





# Concrete Crack Detection

## Authors

---

- **Yu-Sian Lin**  
·  [yslin0114](#)  
Department of Civil Engineering, University of Illinois
- **Yueh-Ti Lee**  
·  [Yueh-Ti](#)  
Department of Civil Engineering, University of Illinois
- **Minjiang Zhu**  
·  [Minjiang-Zhu](#)  
Department of Something, University of Illinois
- **Chengyou Yue**  
·  [TensorYue](#)  
Department of Civil Engineering, University of Illinois

## Description

---

In this project, we plan to use two main datasets to detect concrete crack. First, we use a dataset from Kaggle, titled "Surface Crack Detection" to make the classification of the concrete crack. Second, we use a dataset from Virginia Tech, titled "Concrete Crack Conglomerate Dataset" to make the segmentation of the concrete crack.

[Classification]

Source: Kaggle/ Surface Crack Detection.

Data Format: Images with 227\*227 pixels with RGB channels.

Content: The dataset contains images of various concrete surfaces with and without crack. The image data are divided into two as negative (without crack) and positive (with crack) for image classification. Each class has 20000 images with a total of 40000 images.



Figure 1. Example of Images with and without Cracks

[Segmentation]

Source: Virginia Tech/ Concrete Crack Conglomerate Dataset

Data Format: Images with 448\*448 pixels with RGB channels

Content: The dataset contains over 10,995 crack images. Each image data has its origin image and mask image for image segmentation.

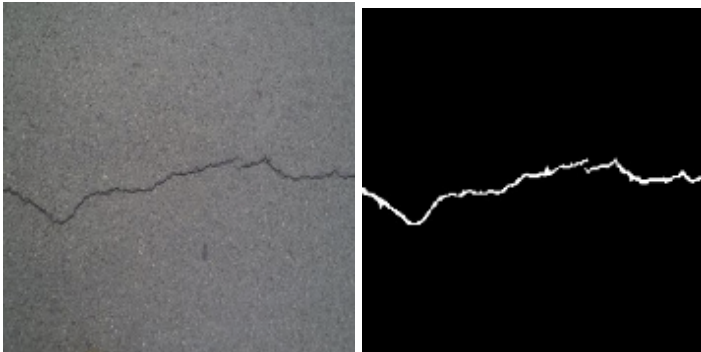


Figure 2. Example of an Original Image and an Mask Image

## Proposal:

---

In our project, we plan to train two models to detect concrete crack. First, by using the classification dataset, we plan to detect concrete crack based on convolutional neural network (CNN) to find out whether the concrete is defective. Second, by using the segmentation dataset, we plan to conduct concrete crack segmentation based on U Net to illustrate the position of the crack on concrete crack images.

## Exploratory Data Analysis

---

In classification, the dataset consists of 20,000 images with cracks and 20,000 images without cracks. We are planning to use convolutional neural network to train a classification model.

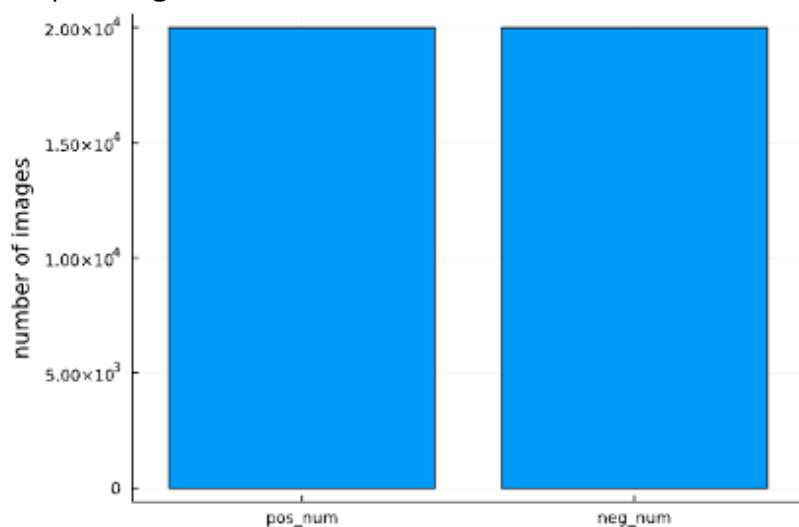


Figure 3. Number of Images with and without Cracks

In segmentation, we use the dataset consists of 21,996 images, splitting it into train data with 19,801 images and train data with 2,195 images. Here, we'll be applying U Net to train a model that is capable

of pointing out the cracks in a concrete image.

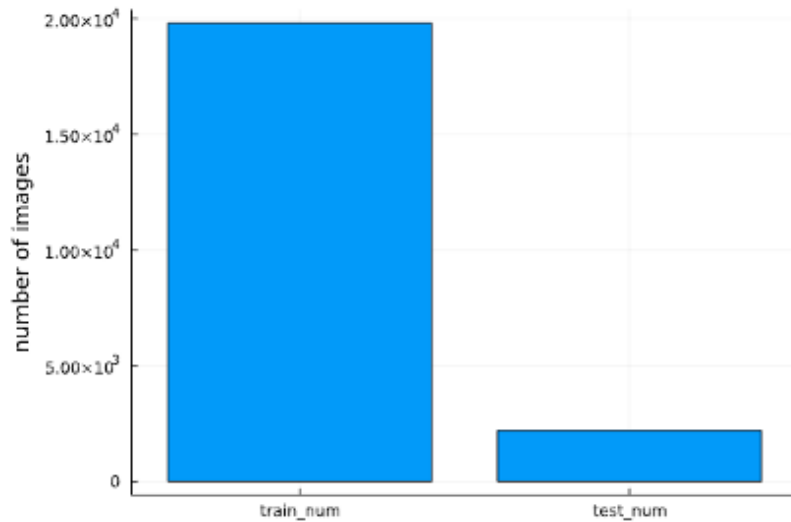


Figure 4. Number of Train Images and Test Images

## Predictive Modeling

In our group project, we trained two models: 1. Concrete crack detection based on convolutional neural network (CNN) 2. Concrete crack segmentation based on U Net.

[Crack detection] Dataset: Surface Crack Detection | Kaggle Positive: 20000 Negative: 20000 Training, validating, testing: 3360, 840, 1800 Network: CNN

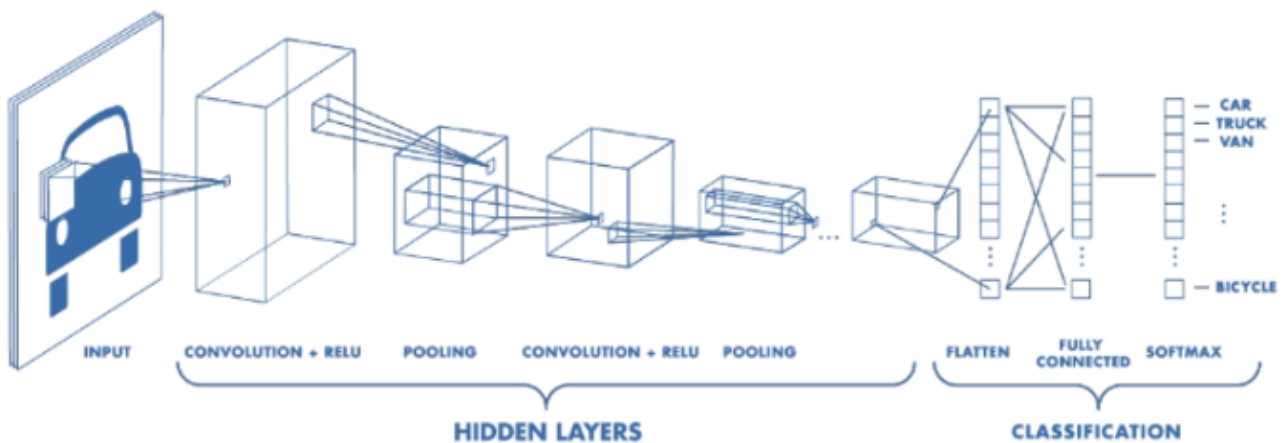


Figure 7

Figure 5. Schematic of CNN

```
inputs = tf.keras.Input(shape=(120,120,3))
x = tf.keras.layers.Conv2D(filters=16, kernel_size=(3,3), activation='relu')(inputs)
x = tf.keras.layers.MaxPool2D(pool_size=(2,2))(x)
x = tf.keras.layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu')(x)
x = tf.keras.layers.MaxPool2D(pool_size=(2,2))(x)

x = tf.keras.layers.GlobalAveragePooling2D()(x)
outputs = tf.keras.layers.Dense(1, activation='sigmoid')(x)
```

Figure 8

Figure 6. Code of CNN

Input: images with 120x120x3 [height, width, channel (RGB)] (resized from 227x227)

Output: some kind of probability (was proceed by sigmoid)

Solver: Adam

Loss: Binary Cross Entropy

Testing result: True Positive, False Positive, True Negative, False Negative.

Positive or negative was predicted by the network, and true or false is the result of prediction (right or wrong)

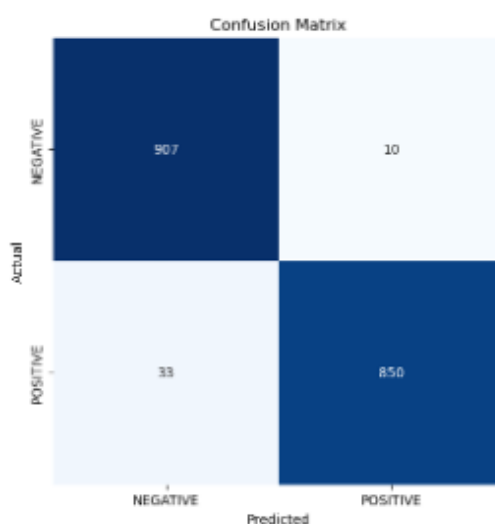


Figure 14

Figure 7. Confusion Matrix of the Test Result

[Crack Segmentation] Dataset: Concrete Crack Conglomerate Dataset (vt.edu)

Would only use a part of it (107 for training and validating, 11 for testing) as U-Net is an effective architecture.

Network: U-Net [1505.04597v1] U-Net: Convolutional Networks for Biomedical Image Segmentation (arxiv.org)

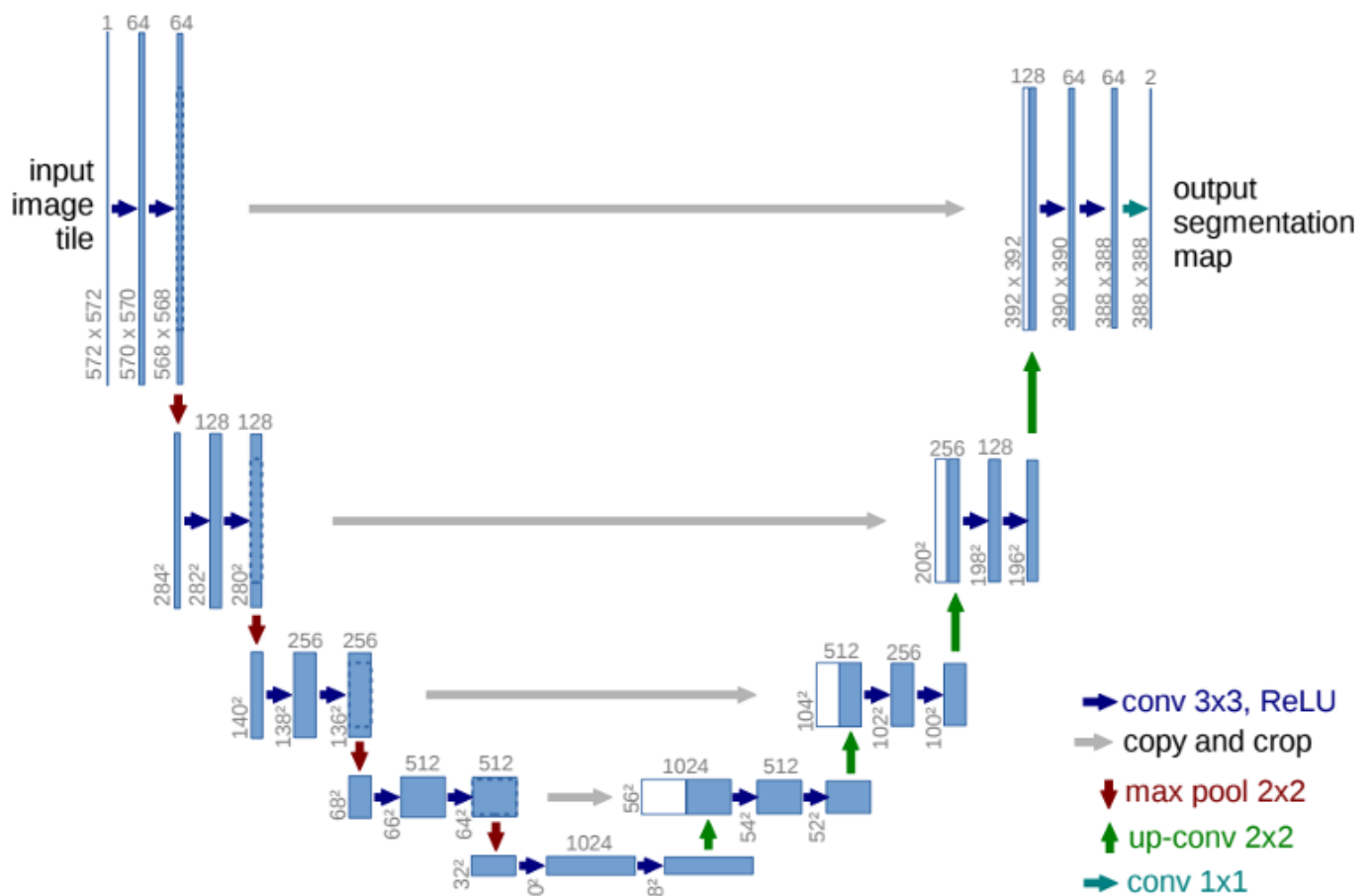


Figure 9

Figure 8. Schematic of a U Net

Here we would have input with 3 channel, and would become 16 channel after the first convolution.



Figure 10

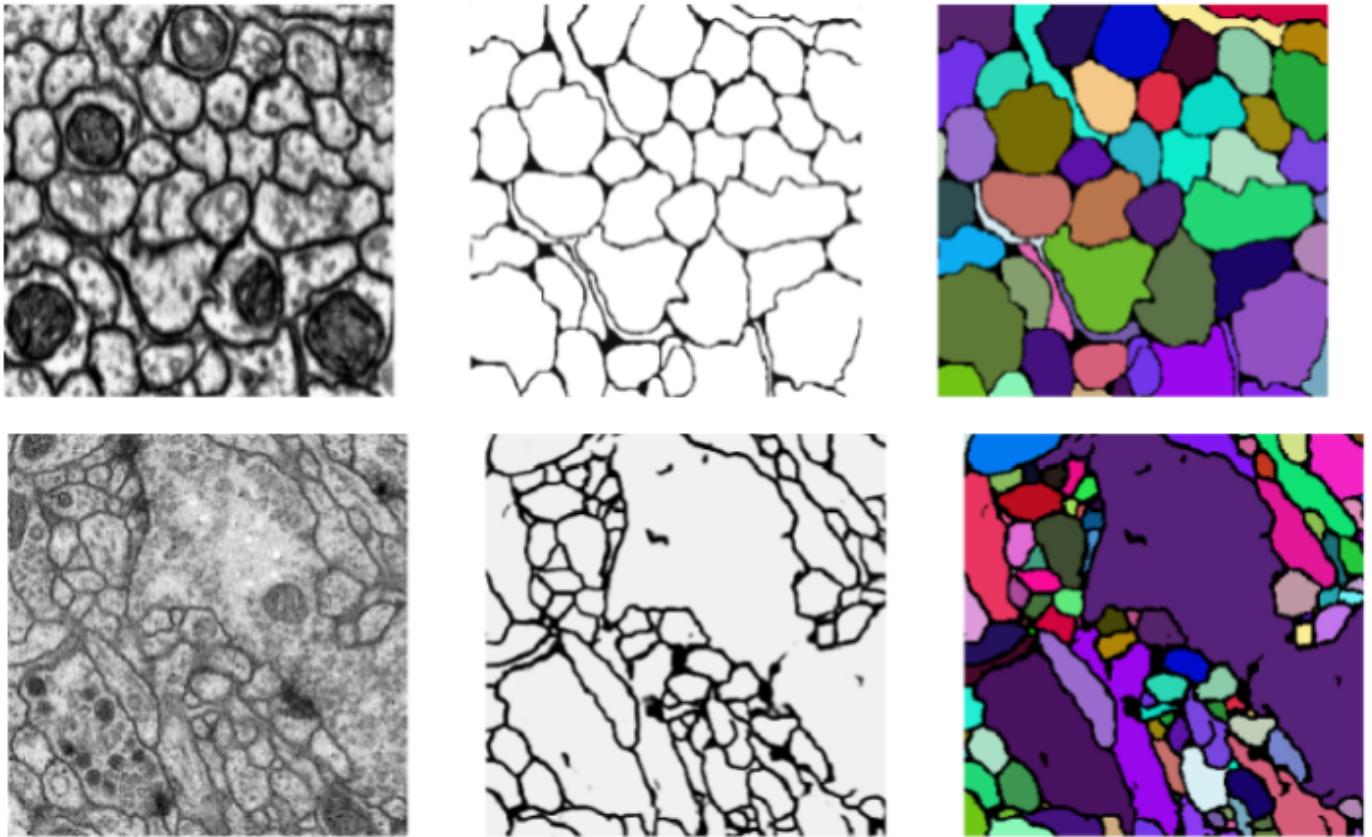


Figure 9. Example of U Net Applied in Biomedical Image

U net applied in crack detection Computer vision-based concrete crack detection using U-net fully convolutional networks - ScienceDirect (It has almost the same structure with original U-net, but we would use a different channel number, we would also use a different architecture with pretrained convolutional neural network as backbone (VGG, ResNet, Inception))

```

# Normalize input
inputs = tf.keras.layers.Input((IMG_WIDTH, IMG_HEIGHT, IMG_CHANNELS))
s = tf.keras.layers.Lambda(lambda x: x / 255)(inputs)

# Downsampling
c1 = tf.keras.layers.Conv2D(3, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(s)
c1 = tf.keras.layers.Conv2D(64, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(c1)
p1 = tf.keras.layers.MaxPooling2D((2,2))(c1)

c2 = tf.keras.layers.Conv2D(64, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(p1)
c2 = tf.keras.layers.Conv2D(128, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(c2)
p2 = tf.keras.layers.MaxPooling2D((2,2))(c2)

c3 = tf.keras.layers.Conv2D(128, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(p2)
c3 = tf.keras.layers.Conv2D(256, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(c3)
p3 = tf.keras.layers.MaxPooling2D((2,2))(c3)

c4 = tf.keras.layers.Conv2D(256, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(p3)
c4 = tf.keras.layers.Conv2D(512, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(c4)
p4 = tf.keras.layers.MaxPooling2D((2,2))(c4)

c5 = tf.keras.layers.Conv2D(512, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(p4)
c5 = tf.keras.layers.Conv2D(1024, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(c5)

# Upsampling
u6 = tf.keras.layers.Conv2DTranspose(512, (2,2), strides=(2,2), padding='same')(c5)
u6 = tf.keras.layers.concatenate([u6, c4])
c6 = tf.keras.layers.Conv2D(1024, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(u6)
c6 = tf.keras.layers.Conv2D(512, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(c6)

u7 = tf.keras.layers.Conv2DTranspose(256, (2,2), strides=(2,2), padding='same')(c6)
u7 = tf.keras.layers.concatenate([u7, c3])
c7 = tf.keras.layers.Conv2D(512, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(u7)
c7 = tf.keras.layers.Conv2D(256, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(c7)

u8 = tf.keras.layers.Conv2DTranspose(128, (2,2), strides=(2,2), padding='same')(c7)
u8 = tf.keras.layers.concatenate([u8, c2])
c8 = tf.keras.layers.Conv2D(256, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(u8)
c8 = tf.keras.layers.Conv2D(128, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(c8)

u9 = tf.keras.layers.Conv2DTranspose(64, (2,2), strides=(2,2), padding='same')(c8)
u9 = tf.keras.layers.concatenate([u9, c1])
c9 = tf.keras.layers.Conv2D(128, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(u9)
c9 = tf.keras.layers.Conv2D(64, (3,3), activation='relu', kernel_initializer='he_normal', padding='same')(c9)

# Output
outputs = tf.keras.layers.Conv2D(1,(1,1), activation='sigmoid')(c9)

```

Figure 13

Figure 10. Code of U Net

Input: images with 120x120x3 [height, width, channel (RGB)] (resized from 227x227)

Output: some kind of probability (was proceed by sigmoid)

Solver: Adam

Loss: Binary Cross Entropy

## References

Surface Crack Detection | Kaggle Concrete Crack Conglomerate Dataset | Virginia Tech Computer Vision-based Concrete Crack Detection using U-net Fully Convolutional Networks Data for: Computer Vision-based Concrete Crack Detection using U-net Fully Convolutional Networks Performance



Evaluation of Deep CNN-Based Crack Detection and Localization Techniques for Concrete Structures  
GitHub - agoor97/Cracks\_Detection: This Repository provides a Cracks Detection Implementation using Deep Learning for a Kaggle Dataset.

## References

---

[Energy Efficiency of buildings in New York Surface Crack Detection | Kaggle Concrete Crack Conglomerate Dataset](#) | [Virginia Tech Computer Vision-based Concrete Crack Detection using U-net Fully Convolutional Networks Network: U-Net Performance Evaluation of Deep CNN-Based Crack Detection and Localization Techniques for Concrete Structures](#)