Detection of Surface Crack using CNN

- 참조 노트북 : [https://www.kaggle.com/code/zeynel7/detection-of-surface-crack-using-cnn (https://www.kaggle.com/code/zeynel7/detection-of-surface-crack-using-cnn)](https://www.kaggle.com/code/zeynel7/detection-of-surface-crack-using-cnn)
- 데이터 셋 : [https://www.kaggle.com/datasets/arunrk7/surface-crack-detection (https://www.kaggle.com/datasets/arunrk7/surface-crack-detection)](https://www.kaggle.com/datasets/arunrk7/surface-crack-detection)
- 내용 : 콘크리트 표면 샘플 이미지의 손상을 확인.

학습 내용

- 폴더에 저장된 사진 이미지를 활용하여 이진 분류를 구현해 본다.
- CNN 모델을 구축해 본다.

목차

01 라이브러리 및 데이터 불러오기

목차로 이동하기

In [26]:

```
import matplotlib.pyplot as plt
import seaborn as sns

import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D , MaxPool2D , Flatten , Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam, RMSprop, Adagrad
from keras.layers import BatchNormalization
from sklearn.metrics import classification_report,confusion_matrix
import tensorflow as tf

import cv2
import os
import time
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [17]:

```
print(keras.__version__)
print(cv2.__version__)
print(sns.__version__)
print(np.__version__)
```

```
2.10.0
4.6.0
0.12.1
1.23.1
```

In [18]:

```
os.getcwd()
```

Out[18]:

```
'D:₩₩Github₩₩DeepLearning_Basic_Class₩₩00_part06_02_CrackDetection'
```

데이터 불러오기

In [19]:

```
labels = ['Negative', 'Positive']
img_size = 120

def read_images(data_dir):
    data = []
    for label in labels:
        path = os.path.join(data_dir, label)
        class_num = labels.index(label)
        for img in os.listdir(path):
            try:
                img_arr = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
                resized_arr = cv2.resize(img_arr, (img_size, img_size))    # 이미지 변경
                data.append([resized_arr, class_num])
            except Exception as e:
                print(e)
    return np.array(data)

Dataset = read_images('../datasets/Surface_Crack_Detection_small')
```

In [20]:

```
print( Dataset.shape )
print( Dataset[0][0], Dataset[0][1]) # 피처와 Target
```

```
(990, 2)
[[172 161 149 ... 183 182 184]
 [174 166 147 ... 176 175 177]
 [176 171 170 ... 180 176 176]
 ...
 [163 166 170 ... 175 175 173]
 [158 155 164 ... 172 173 171]
 [161 153 154 ... 170 172 170]] 0
```

폴더의 데이터 확인 - 데이터 시각화

In [61]:

```
# 폴더 내의 파일 수
data_dir = '../datasets/Surface_Crack_Detection_small'
path_negative = os.path.join(data_dir, "Negative")
path_positive = os.path.join(data_dir, "Positive")

print( len(os.listdir(path_negative) ))
print( len(os.listdir(path_positive) ))

num_n = len(os.listdir(path_negative) )
num_p = len(os.listdir(path_positive) )
num = [num_n, num_p]
num
```

```
490
500
```

Out[61]:

```
[490, 500]
```

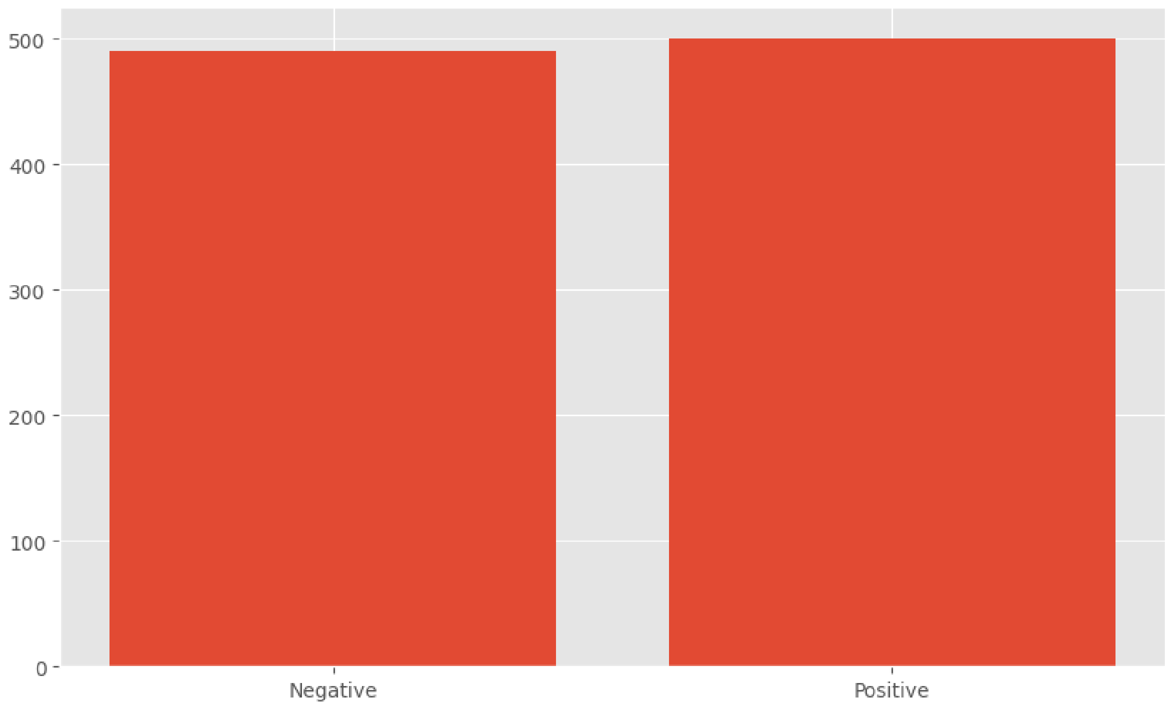
In [62]:

```
lm = ['Negative', 'Positive']
num = [num_n, num_p]

plt.figure(figsize=(10, 6))
x = np.arange(2)
plt.bar(lm, num)
```

Out[62]:

```
<BarContainer object of 2 artists>
```



02 데이터 전처리

목차로 이동하기

In [63]:

```
x = []
y = []

for feature, label in Dataset:
    x.append(feature)
    y.append(label)

x = np.array(x).reshape(-1, img_size, img_size, 1)
x = x / 255
y = np.array(y)
```

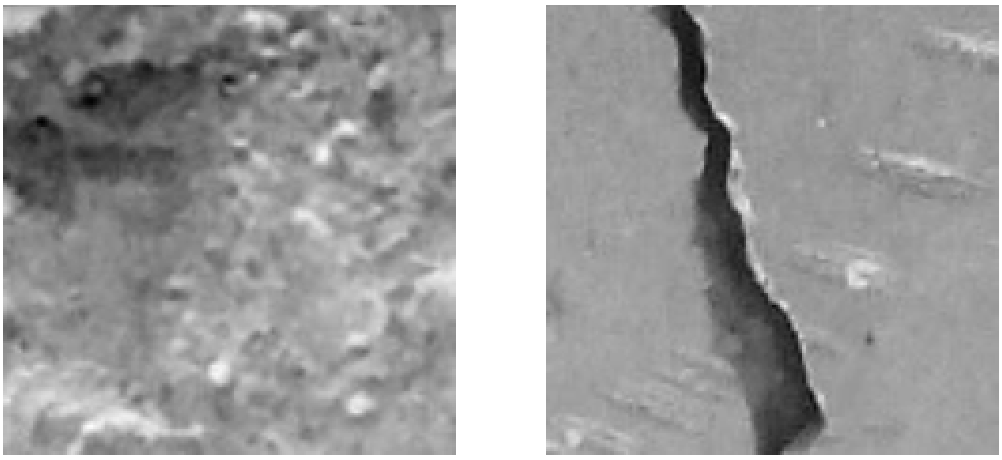
In [64]:

```
plt.subplot(1, 2, 1)
plt.imshow(x[300].reshape(img_size, img_size), cmap='gray')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(x[500].reshape(img_size, img_size), cmap='gray')
plt.axis('off')
```

Out[64]:

(-0.5, 119.5, 119.5, -0.5)



03 CNN 모델 구축

목차로 이동하기

In [74]:

```
x.shape[1:]
```

Out[74]:

(120, 120, 1)

In [75]:

```
model = Sequential()
model.add(Conv2D(64,3,padding="same", activation="relu", input_shape = x.shape[1:]))
model.add(MaxPool2D())

model.add(Conv2D(64, 3, padding="same", activation="relu"))
model.add(MaxPool2D())

model.add(Conv2D(128, 3, padding="same", activation="relu"))
model.add(MaxPool2D())

model.add(Flatten())
model.add(Dense(256,activation="relu"))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(1, activation="sigmoid"))

model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_15 (Conv2D)          (None, 120, 120, 64)      640

 max_pooling2d_15 (MaxPoolin  (None, 60, 60, 64)       0
 g2D)

 conv2d_16 (Conv2D)          (None, 60, 60, 64)        36928

 max_pooling2d_16 (MaxPoolin  (None, 30, 30, 64)       0
 g2D)

 conv2d_17 (Conv2D)          (None, 30, 30, 128)       73856

 max_pooling2d_17 (MaxPoolin  (None, 15, 15, 128)      0
 g2D)

 flatten_5 (Flatten)         (None, 28800)             0

 dense_10 (Dense)            (None, 256)               7373056

 dropout_5 (Dropout)         (None, 256)               0

 batch_normalization_5 (Batc  (None, 256)              1024
 hNormalization)

 dense_11 (Dense)            (None, 1)                 257

=================================================================
Total params: 7,485,761
Trainable params: 7,485,249
Non-trainable params: 512
_________________________________________________________________
```

모델 학습하기

In [67]:

```
start = time.time()

opt = Adam(lr=1e-5)
model.compile(loss="binary_crossentropy",
              optimizer=opt, metrics=["accuracy"])
history = model.fit(x, y, epochs = 15,
                    batch_size = 128, validation_split = 0.25, verbose=1)

print("소요시간", time.time() - start )
```

```
Epoch 1/15
6/6 [==============================] - 14s 2s/step - loss: 0.6723 - accuracy: 0.6051
- val_loss: 0.8110 - val_accuracy: 0.0000e+00
Epoch 2/15
6/6 [==============================] - 13s 2s/step - loss: 0.5471 - accuracy: 0.7129
- val_loss: 0.8786 - val_accuracy: 0.0000e+00
Epoch 3/15
6/6 [==============================] - 13s 2s/step - loss: 0.5044 - accuracy: 0.7615
- val_loss: 0.9086 - val_accuracy: 0.0000e+00
Epoch 4/15
6/6 [==============================] - 13s 2s/step - loss: 0.5025 - accuracy: 0.7655
- val_loss: 0.9090 - val_accuracy: 0.0000e+00
Epoch 5/15
6/6 [==============================] - 13s 2s/step - loss: 0.4722 - accuracy: 0.7978
- val_loss: 0.8947 - val_accuracy: 0.0000e+00
Epoch 6/15
6/6 [==============================] - 13s 2s/step - loss: 0.4586 - accuracy: 0.7911
- val_loss: 0.8742 - val_accuracy: 0.0000e+00
Epoch 7/15
6/6 [==============================] - 13s 2s/step - loss: 0.4354 - accuracy: 0.8059
- val_loss: 0.8462 - val_accuracy: 0.0000e+00
Epoch 8/15
6/6 [==============================] - 13s 2s/step - loss: 0.4236 - accuracy: 0.8127
- val_loss: 0.8175 - val_accuracy: 0.0000e+00
Epoch 9/15
6/6 [==============================] - 13s 2s/step - loss: 0.4013 - accuracy: 0.8369
- val_loss: 0.7776 - val_accuracy: 0.0000e+00
Epoch 10/15
6/6 [==============================] - 13s 2s/step - loss: 0.3868 - accuracy: 0.8464
- val_loss: 0.7484 - val_accuracy: 0.0444
Epoch 11/15
6/6 [==============================] - 13s 2s/step - loss: 0.3750 - accuracy: 0.8544
- val_loss: 0.7214 - val_accuracy: 0.2540
Epoch 12/15
6/6 [==============================] - 12s 2s/step - loss: 0.3551 - accuracy: 0.8625
- val_loss: 0.6943 - val_accuracy: 0.5403
Epoch 13/15
6/6 [==============================] - 13s 2s/step - loss: 0.3403 - accuracy: 0.8733
- val_loss: 0.6664 - val_accuracy: 0.7540
Epoch 14/15
6/6 [==============================] - 13s 2s/step - loss: 0.3122 - accuracy: 0.8881
- val_loss: 0.6420 - val_accuracy: 0.8831
Epoch 15/15
6/6 [==============================] - 13s 2s/step - loss: 0.3019 - accuracy: 0.8922
- val_loss: 0.6226 - val_accuracy: 0.9516
소요시간 194.53148007392883
```

04 모델 결과 확인

목차로 이동하기

In [69]:

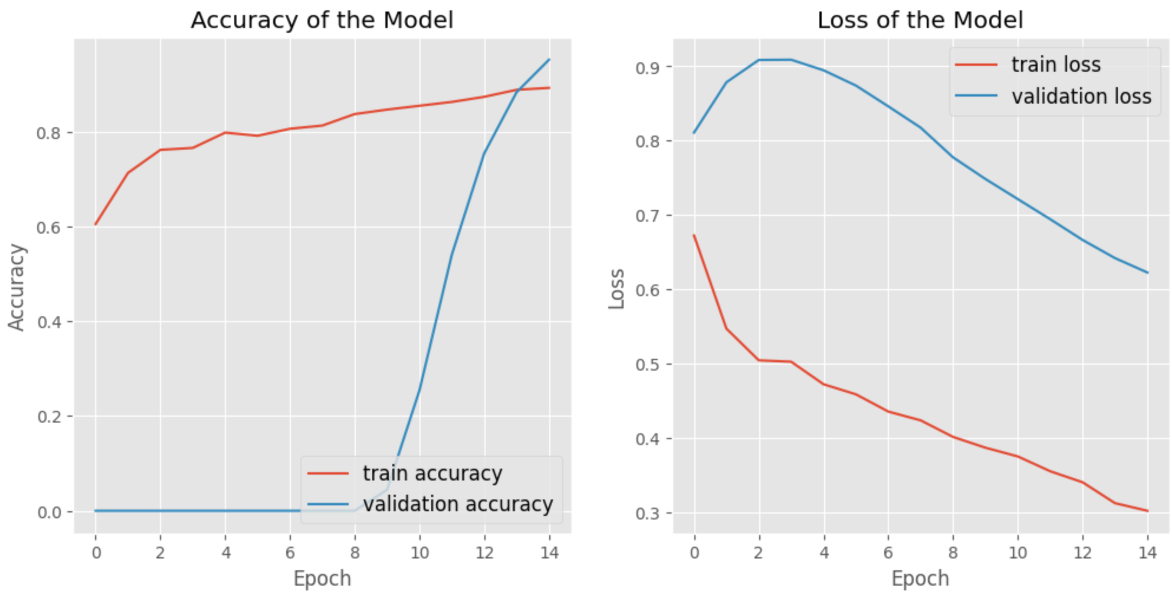
```
plt.figure(figsize=(12, 12))
plt.style.use('ggplot')

plt.subplot(2,2,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy of the Model')
plt.ylabel('Accuracy', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.legend(['train accuracy', 'validation accuracy'],
                     loc='lower right', prop={'size': 12})

plt.subplot(2,2,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss of the Model')
plt.ylabel('Loss', fontsize=12)
plt.xlabel('Epoch', fontsize=12)
plt.legend(['train loss', 'validation loss'],
                     loc='best', prop={'size': 12})
```

Out[69]:

```
<matplotlib.legend.Legend at 0x1c336b075e0>
```



Classification Report : 평가 검증 결과

In [70]:

```
x.shape, y.shape
```

Out[70]:

```
((990, 120, 120, 1), (990,))
```

In [71]:

```
from sklearn.metrics import classification_report,confusion_matrix
```

In [72]:

```
predictions = (model.predict(x) > 0.5).astype("int32")
print( predictions.shape )
```

```
31/31 [==============================] - 3s 106ms/step
(990, 1)
```

In [73]:

```
print(classification_report(y, predictions, target_names = ['Negative','Positive']))
```

```
              precision    recall  f1-score   support

    Negative       0.95      0.89      0.92       490
    Positive       0.90      0.96      0.92       500

    accuracy                           0.92       990
   macro avg       0.92      0.92      0.92       990
weighted avg       0.92      0.92      0.92       990
```