# Image Classification For Quality Control In Construction Businesses

**Yueru He**
**Columbia University**
**yh3507@columbia.edu**

## Abstract

This research paper focuses on applying Convolutional Neural Networks (CNN) to the Quality Control Processes in the business field. Specifically, this paper illustrates how to use images of concrete surfaces to train a CNN for binary classification purposes, and then use the trained model to detect whether a building or road has structural issues. With parameter tuning and adjustment in structures and activation functions, the final CNN model reaches a remarkably high test accuracy of 99.5%. The data used for training is from construction businesses, but after fine-tuning, the trained model can be applied to broader industries, including automotive, retail, and food industries. The implementation of this model could facilitate companies' budgets on human resources for quality control and on loss reserves for possible accident claims.

## I. Business Background And Dataset

The quality control process is one of the most crucial steps before and after the products have been in use. This process is almost implemented in every business field. In the construction field, this is especially important to detect potential deficits in buildings and roads. Construction companies, building owners, or property management firms not only build buildings and roads but also do maintenance regularly. As years go by, there will be structural issues with the buildings and roads. For roads, if there is no on-time maintenance, transportation will worsen and will potentially increase the number of accidents caused by bad road conditions.

Therefore, the quality control of buildings and roads plays an important role in the construction businesses. Given the large number of roads and buildings across the country, it is very time-consuming or even not possible for companies to maintain every building on time, therefore an algorithm that can help them detect abnormalities on buildings and roads could be very valuable. By utilizing already implemented cameras, companies are able to collect pictures of every corner of buildings and roads. By image classification models, companies will be able to speed up the process of identifying potential structural issues before they become major problems. This could potentially save a huge amount of money on human resources and can prevent life-threatening situations (car accidents, house collapsing).

The classification model can also be applied to other businesses after fine-tuning, such as automobile manufacturing, and retail services. The model will be able to detect cracks in a car or cracks on retail products such as canned food and toys.

The original dataset uses contains images of concrete surfaces. There are 40,000 images in total, with 20,000 images of surfaces with cracks, and the rest of 20,000 images without cracks. Each picture is 227 by 227 pixels and with 3 color channels. Examples of such images are shown in Appendix 2. There is no data augmentation in terms of rotation or flipping included. Due to the limited computational power on the local laptop, this paper uses 1,012 images in total for training and testing purposes, with 506 images in negative and positive classes, respectively.

## II. Data Preparation

The data are from Kaggle, with the link attached in Appendix 1. So far I haven't found a related paper on this image set. The data points are colored pictures of concrete surfaces, some with cracks and some without cracks. These image data need to be first converted into tensors. Examples of original images are in Appendix 2. By utilizing the *jpeg* package in R, I read the 1,012 images as a large list with a length of 1,012, with each element as a tensor with dimensions 227*227. Since the image is colored image, there are three color channels. Each read-in image is of dimension 227*227*3. Therefore I reshape the data to a 4-D tensor with the following dimensions:

*(sample size, height, width, channels)*

The total data now has the dimension of 1012*277*277*3. I then split them into train and test sets. The training set contains 800 randomly selected images and the test set contains the rest 212 images. The values are already from 0 to 1, representing the degrees of red, green, and blue, so there is no need to standardize the data.

## III. Modeling

### 3.1 Model Selection

To do binary classification for image data, the best model is Convolutional Neural Network (CNN). It can automatically extract local features and build up from simple features to complex ones with multiple convolution and pooling layers. Compare to plain neural networks, CNN can detect local features instead of global features and therefore obtains translation invariance characteristics. This is suitable for detecting cracks which can potentially be at any place on the image.

Although the training data is a 4D tensor, the filters can only slide over the height and width axis, but not the channel axis. Therefore the filters need to be in 2-dimension, so as the shape of pooling layers. The input of the CNN is set to (227,227,3) for RGB images, and the activation functions need to be suitable for classification problems rather than regression problems.

### 3.2 Baseline Model

The baseline CNN model contains two convolution layers, two max-pooling layers, and one dense layer. For the first layer, I choose to use 32 3*3 filters and a 2*2 pooling size. I used 64 3*3 filters for the second convolution layer with a 2*2 pooling size. For the dense layer, I use 128 hidden units and add a dropout layer with a 50% dropout rate. I use ReLU for convolution and dense layer, and softmax for output layer. The baseline model structure and hyperparameters are similar to what we have learned in class. The evaluation metrics are validation loss and accuracy.

**Results** For training, I trained the model with 20 epochs and a validation set of 160 randomly selected data. The performance of this baseline model is not promising. With 20 epochs of training, while

the loss improves a lot after 10 epochs, the accuracy does not improve at all. The test accuracy is 51.4%, and the loss is 0.04.

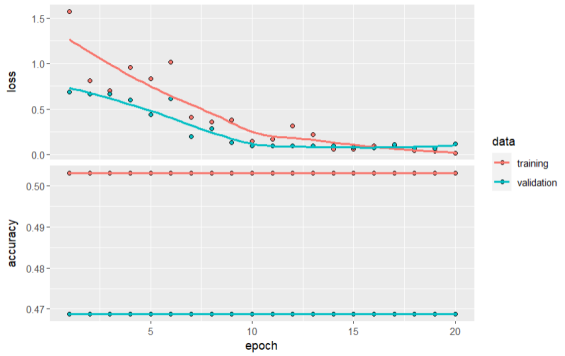


Figure. 3.2 Loss and Accuracy Plot For Baseline Model

I suspect the reason for the poor performance is that the model is not flexible enough and is underfitting. Therefore, in the following improvement, I add more filters and training epochs to increase the complexity of the model, expecting increased complexity to catch more patterns and information.

### 3.3 Improvement 1: Increase The Number Of Epochs and Filters Per Layer

I increase the number of filters to 64 and 128 for the first and second layers, respectively, and the number of hidden units in the dense layer remains the same. I also increase the number of epochs and add an early stopping algorithm to prevent overfitting. I set the patience for early stopping to be 20, meaning if there's no improvement in validation loss after 20 epochs, the training will be automatically stopped.

**Results** This time the improved model still has a similar performance as the baseline model, with good validation loss but never improved validation accuracy. The test accuracy is 51.4%, exactly the same as the baseline model.

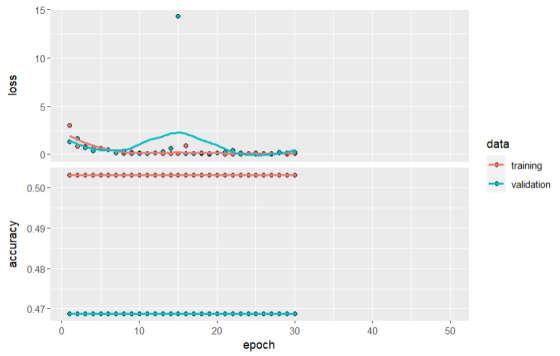


Figure 3.3 Improved Model Performance

The model has increased complexity the loss has improved except for one outlier. But it still has poor accuracy. One possible reason is that the model becomes better at identifying easy examples in the dataset but struggles with more complex ones. In this case, the decreased loss is largely due to the improvement on easier examples but is still making the same number of mistakes on the more difficult examples, resulting in no improvement in accuracy. Therefore the problem is not with the structure of the model but with other hyperparameters.

### 3.4 Improvement 2: Change Activation Function, Change Learning Rate of Optimization

After several iterations of changing different parameters, including types of the optimizer, max pooling size, dropout rate, and activation function. The final finding is that the poor performance is due to using the inappropriate activation function at the output layer. When changing the activation function of the output layer from Softmax to Sigmoid, the model accuracy improves dramatically.

**Possible Reasons** Although Softmax and Sigmoid activation functions are both used for classification, Sigmoid is often used as the activation function in binary classification because it maps any input value to a probability value between 0 and 1.

Softmax is typically used in multi-class classification tasks, where the goal is to classify an image into one of several classes. The output layer of the neural network has multiple units, one for each class, and the Softmax function normalizes the outputs so that they sum to 1, producing a probability distribution over the classes. Both activation functions output probability, but Sigmoid is more suitable for crack detection purposes since the dataset only contains two classes and has a small data size. Another potential reason for not using Soft max is that Softmax may be penalizing errors in the wrong class too heavily. In a multi-class classification task, Softmax is designed to penalize errors in all classes simultaneously, which can make it difficult to optimize the model for one class in particular. In a binary classification task, using Sigmoid can focus the optimization on the positive class, which can lead to better results.

For this model, I keep 64 and 128 filters for convolutional layers, and 128 hidden units for the dense layer. I also add two more dropout layers between convolution layers to prevent overfitting.

**Results** The test accuracy of this CNN model with Sigmoid activation function is 95.8%, and a loss of 0.29. The accuracy dramatically improves but the loss increases compare to previous models. The model is potentially overfitting the training data. When a model overfits, the training loss continues to decrease while the validation loss starts to increase. In this scenario, the model has effectively "memorized" the training data instead of learning the underlying patterns, resulting in poor performance on new data. The accuracy may improve on the test data, but the loss increases because the model is unable to capture the underlying patterns in the data.

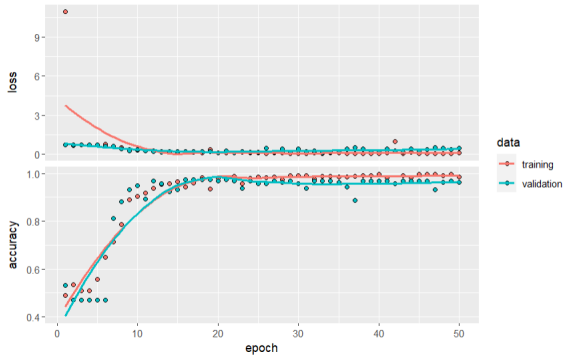


Figure 3.4 Improved Model With Sigmoid Activation Function

### 3.5 Improvement 3: Add More Layers

Since the model after the second improvement has remarkably higher accuracy, I perform further improvement based on it by adding one more convolutional layer and one more dense layer. By adding more complexity, I expect the model to either perform better because it captures more patterns. But it is also possible for the model to perform worse due to overfitting.

The additional convolution layer uses 256 filters, and both dense layers in this model have 156 hidden units. I also add two more dropout layers due to the overfitting results and increased complexity of this improved model.

**Results** The results for this final improved model are promising, with 99.5% of test accuracy, with a 3.7% increase. Because I use the early stopping technique, the model has a lower possibility of overfitting. The improved loss and accuracy plot is as follows. The test accuracy is 99.5% and the loss is 0.04. The loss is low again and the test accuracy is even higher than the previous model, suggesting that the improved model is the best model so far.

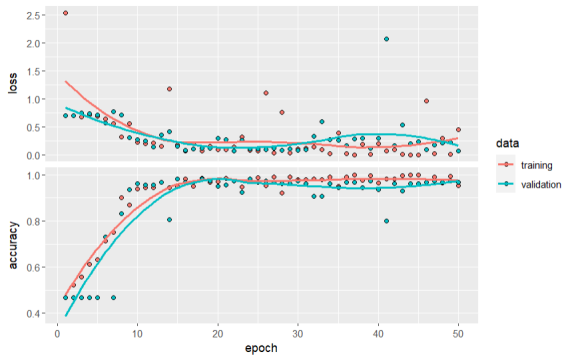


Figure 3.5 Final Improved Model With Increased Complexity

### 3.6 Model Comparison

| Model | Improvement | Test Accuracy | Loss |
|---|---|---|---|
| Baseline | - | 51.4% | 0.044 |
| Improvement 1 | Increase the number of epochs and the number of filters per layer | 51.4% | 0.089 |
| Improvement 2 | Change activation function, specify the learning rate of optimizer | 95.8% | 0.292 |
| Improvement 3 | Add one more convolution layer and one dense layer | 99.5% | 0.037 |

## IV. Conclusion and Further Discussion

### 4.1 Goals and Results

The above four models are trained and tested in R language on a local laptop. They jointly show the importance of hyperparameter tuning, the structure of the model, and the importance of the choice of activation functions. I developed a CNN-based image detection system aiming at speeding up the quality control process for construction businesses. It can detect structural issues in buildings and roads effectively.

The study used a dataset of 1,012 images of concrete surfaces, categorized as either having cracks or not having cracks. Four different CNN models were evaluated based on their accuracy in classifying the images. The models achieved accuracies of 51.4%, 51.4%, 95.8%, and 99.5%, respectively. The model improved because of the change in activation function, and increased level of complexity. The more complex model can capture more detailed patterns such as smaller cracks and cracks with grass on them. The results demonstrate the potential of CNN-based image detection systems in the construction industry, with high-accuracy models able to identify anomalies in concrete surfaces.

### 4.2 Further Discussion

Despite the high accuracy and low loss of the final classification model, the data size is not big enough. From the four loss and accuracy plots we can observe that there are comparably big variations between each epoch, which is an indication of lack of data. With only one thousand data points, the model suffers from biases. However, due to the limitation of computational power, this is the largest model that can be trained on a local laptop. Further improvement could focus on the scalability and practicality of the approach for real-world use in construction businesses.

**Generalization To Other Industries** The model developed in this research study was specifically trained to detect cracks on concrete surfaces. Nevertheless, the potential applications of the model extend beyond the construction industry. Cracks are a common anomaly in a variety of materials, including metal surfaces and paper. By utilizing fine-tuning or feature extraction techniques, the crack detection model can be adapted to detect anomalies in other industries, such as the automotive industry for car surfaces, the retail industry for product wrap, and the food industry for

canned food. Therefore, the developed model holds promise for enhancing quality control measures in a range of industries beyond the construction sector. Future research can explore the transferability of this model to these industries and investigate its potential benefits for various applications.

## V. Appendix

1. Data Source: https://www.kaggle.com/datasets/arunrk7/surface-crack-detection
2. Examples of input image
   Surface Image Without Cracks Surface Image With Cracks


3. Capstone Project Report: https://docs.google.com/document/d/1RJDPBoNwoX9tLSBqKeG-SNm8atPrPug-MoyCTx-NRW8/edit?usp=sharing
4. R notebook for this research paper: https://drive.google.com/drive/folders/1eBW-CTFyzrf7DX-sZKn2tZ6MpyLtLKm0?usp=sharing