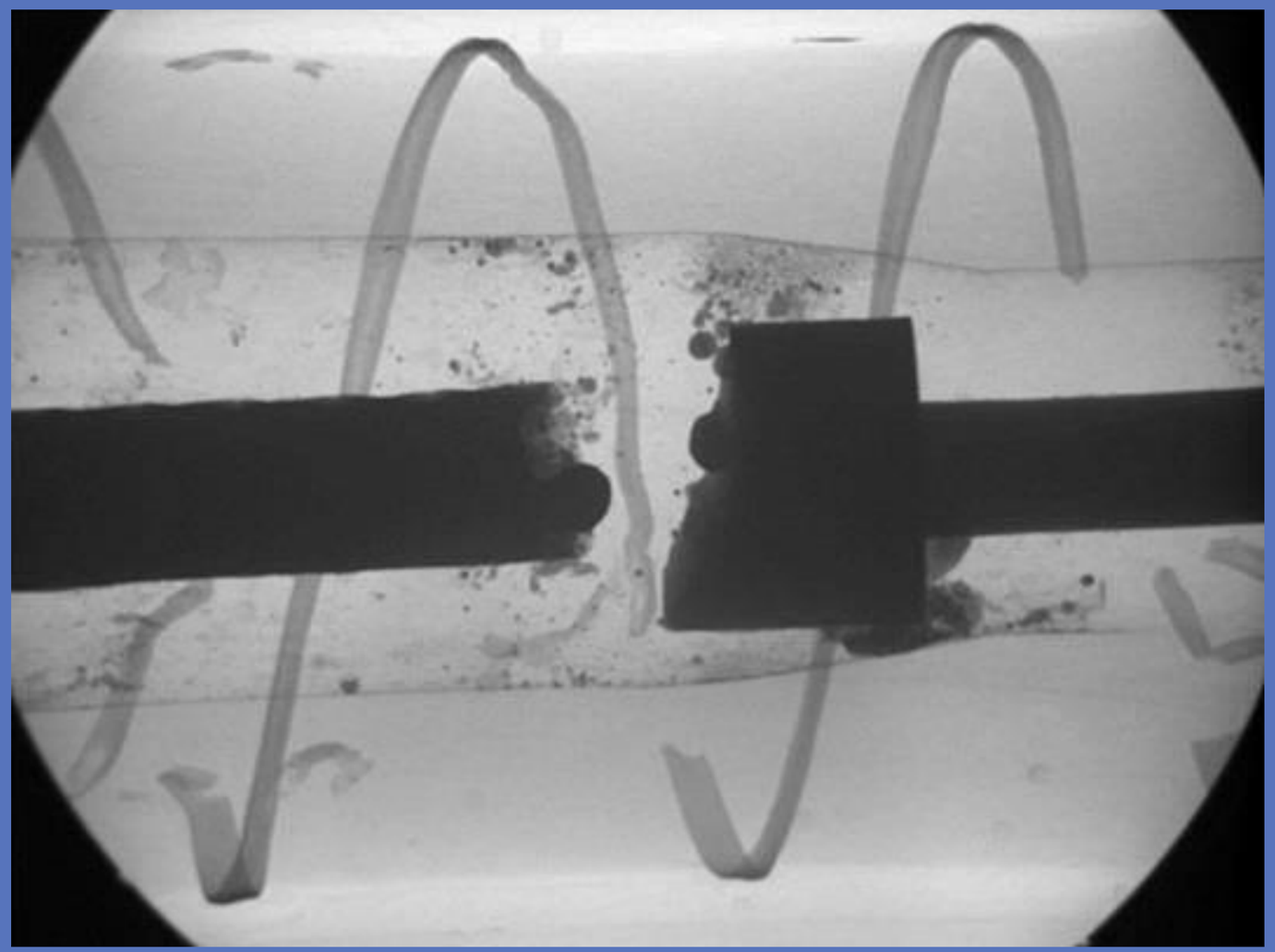
hidden layer	epochs	accuracy							time
		total	average	fuse_electrical_cutoff	fuse_fire_cutoff	fuse_not_cutoff	tempfuse_cutoff	tempfuse_not_cutoff	
128	30	79%	46%	54%	19%	78%	65%	13%	0:24:24
	100	77%	61%	67%	50%	72%	64%	52%	1:02:56
	200	87%	66%	68%	30%	89%	82%	59%	2:35:33
	300	85%	71%	81%	43%	83%	94%	52%	4:01:02
256	200	85%	59%	60%	23%	95%	85%	30%	2:31:52

- Epochs 300 is the best value
- The 256-layer model distinguishes clear things better than the 128-layer model
  (256레이어 모델은 128레이어 모델보다 명확한 것을 더 잘 구별해 냄)

Epochs 300

20	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0020.jpg	fuse_electrical_cutoff	fuse_not_cutoff	0%	0%	100%	0%	0%

- True value : Cartridge Fuse electrical cutoff
- Predict : Cartridge Fuse Not cutoff(100%)



Epochs 300

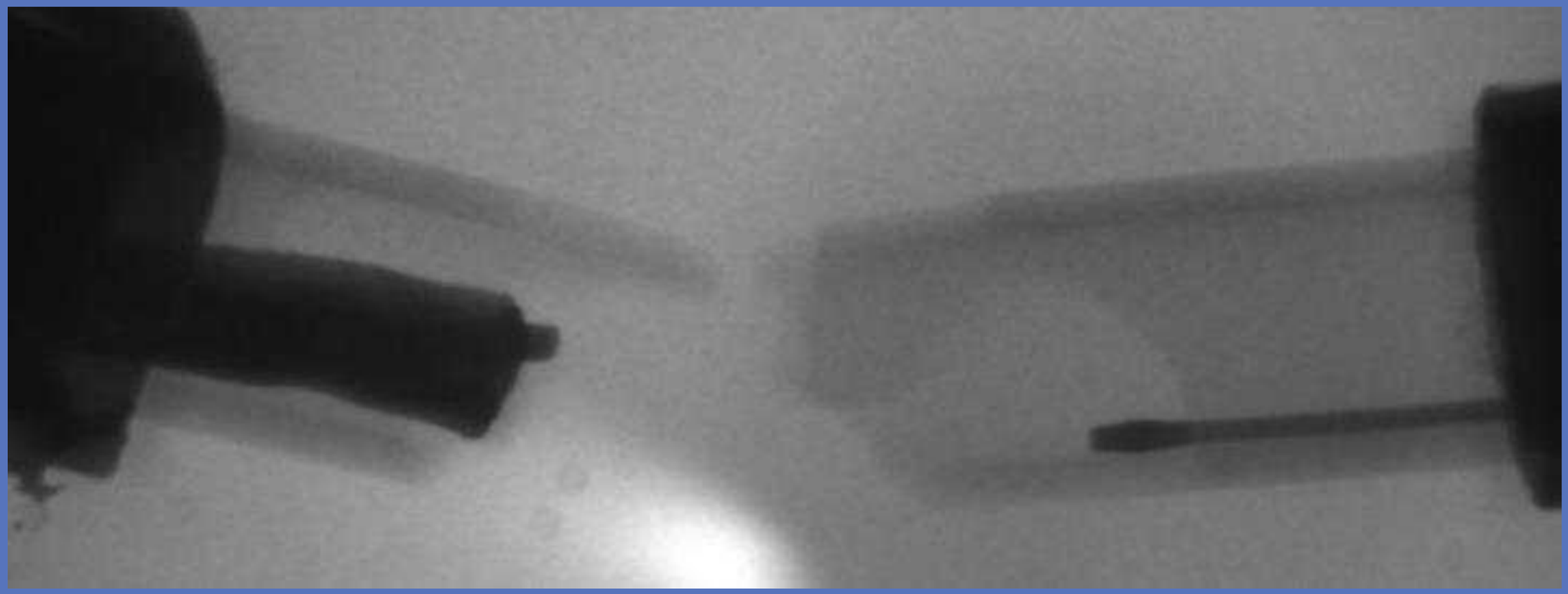
51	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0051.jpg	fuse_electrical_cutoff	fuse_not_cutoff	0%	0%	100%	0%	0%

- True value : Cartridge Fuse electrical cutoff
- Predict : Cartridge Fuse Not cutoff(100%)

Epochs 300

158	C:/pattern_recognition/fuse_status_detection/images₩fuse_fire_cutoff₩0008.jpg	fuse_fire_cutoff	fuse_electrical_cutoff	100%	0%	0%	0%	0%

- True value : Cartridge Fire cutoff
- Predict : Cartridge Fuse electrical cutoff(100%)



Epochs 300

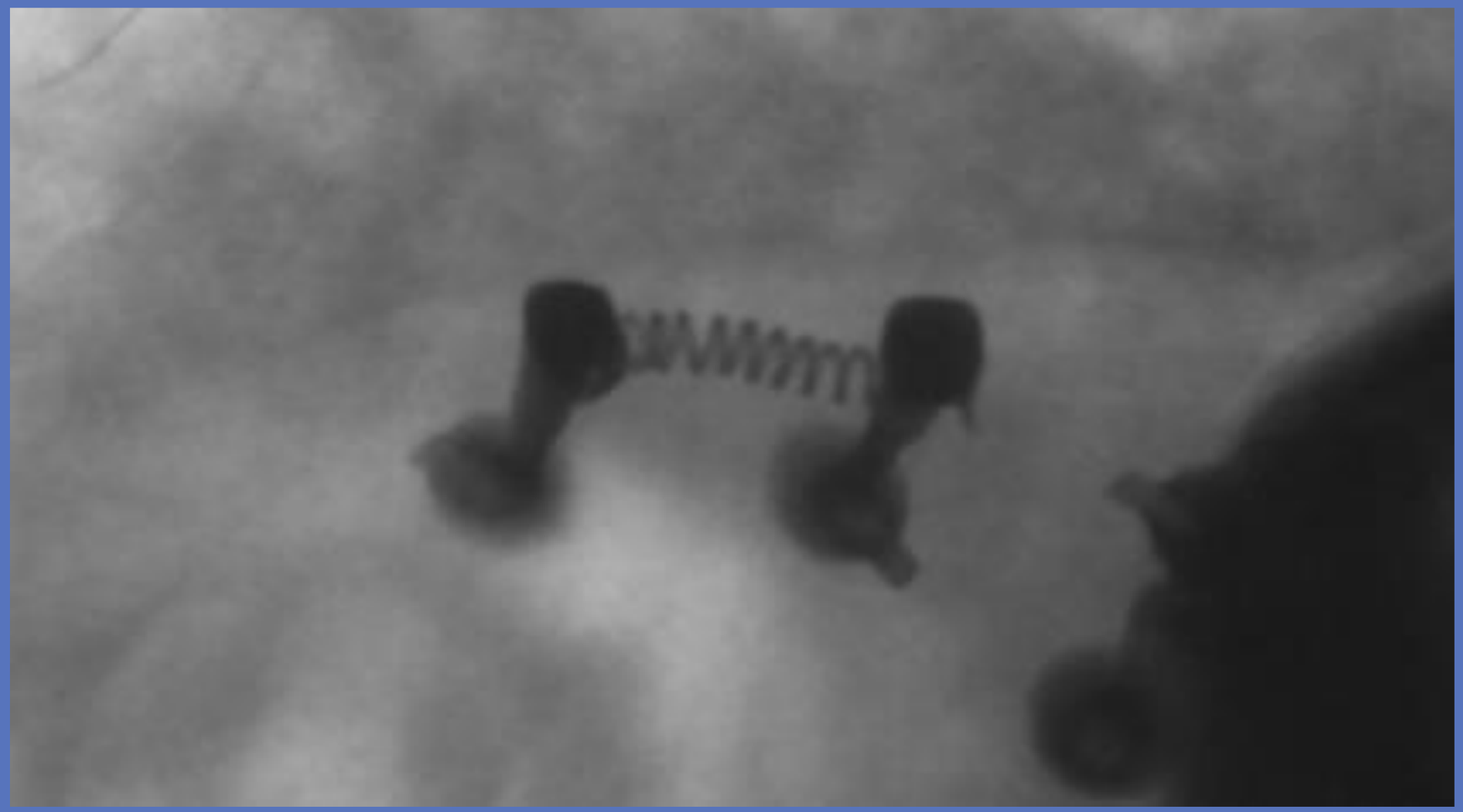
186	C:/pattern_recognition/fuse_status_detection/images₩fuse_fire_cutoff₩0036.jpg	fuse_fire_cutoff	fuse_not_cutoff	0%	0%	100%	0%	0%

- True value : Cartridge Fuse Fire cutoff
- Predict : Cartridge Fuse Not cutoff(100%)

Epochs 300

271	C:/pattern_recognition/fuse_status_detection/images₩fuse_not_cutoff₩0062.jpg	fuse_not_cutoff	fuse_electrical_cutoff	100%	0%	0%	0%	0%

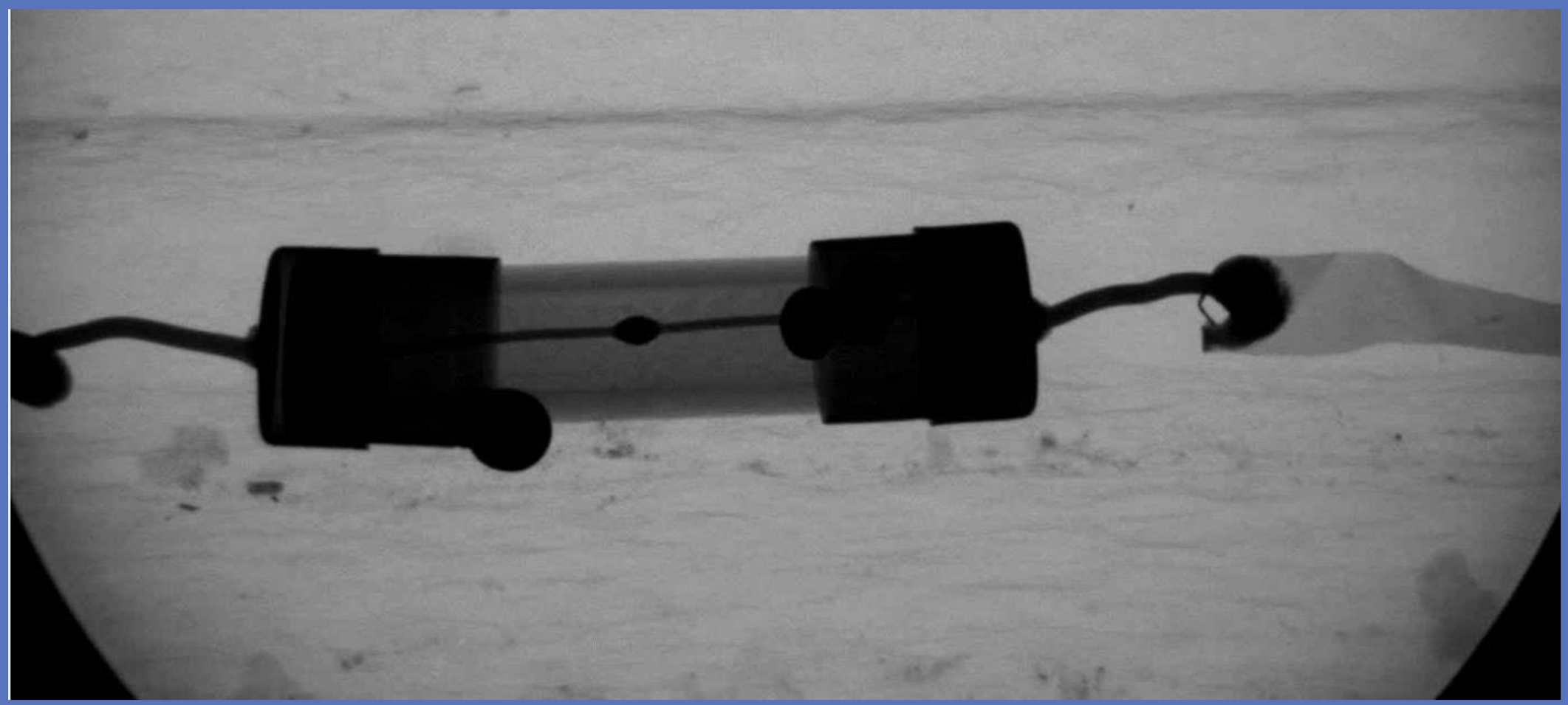
- True value : Cartridge Fuse Not cutoff
- Predict : Cartridge Fuse electrical cutoff(100%)



Epochs 300

599	C:/pattern_recognition/fuse_status_detection/images₩fuse_not_cutoff₩0390.jpg	fuse_not_cutoff	fuse_electrical_cutoff	100%	0%	0%	0%	0%

- True value : Cartridge Fuse Not cutoff
- Predict : Cartridge Fuse electrical cutoff(100%)



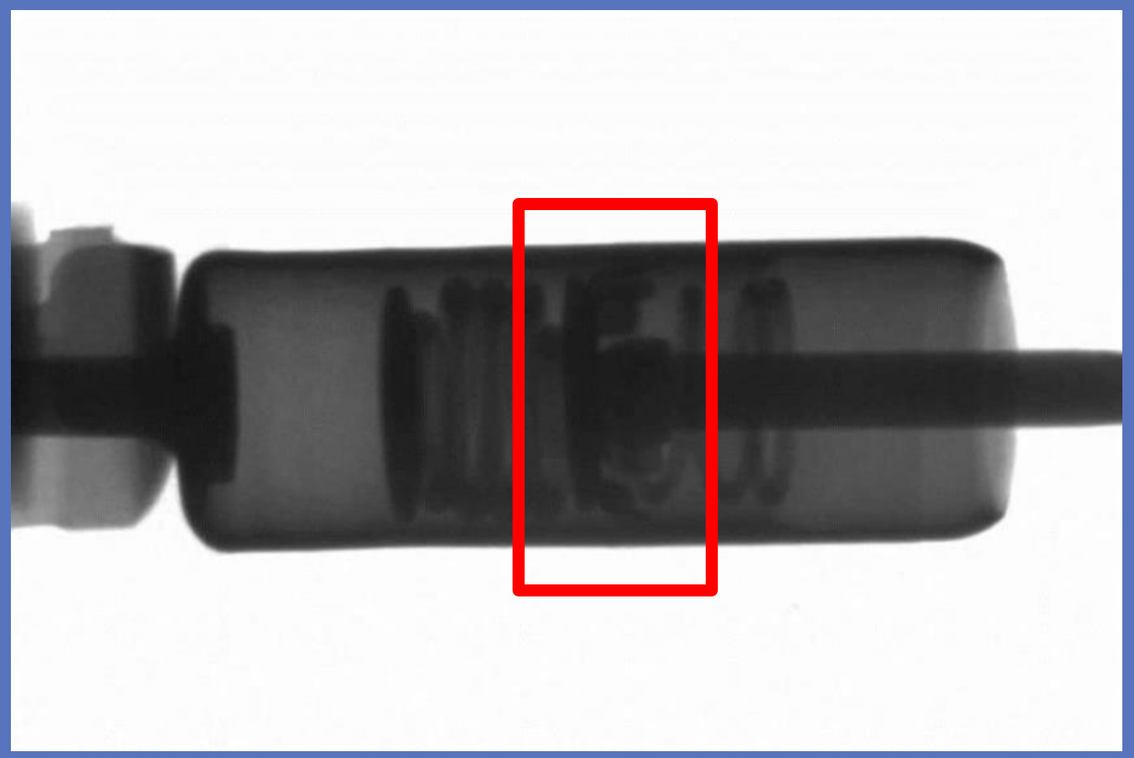
Epochs 300

795	C:/pattern_recognition/fuse_status_detection/images₩tempfuse_not_cutoff₩0002.jpg	tempfuse_not_cutoff	tempfuse_cutoff	6%	0%	1%	86%	7%

- True value : Temperature Fuse Not cutoff
- Predict : Temperature Fuse cutoff(86%)

  Temperature Fuse Not cutoff(7%)

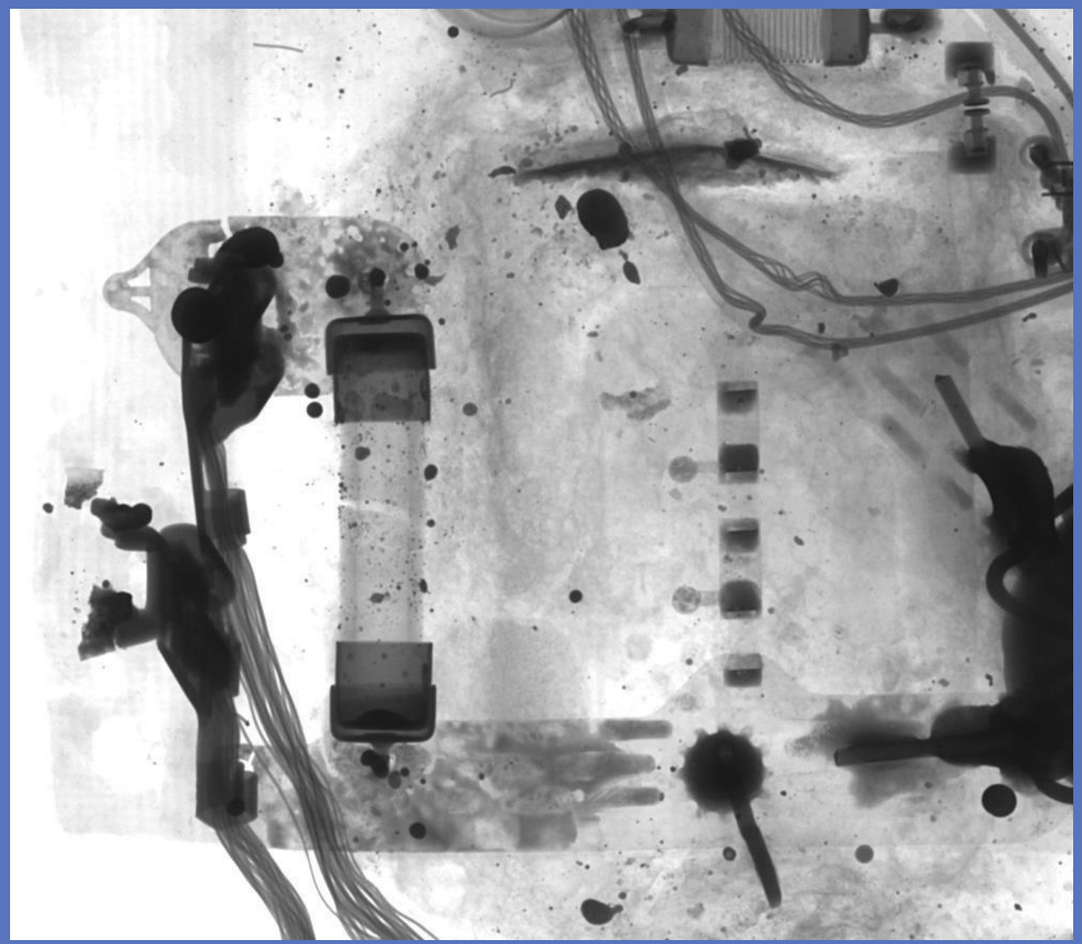
Cartridge Fuse electrical cutoff(6%)



04

Image set correction

Image correction



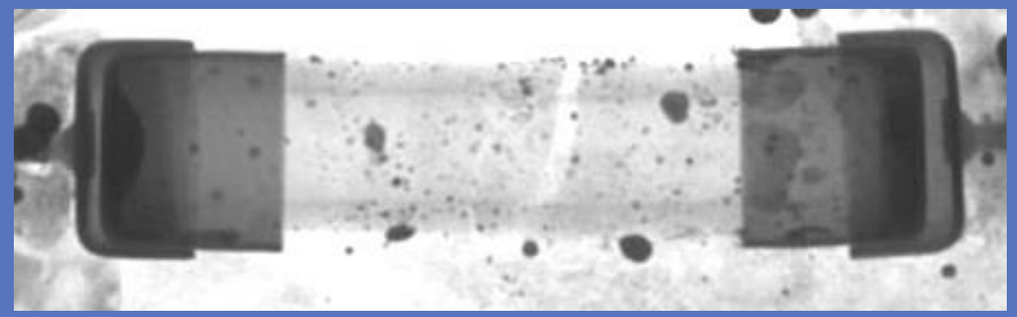
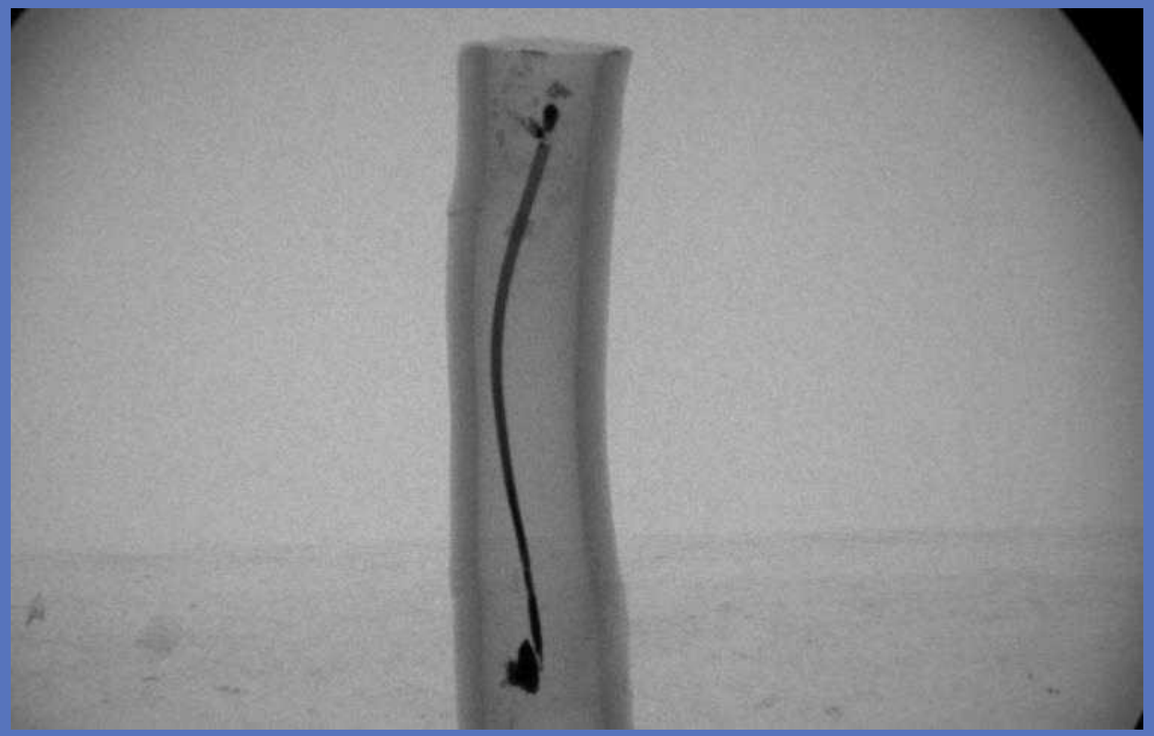


Image correction



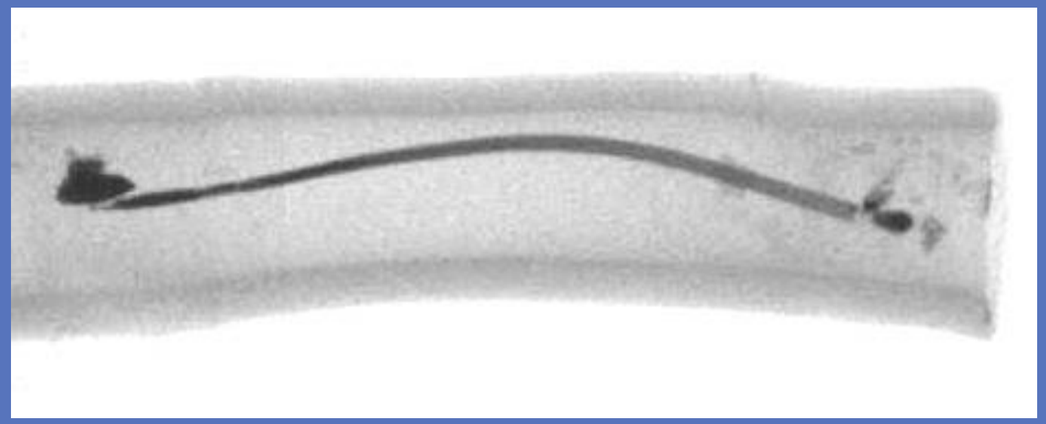
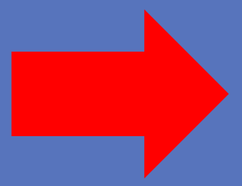


Image correction



result

hidden layer	epochs	accuracy							time
		total	average	fuse_electrical_cutoff	fuse_fire_cutoff	fuse_not_cutoff	tempfuse_cutoff	tempfuse_not_cutoff	
64	**300**	91%	75%	70%	75%	96%	98%	38%	4:10:42
128		93%	73%	86%	49%	96%	99%	36%	4:24:20
256		92%	80%	74%	79%	96%	89%	60%	3:37:42
512		94%	80%	78%	64%	98%	97%	62%	3:31:41

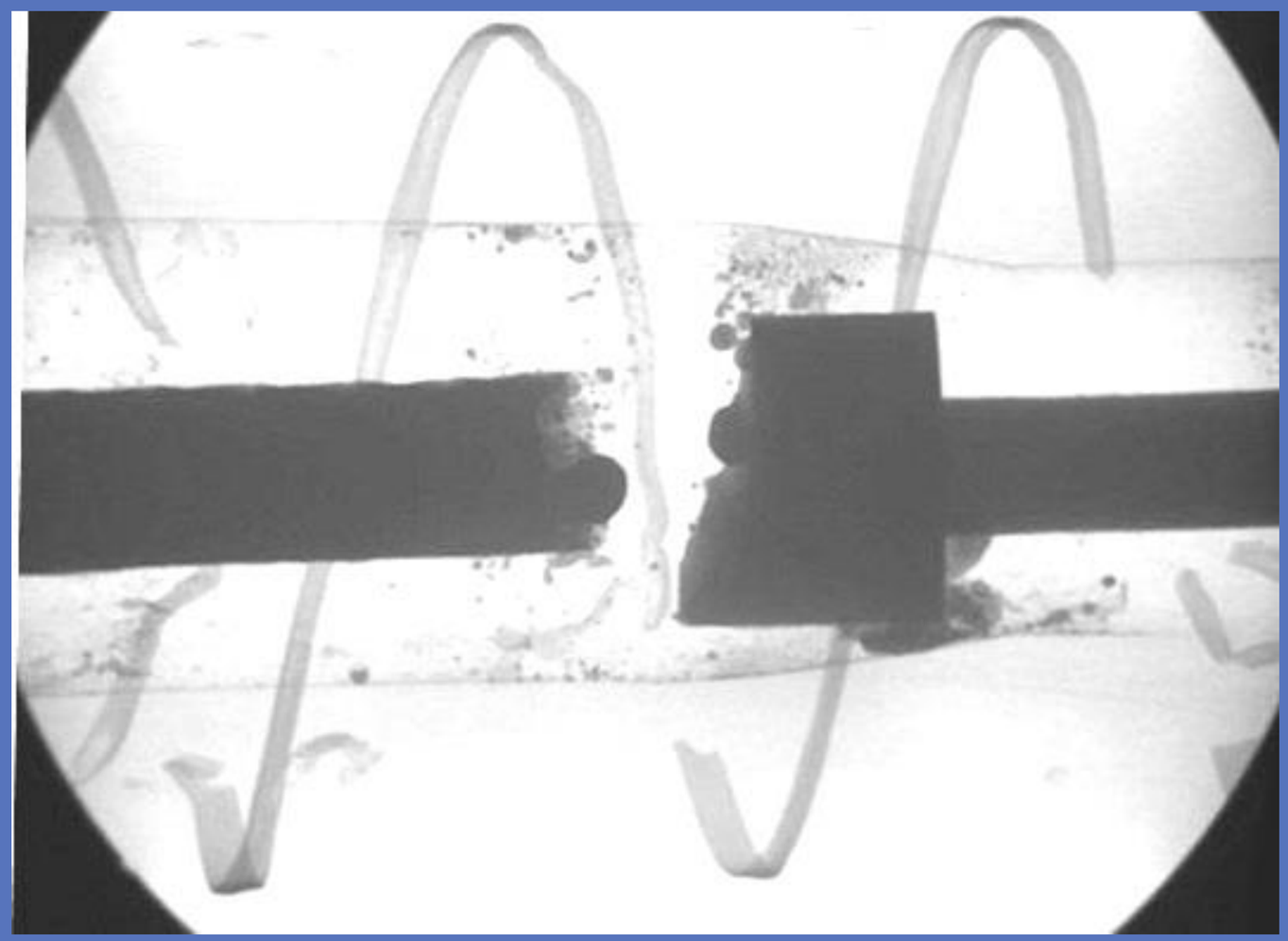
- The larger the layer, the better the distinction between clear things(레이어가 클 수록 명확한 것을 더 잘 구별함)

- A model that can clearly distinguish the obvious is a better model(명확한 것을 확실히 구별할 수 있는 모델이 더 좋은 모델임)

Hidden layer = 512

20	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0020.jpg	fuse_electrical_cutoff	fuse_not_cutoff	2%	2%	96%	0%	0%

- True value : Cartridge Fuse electrical cutoff
- Predict : Cartridge Fuse Not cutoff(96%)



Hidden layer = 512

51	C:/pattern_recognition/fuse_status_detection/images₩fuse_electrical_cutoff₩0051.jpg	fuse_electrical_cutoff	fuse_not_cutoff	0%	0%	100%	0%	0%

- True value : Cartridge Fuse electrical cutoff
- Predict : Cartridge Fuse Not cutoff(100%)

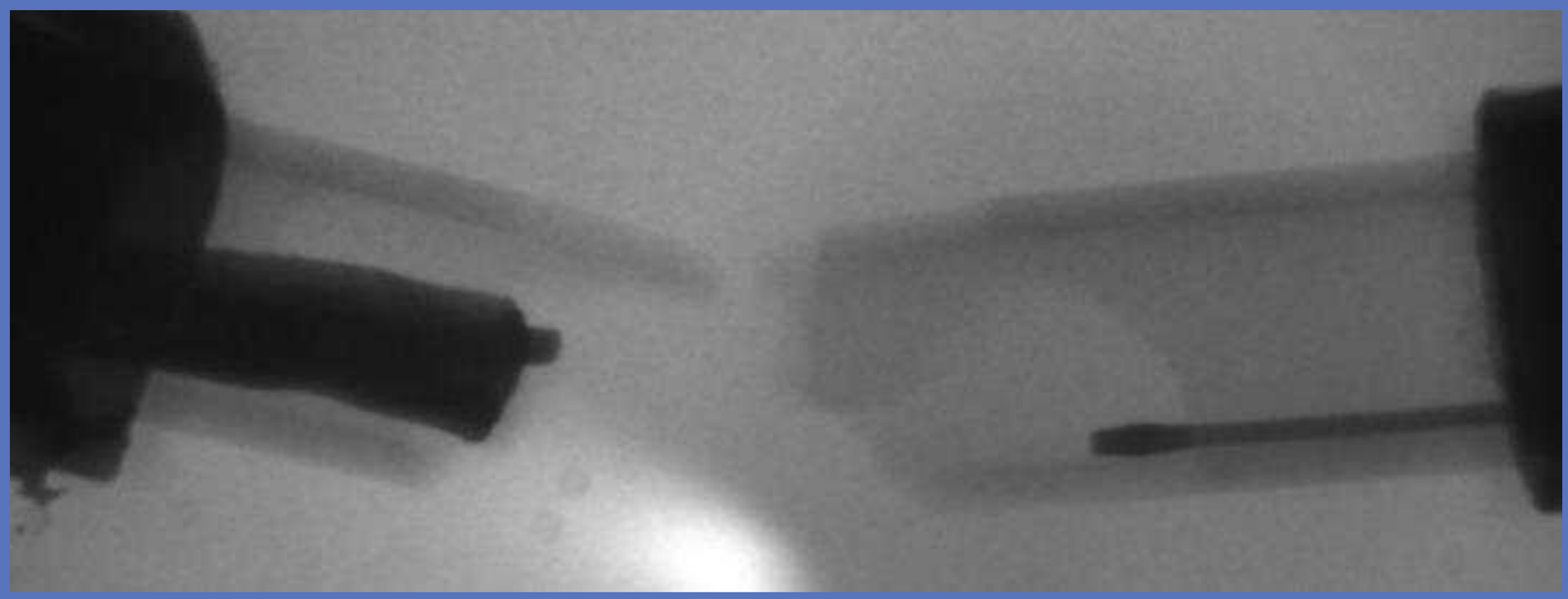
Hidden layer = 512

158	C:/pattern_recognition/fuse_status_detection/images₩fuse_fire_cutoff₩0008.jpg	fuse_fire_cutoff	fuse_not_cutoff	38%	8%	54%	0%	0%

- True value : Cartridge Fire cutoff
- Predict : Cartridge Fuse Not cutoff(54%)

  Cartridge Fuse Electrical cutoff(38%)

  Cartridge Fire cutoff(8%)



Hidden layer = 512

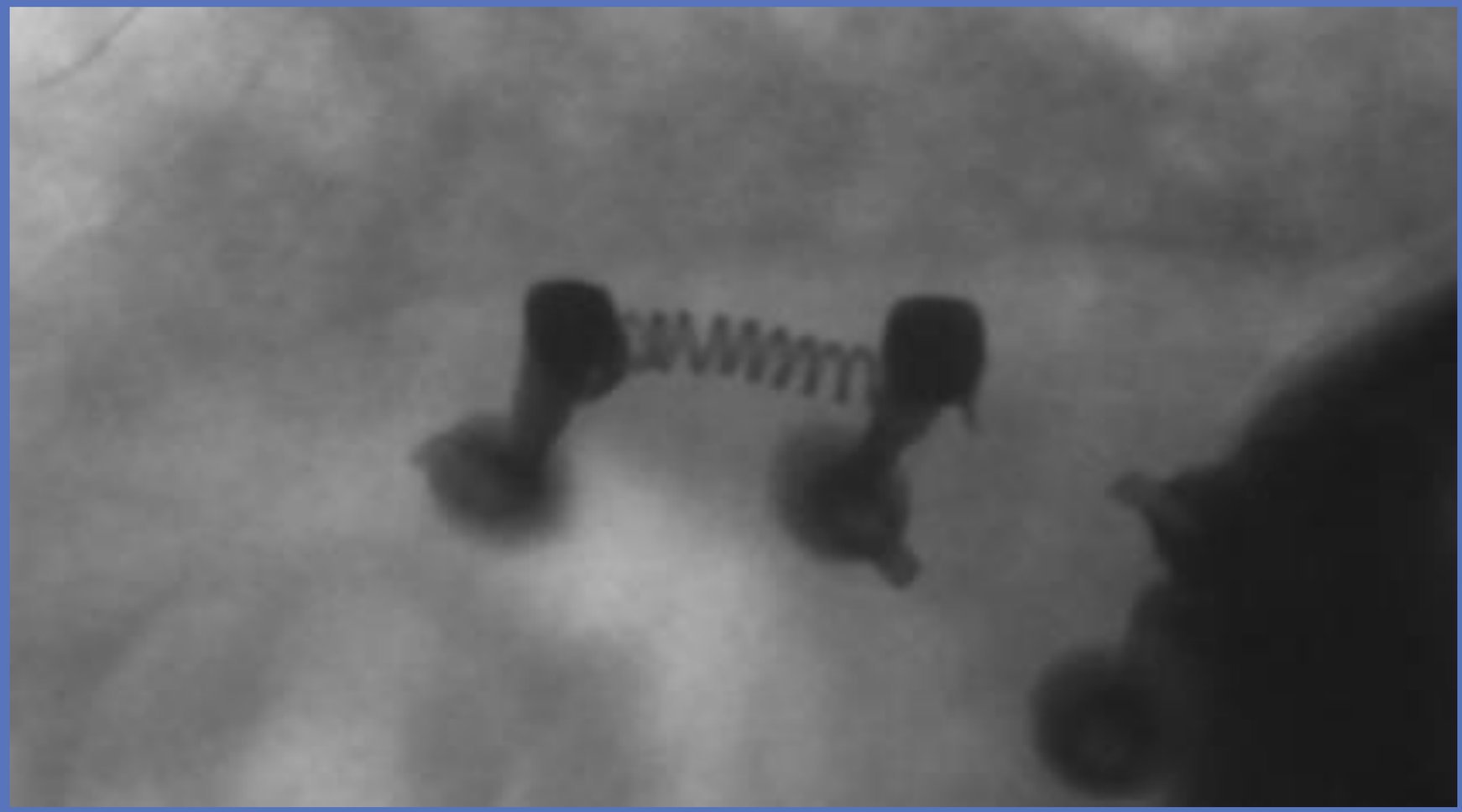
186	C:/pattern_recognition/fuse_status_detection/images₩fuse_fire_cutoff₩0036.jpg	fuse_fire_cutoff	fuse_not_cutoff	0%	0%	100%	0%	0%

- True value : Cartridge Fuse Fire cutoff
- Predict : Cartridge Fuse Not cutoff(100%)

Hidden layer = 512

271	C:/pattern_recognition/fuse_status_detection/images₩fuse_not_cutoff₩0062.jpg	fuse_not_cutoff	fuse_not_cutoff	0%	0%	100%	0%	0%

- True value : Cartridge Fuse Not cutoff
- Predict : Cartridge Fuse Not cutoff(100%)

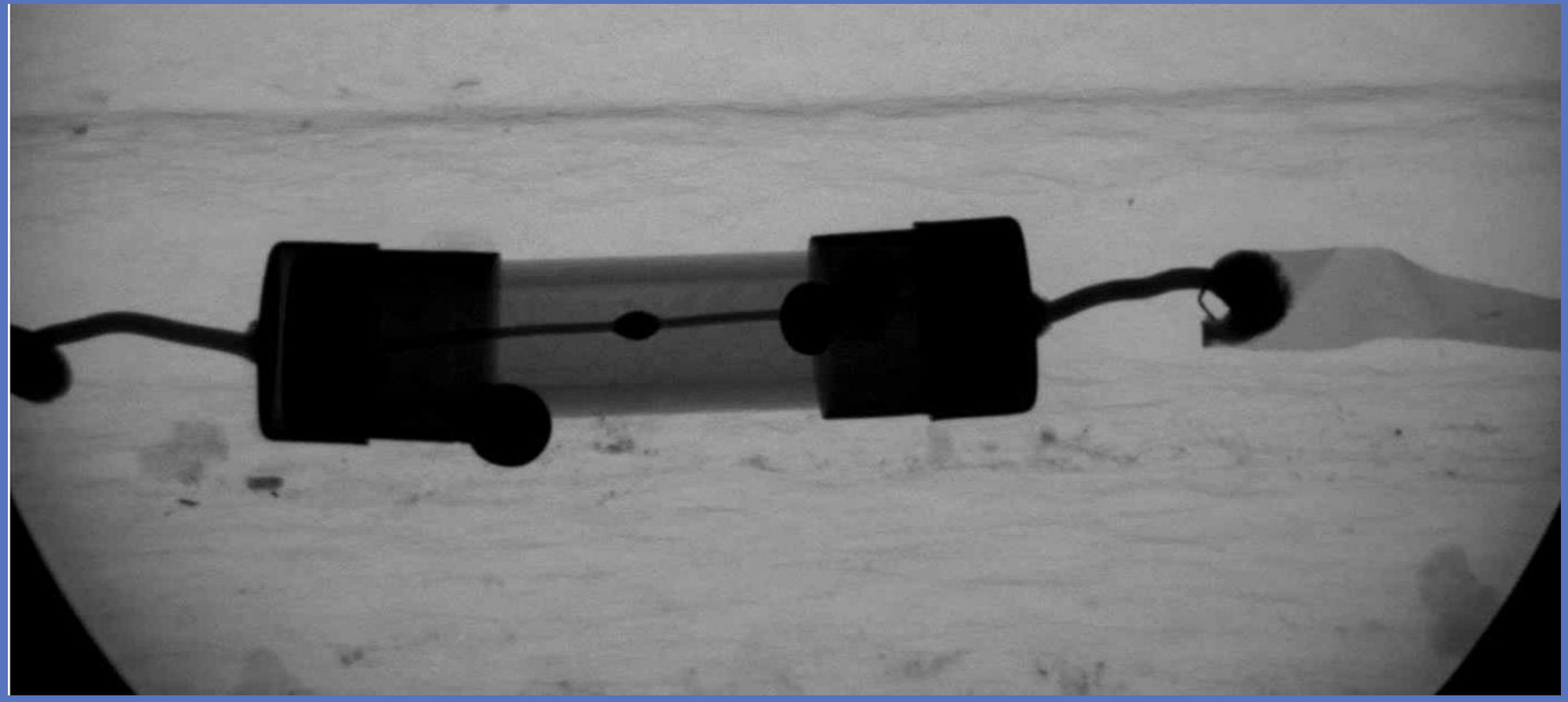


Hidden layer = 512

599	C:/pattern_recognition/fuse_status_detection/images₩fuse_not_cutoff₩0390.jpg	fuse_not_cutoff	fuse_not_cutoff	47%	0%	53%	0%	0%

- True value : Cartridge Fuse Not cutoff
- Predict : Cartridge Fuse Not cutoff(53%)

  Cartridge Fuse Electrical cutoff(47%)

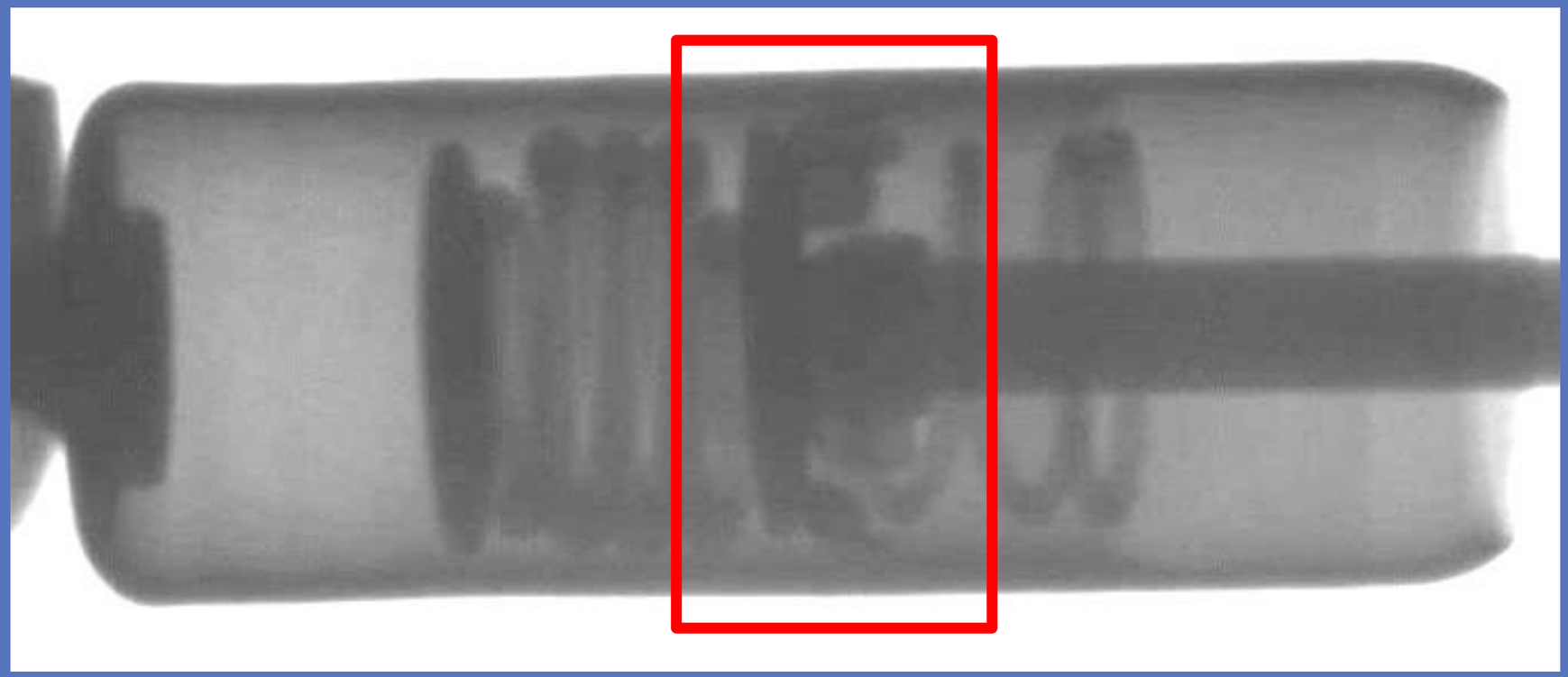


Hidden layer = 512

795	C:/pattern_recognition/fuse_status_detection/images₩tempfuse_not_cutoff₩0002.jpg	tempfuse_not_cutoff	tempfuse_cutoff	0%	0%	0%	99%	1%

- True value : Temperature Fuse Not cutoff
- Predict : Temperature Fuse cutoff(99%)

  Temperature Fuse cutoff(1%)



Summary

- Maximum accuracy recorded up to 94%
  (최대 정확도는 94%까지 기록하였음)
- Actual accuracy is 80% (실제 정확도는 80%임)
- Some images are completely wrong
  (몇몇 사진은 완전히 잘못 판단하는 경우가 있음)
- In some cases too few images
  (Cartridge Fuse(Not Cutoff)를 제외한 나머지는 이미지 개수가 많이 적음 → 낮은 정확도 도출)

	fuse_electrical_cutoff	fuse_fire_cutoff	fuse_not_cutoff	tempfuse_cutoff	tempfuse_not_cutoff
number of images	150	59	571	13	3

Summary

- Higher accuracy can be expected by increasing the number of images and increasing the epochs

(이미지 수를 증가시키고 epochs를 증가시키면 더 높은 정확도를 기대할 수 있음)

- Reducing the batch size can result in higher accuracy

(Batch size 를 감소시키면 더 높은 정확도가 나올 수 있음)

- Higher accuracy can be achieved by training the image by processing it in different ways

(이미지를 여러가지 방법으로 처리하여 훈련시키면 더 높은 정확도가 나올 수 있음)

05

Making Windows Application using PYQT5

CODE

```python
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QImage, QPixmap, QPalette, QPainter
from PyQt5.QtPrintSupport import QPrintDialog, QPrinter
from PyQt5.QtWidgets import *
import cv2
from tensorflow.keras.models import load_model
import numpy as np
model = load_model('C:/pattern_recognition/fuse_status_detection/model.h5')
image_resize_width = 500
image_resize_height = 300

class QImageViewer(QMainWindow):
    def __init__(self):
        super().__init__()

        self.printer = QPrinter()
        self.scaleFactor = 0.0

        self.imageLabel = QLabel()
        self.imageLabel.setBackgroundRole(QPalette.Base)
        self.imageLabel.setSizePolicy(QSizePolicy.Ignored, QSizePolicy.Ignored)
        self.imageLabel.setScaledContents(True)

        self.scrollArea = QScrollArea()
        self.scrollArea.setBackgroundRole(QPalette.Dark)
        self.scrollArea.setWidget(self.imageLabel)
        self.scrollArea.setVisible(False)

        self.setCentralWidget(self.scrollArea)

        dock = QDockWidget("Result", self)
        dock.setAllowedAreas(Qt.LeftDockWidgetArea |
                             Qt.RightDockWidgetArea |
                             Qt.BottomDockWidgetArea)
        self.result = QListWidget(dock)
        dock.setWidget(self.result)
        self.addDockWidget(Qt.BottomDockWidgetArea, dock)

        self.createActions()
        self.createMenus()

        self.setWindowTitle("Fuse Status Detector")
        self.resize(1200, 1000)
```

```python
    def open(self):
        options = QFileDialog.Options()
        # fileName = QFileDialog.getOpenFileName(self, "Open File", QDir.currentPath())
        fileName, _ = QFileDialog.getOpenFileName(self, 'QFileDialog.getOpenFileName()', '',
                                                  'Images (*.png *.jpeg *.jpg *.bmp *.gif)', options=options)
        if fileName:
            image = QImage(fileName)
            if image.isNull():
                QMessageBox.information(self, "Image Viewer", "Cannot load %s." % fileName)
                return

            self.imageLabel.setPixmap(QPixmap.fromImage(image))
            self.scaleFactor = 1.0
            self.scrollArea.setVisible(True)
            self.printAct.setEnabled(True)
            self.fitToWindowAct.setEnabled(True)
            self.updateActions()
            #------------------------------------------------
            image2 = cv2.imread(fileName)
            image2 = cv2.resize(image2, dsize=(image_resize_width, image_resize_height), interpolation=cv2.INTER_LINEAR)
            image3 = image2.reshape(1, image_resize_height, image_resize_width, 3)
            y_pred = model.predict(image3)
            pred = np.argmax(y_pred, axis=1)
            if pred == 0:
                result1 = 'fuse_electrical_cutoff'
            if pred == 1:
                result1 = 'fuse_fire_cutoff'
            if pred == 2:
                result1 = 'fuse_not_cutoff'
            if pred == 3:
                result1 = 'tempfuse_cutoff'
            if pred == 4:
                result1 = 'tempfuse_not_cutoff'

            if not self.fitToWindowAct.isChecked():
                self.imageLabel.adjustSize()

            self.result.addItem('fuse_electrical_cutoff =' + str(round(y_pred[0, 0] * 100, 1)) + '%')
            self.result.addItem('fuse_fire_cutoff =' + str(round(y_pred[0, 1] * 100, 1)) + '%')
            self.result.addItem('fuse_not_cutoff =' + str(round(y_pred[0, 2] * 100, 1)) + '%')
            self.result.addItem('tempfuse_cutoff =' + str(round(y_pred[0, 3] * 100, 1)) + '%')
            self.result.addItem('tempfuse_not_cutoff =' + str(round(y_pred[0, 4] * 100, 1)) + '%')
            self.result.addItem('result =' + str(result1))
            if max(y_pred[0]) <= 0.9:
                self.result.addItem('예측치가 낮습니다. 사진 수정 등이 필요합니다.')
            self.result.scrollToBottom()
```

CODE

```python
    def print_(self):
        dialog = QPrintDialog(self.printer, self)
        if dialog.exec_():
            painter = QPainter(self.printer)
            rect = painter.viewport()
            size = self.imageLabel.pixmap().size()
            size.scale(rect.size(), Qt.KeepAspectRatio)
            painter.setViewport(rect.x(), rect.y(), size.width(), size.height())
            painter.setWindow(self.imageLabel.pixmap().rect())
            painter.drawPixmap(0, 0, self.imageLabel.pixmap())

    def zoomIn(self):
        self.scaleImage(1.25)

    def zoomOut(self):
        self.scaleImage(0.8)

    def normalSize(self):
        self.imageLabel.adjustSize()
        self.scaleFactor = 1.0

    def fitToWindow(self):
        fitToWindow = self.fitToWindowAct.isChecked()
        self.scrollArea.setWidgetResizable(fitToWindow)
        if not fitToWindow:
            self.normalSize()

        self.updateActions()

    def about(self):
        QMessageBox.about(self, "About Image Viewer")

    def createActions(self):
        self.openAct = QAction("&Open...", self, shortcut="Ctrl+O", triggered=self.open)
        self.printAct = QAction("&Print...", self, shortcut="Ctrl+P", enabled=False, triggered=self.print_)
        self.exitAct = QAction("E&xit", self, shortcut="Ctrl+Q", triggered=self.close)
        self.zoomInAct = QAction("Zoom &In (25%)", self, shortcut="Ctrl++", enabled=False, triggered=self.zoomIn)
        self.zoomOutAct = QAction("Zoom &Out (25%)", self, shortcut="Ctrl+-", enabled=False, triggered=self.zoomOut)
        self.normalSizeAct = QAction("&Normal Size", self, shortcut="Ctrl+S", enabled=False, triggered=self.normalSize)
        self.fitToWindowAct = QAction("&Fit to Window", self, enabled=False, checkable=True, shortcut="Ctrl+F",
                                      triggered=self.fitToWindow)
        self.aboutAct = QAction("&About", self, triggered=self.about)
        self.aboutQtAct = QAction("About &Qt", self, triggered=qApp.aboutQt)
```

```python
    def createMenus(self):
        self.fileMenu = QMenu("&File", self)
        self.fileMenu.addAction(self.openAct)
        self.fileMenu.addAction(self.printAct)
        self.fileMenu.addSeparator()
        self.fileMenu.addAction(self.exitAct)

        self.viewMenu = QMenu("&View", self)
        self.viewMenu.addAction(self.zoomInAct)
        self.viewMenu.addAction(self.zoomOutAct)
        self.viewMenu.addAction(self.normalSizeAct)
        self.viewMenu.addSeparator()
        self.viewMenu.addAction(self.fitToWindowAct)

        self.helpMenu = QMenu("&Help", self)
        self.helpMenu.addAction(self.aboutAct)
        self.helpMenu.addAction(self.aboutQtAct)

        self.menuBar().addMenu(self.fileMenu)
        self.menuBar().addMenu(self.viewMenu)
        self.menuBar().addMenu(self.helpMenu)

    def updateActions(self):
        self.zoomInAct.setEnabled(not self.fitToWindowAct.isChecked())
        self.zoomOutAct.setEnabled(not self.fitToWindowAct.isChecked())
        self.normalSizeAct.setEnabled(not self.fitToWindowAct.isChecked())

    def scaleImage(self, factor):
        self.scaleFactor *= factor
        self.imageLabel.resize(self.scaleFactor * self.imageLabel.pixmap().size())

        self.adjustScrollBar(self.scrollArea.horizontalScrollBar(), factor)
        self.adjustScrollBar(self.scrollArea.verticalScrollBar(), factor)

        self.zoomInAct.setEnabled(self.scaleFactor < 3.0)
        self.zoomOutAct.setEnabled(self.scaleFactor > 0.333)

    def adjustScrollBar(self, scrollBar, factor):
        scrollBar.setValue(int(factor * scrollBar.value()
                               + ((factor - 1) * scrollBar.pageStep() / 2)))


if __name__ == '__main__':
    import sys
    from PyQt5.QtWidgets import QApplication

    app = QApplication(sys.argv)
    imageViewer = QImageViewer()
    imageViewer.show()
    sys.exit(app.exec_())
```

Core CODE

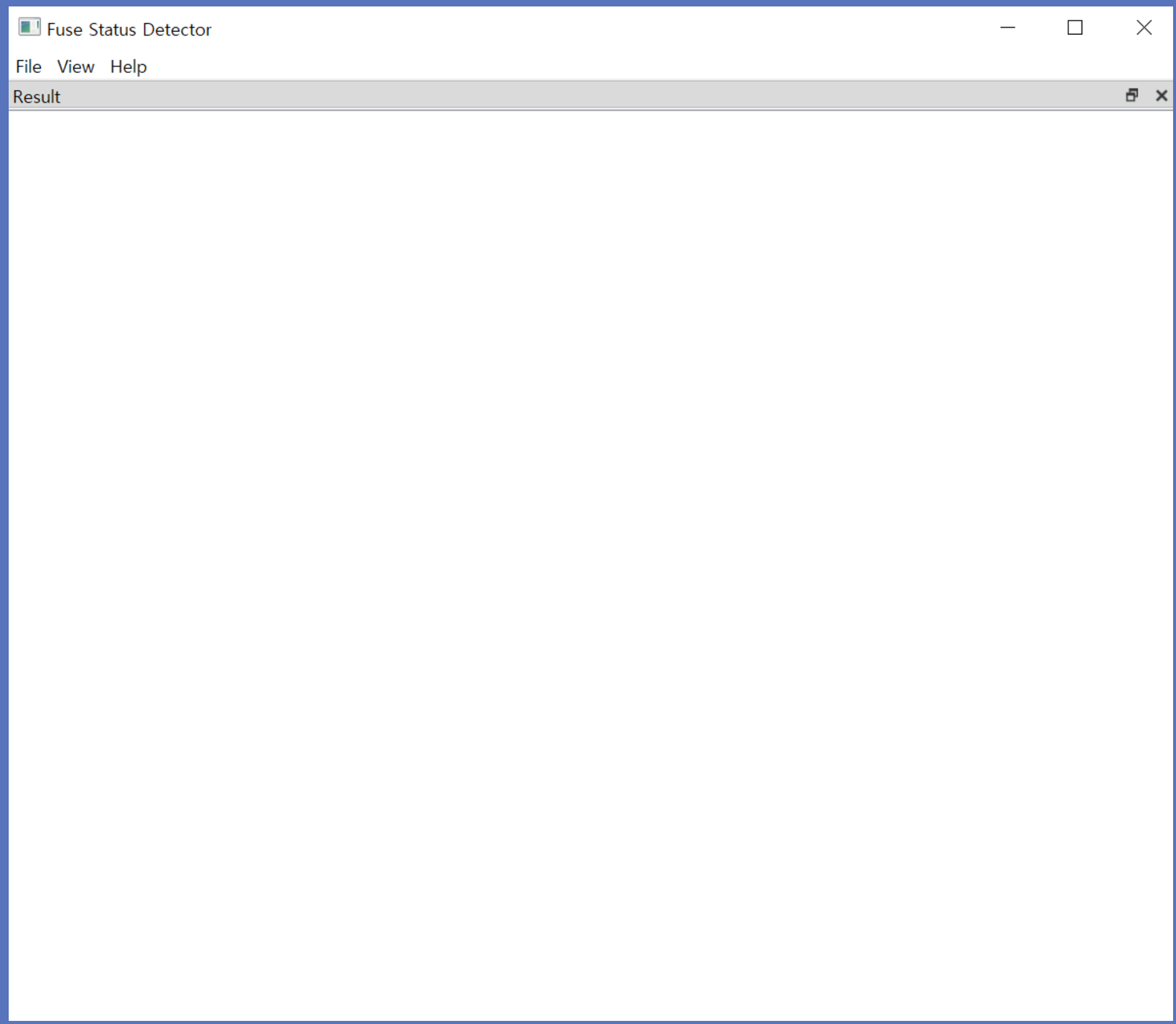
```
model = load_model('C:/pattern_recognition/fuse_status_detection/model.h5')
```

- Load trained model.

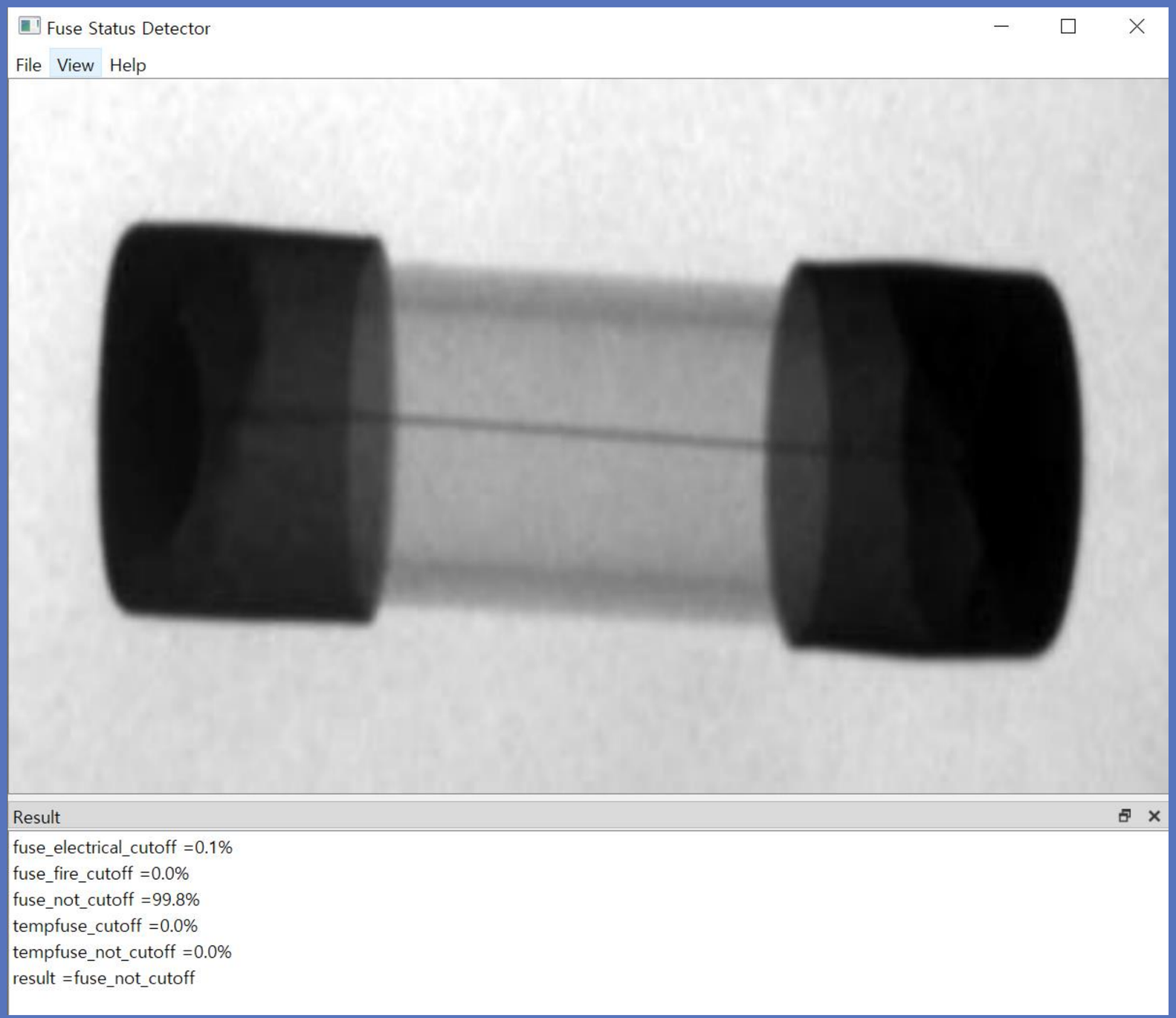
```
#------------------------------------------------
image2 = cv2.imread(fileName)
image2 = cv2.resize(image2, dsize=(image_resize_width, image_resize_height), interpolation=cv2.INTER_LINEAR)
image3 = image2.reshape(1, image_resize_height, image_resize_width, 3)
y_pred = model.predict(image3)
pred = np.argmax(y_pred, axis=1)
if pred == 0:
    result1 = 'fuse_electrical_cutoff'
if pred == 1:
    result1 = 'fuse_fire_cutoff'
if pred == 2:
    result1 = 'fuse_not_cutoff'
if pred == 3:
    result1 = 'tempfuse_cutoff'
if pred == 4:
    result1 = 'tempfuse_not_cutoff'

if not self.fitToWindowAct.isChecked():
    self.imageLabel.adjustSize()

self.result.addItem('fuse_electrical_cutoff =' + str(round(y_pred[0, 0] * 100, 1)) + '%')
self.result.addItem('fuse_fire_cutoff =' + str(round(y_pred[0, 1] * 100, 1)) + '%')
self.result.addItem('fuse_not_cutoff =' + str(round(y_pred[0, 2] * 100, 1)) + '%')
self.result.addItem('tempfuse_cutoff =' + str(round(y_pred[0, 3] * 100, 1)) + '%')
self.result.addItem('tempfuse_not_cutoff =' + str(round(y_pred[0, 4] * 100, 1)) + '%')
self.result.addItem('result =' + str(result1))
if max(y_pred[0]) <= 0.9:
    self.result.addItem('예측치가 낮습니다. 사진 수정 등이 필요합니다.')
self.result.scrollToBottom()
```

- Opened image predict and display
- If the predicted value is low, display "The predicted value is low"

Program



Program


Fuse Status Detector
File View Help
Result
fuse_electrical_cutoff =0.1%
fuse_fire_cutoff =0.0%
fuse_not_cutoff =99.8%
tempfuse_cutoff =0.0%
tempfuse_not_cutoff =0.0%
result =fuse_not_cutoff

Program


Fuse Status Detector
File View Help
Result
fuse_electrical_cutoff =37.8%
fuse_fire_cutoff =8.0%
fuse_not_cutoff =54.1%
tempfuse_cutoff =0.0%
tempfuse_not_cutoff =0.1%
result =fuse_not_cutoff
예측치가 낮습니다. 사진 수정 등이 필요합니다.

Thank you