Machine Learning Engineer Nanodegree

Capstone Project

Min-Shan Huang

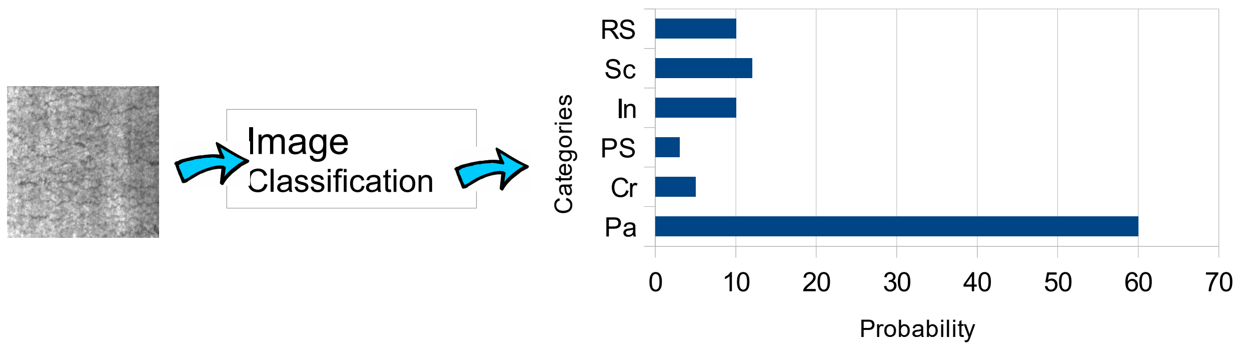
March 26, 2018

Definition

Project Overview

In real-life, people take amounts of photos for different goals every day. However, how to efficiently recognize or classify these photos is a important issue. With the improvement of Machine Learning, it has been found that Neural Networks which is a kind of model of deep learning have a dramatic impact on image recognition or classification. Even a simple model of Neural Networks is able to arrive a high accuracy for image predictions on most datasets from a variety of areas.

In industry, it is troublesome problem to find out which root cause which lead products to appear defects. To solve this issue, the first step is that classify the photos taken from cameras embedded on machine in manufacturing process which could help us whether defects have produced or not and understand how many types of defects exist.



Traditional approaches for classify defect images belong to Supervised Learning[1], so at first the definitions of features for each type of defects need to be defined by humans to help model distinguish them effectively before starting to classify. Then where the defects locate should be defected or labeled in datasets. The final step is

1 Q. Shubo, G. Shuai and Z. Tongxing, "Research on Paper Defects Recognition Based on SVM," *2010 WASE International Conference on Information Engineering*, Beidaihe, Hebei, 2010, pp. 177-180.

recognize which type each defect image belongs to.

In this project, I build a Convolutional Neural Network to classify steel surface defects effectively. The classifier is trained by using steel surface defect database from Northeastern University (NEU)[2].

Problem Statement

It exists a serious issue if using supervised model to classify defect images. Because each type of defects require to be defined the specific features to distinguish them, those classifiers could not recognize these images effectively if a new type of defects appears.

The goal of this project is to train a deep learning model that is able to classify a defect image automatically even if it does not belong to existed classes. Then the model is expected to be useful for distinguishing different types of defects, which means it owns precise predictions.

The tasks involved are the following:

1. Download NEU steel surface defect database[3] provided from Northeastern University
2. Observe the dataset and preprocess these images
3. Train the classifier that can recognize each type of defects
4. Do experiment to verify the classifier is effective

Metrics

This project adopts ***accuracy*** as the evaluation metric on the test images which are split from NEU steel surface defect database. ***Accuracy*** is a common metric which is suitable for multiple classes classifiers whose training datasets own balanced the amounts of each class.

$$accuracy = \frac{number\ of\ testing\ images\ which\ are\ classified\ correctly}{number\ of\ testing\ images}$$

According to the analysis of distribution in NEU database, we understand that

2 Song, K., and Yan, Y., "A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects," *Applied Surface Science*, vol. 285, pp. 858-864, 2013.

3 NEU surface defect database: http://faculty.neu.edu.cn/yunhyan/NEU_surface_defect_database.html

accuracy is a suitable metrics instead of F2-score which is used to measure the unbalanced classes.

Analysis

Data Exploration

The NEU steel surface defect database (NEU-CLS) totally includes 1,800 gray-scale images which contain six kinds of typical surface defects of the hot-rolled steel strip, i.e., rolled-in scale (RS), patches (Pa), crazing (Cr), pitted surface (PS), inclusion (In) and scratched (Sc).

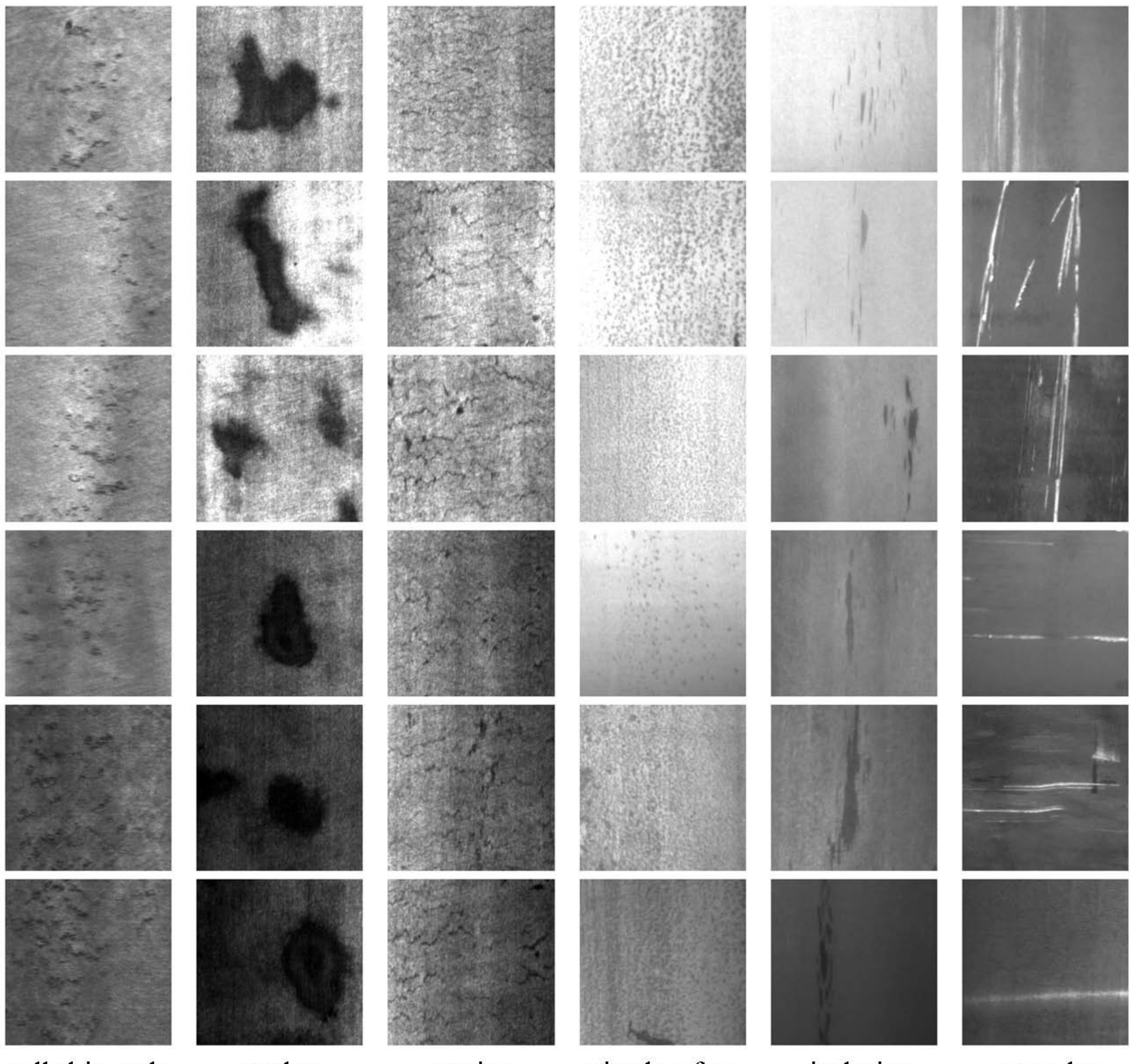


Fig. 1: NEU-CLS

In Fig. 1, it shows the sample image of six kind of typical surface defects, and the original resolution of each image is 200*200 pixels. Through observing these images, it is found this project may face two difficult challenges. First, these intra-class, the same kind of defects, should have similar aspects but some of them exist large differences in appearance, which means the kind of defects is difficult to be defined by features. Second, the defect images suffer from the impact of illumination and material changes. We could find some of images with high brightness, but remains look more dark, which may cause the model make wrong judgment.

Exploratory Visualization

Each kind of defects owns three hundreds samples in NEU-CLS database. In Fig. 2, it illustrates the distribution of six categories. It is obvious from this figure that the distribution is uniform.

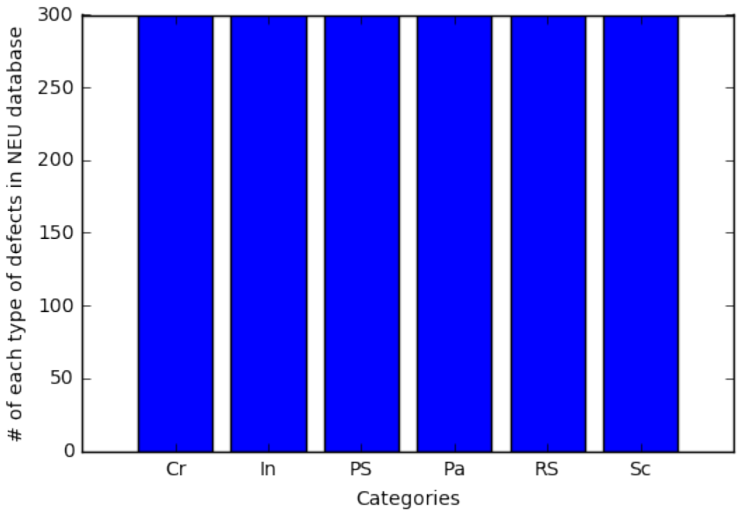


Fig. 2: # of occurrence of each category in NEU-CLS database

Pixel intensity is the brightness of each pixel and it's value is between 0 and 255. Here, I calculate average pixel intensity of each image called *pixel intensity means* and comparing *pixel intensity means* of each image in dataset could present the difference of image's brightness between those images. I compute standard deviation pixel intensity of each image named *pixel intensity std. devs.*. which represents the difference of pixel's brightness in one image.

In NEU-CLS database, average *pixel intensity means* is 128.30 and standard deviation of *pixel intensity means* is 44.50, which indicates that the brightness of these images exist variation. In addition, average *pixel intensity std. devs.* is 26.71, which means it owns smaller variation of pixel's brightness. The detail information of pixel intensity show in Fig. 3 and Fig.4.

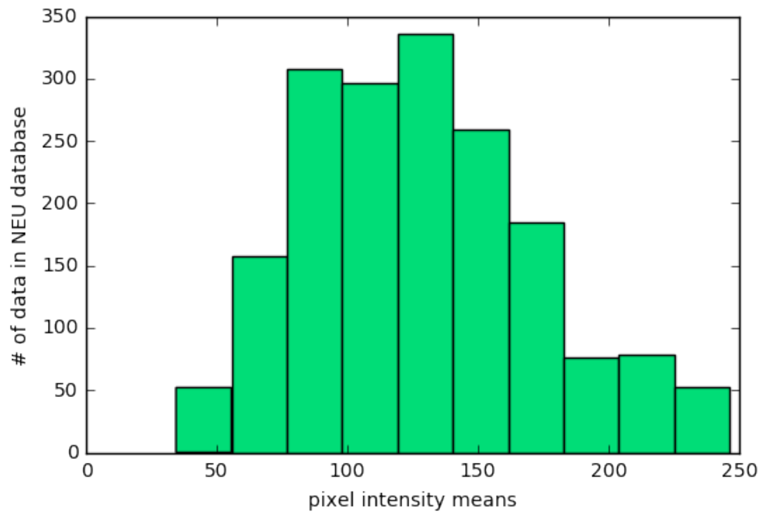


Fig. 3: Average pixel intensity means of each image in NEU-CLS database

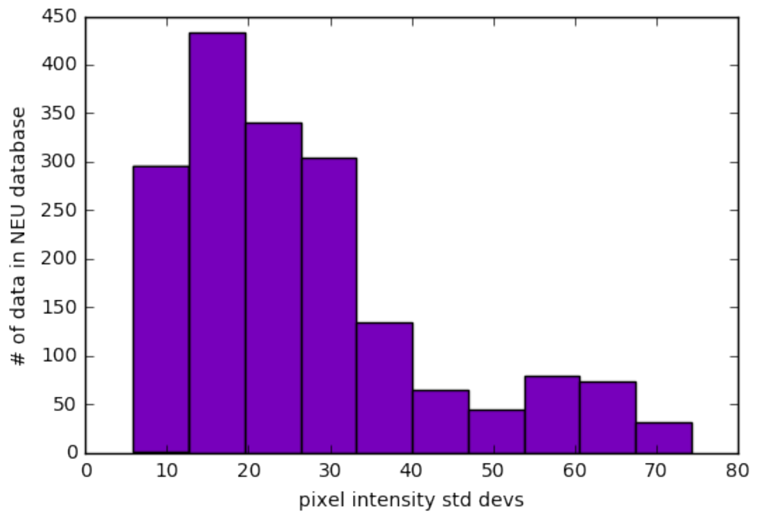


Fig. 4: Average pixel intensity std. devs. of each image in NEU-CLS database

Algorithms and Techniques

One of the state-of-the-art algorithm for image processing tasks which include image classification is Neural Network (NN). The algorithm could effectively find out feature patterns of images automatically instead of human definition. Higher-level features are class-sensitive but lower-level features are generic. Convolutional Neural Network (CNN) which is a special instance of NN is sensitive to local receptive field by sparsely connected to a small set of neurons and reduce the computation complexity efficiently.

Next, I introduce the basic components of CNN architecture show in Fig. 5. CNN algorithm consists of several convolution operations, including convolutional layer, pooling layer and fully connected layer.

- *Convolutional Layer (CNV Layer)*

  - Convolution is an operation of filtering the input data. In image classification, two dimensions pre-defined filters are used to process data at different layers to extract features.

- *Pooling Layer (PL Layer)*

  1. *Activation Function*

     - Activation Functions that is applied to each elements of convolition decide whether a neuron should be activated or not, which represents the information that neuron is receiving is relevant for the given information or should it be ignored. Basically, they can be divided into two types: linear (ReLU) and non-linear (Sigmoid).

  2. *Pooling*

     - Pooling sub-sample the input data after activation to reduce the size of the input data. Window size would influence the remain number at each dimension when doing reduction. The algorithm includes Max-Pooling and Global-Average-Pooling.

- *Fully Connected Layer (FC Layer)*

  - FC Layer will reduce the size of the output of PL Layer to the size of classes we want to predict.

- *Classification Layer (CL)*

  - CL is the final layer of CNN that converts the output of FC Layer to probability of each image being in a certain class. Soft-max type of algorithms are often adopted in this layer.

While compiling one CNN model, loss function and optimizer require to be set. In most learning networks, error is calculated as the difference between the real label and the predicted label, and loss function is used to compute this error. The common loss functions is mean squared error. Of course, we want to achieve minimum error when training a model, which represents our predict result is close to the ground truth. Therefore, optimizer whose algorithms include stochastic gradient descent is used to solve this problem. After setting these parameters, the model could start to be trained.

However, CNN needs a large amount of training data compared to other approaches, but NEU-CLS database does not have enough defect images. Therefore, I adopt data augmentation to solve this issue. Data augmentation which is used to increase the amount of training samples by existing images in NEU-CLS database could not only solve the insufficient dataset but also reduce the impact of varying lighting conditions without reducing the effective capacity.

Data augmentation generate data by below techniques: scaling, translation, rotation, flipping, adding salt and pepper noise, lighting condition, perspective transform, ... and so on.

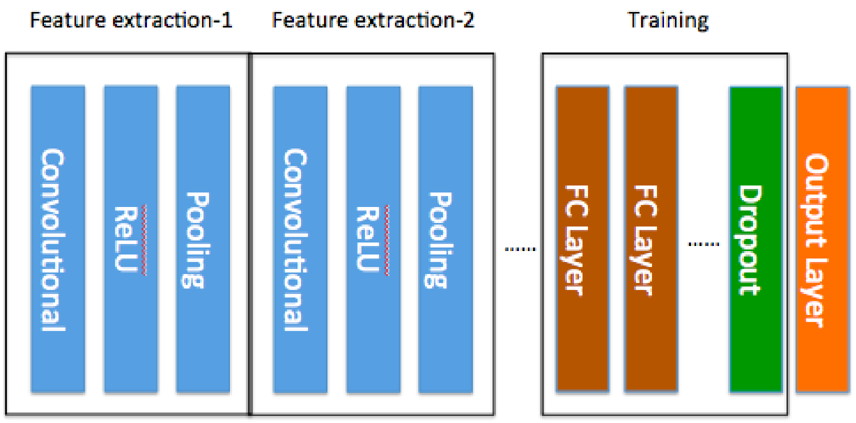


Fig. 5: Simple CNN Architecture

As a result, a CNN model with data augmentation would be built to classify NEU-CLS database effectively.

Benchmark

RESNET, GoogleNet, VGG-16 proposed from ImageNet Large Scale Visual Recognition Competition[4] (ILSVRC) are well-known networks to do image classification tasks. Incetion-V3 is advanced method based on GoogleNet which won the champion of ILSVRC in 2014 and it arrives high accuracy with 93.9%[5] on ImageNet Dataset.

The project would adopt transfer learning technique to recognize these steel surface defect images. A Incetion-V3 model pre-trained on ImageNet dataset which

4 ImageNet Large Scale Visual Recognition Competition (ILSVRC): http://www.image-net.org/challenges/LSVRC/

5 Google Research Blog: https://research.googleblog.com/2016/08/improving-inception-and-image.html

includes a variety of images is the benchmark model in this project.

It is excepted that our new designed CNN model could classify NEU dataset better than the benchmark model.

Methodology

Data Preprocessing

- *One-Hot Encoding*
  - One-Hot Encoding is a process by which categorical variables are converted into an array consists of 0 and 1 that could bring convenience to Machine Learning algorithms to do a better job in prediction.

RS	Pa	Cr	PS	In	Sc
[0, 0, 0, 0, 1, 0]	[0, 0, 0, 1, 0, 0]	[1, 0, 0, 0, 0, 0]	[0, 0, 1, 0, 0, 0]	[0, 1, 0, 0, 0, 0]	[0, 0, 0, 0, 0, 1]

Table 1: One-Hot Encoding for six categories in NEU

- *Split database into training data and testing data*
  - # of training dataset =1440
    - The detailed data distribution shows in Fig. 6
  - # of testing dataset = 360
    - The detailed data distribution shows in Fig. 7
  - Ratio between training and testing = 4 : 1

- *Split training dataset into validation training set and validation testing set in order to avoid overfitting*
  - # of validation training dataset =1152
    - The detailed data distribution shows in Fig. 8
  - # of validation testing dataset = 288
    - The detailed data distribution shows in Fig. 9
  - Ratio between training and testing = 4 : 1

- *Convert gray-scale images to RGB images*
  - The dimensions of input data of Inception-V3 pre-trained model = 3
-
- *Resize the images*
  - *224 * 224*

- The size of input data of Inception-V3 pre-trained model = 224 * 224

◦ *<u>40 * 40</u> , <u>200 * 200</u>*

- Zoom out the size of input data under reserving the characteristic of one defect image for to reduce the parameters of CNN which could improve computing speed.

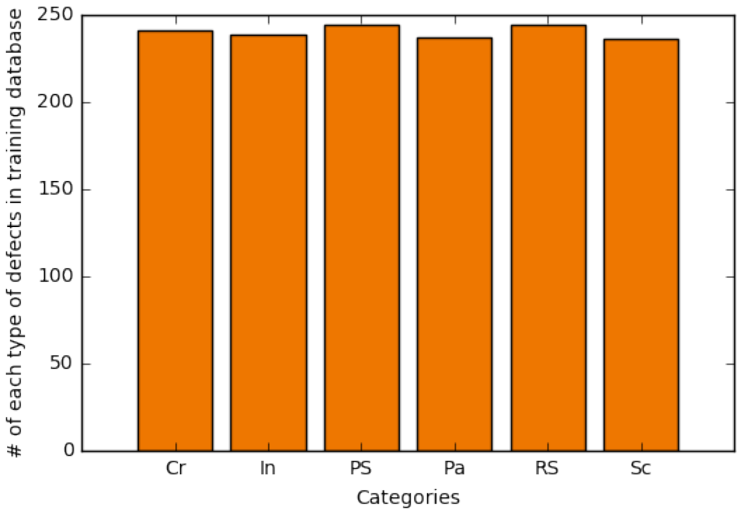


Fig. 6: # of occurrence of each category in training dataset

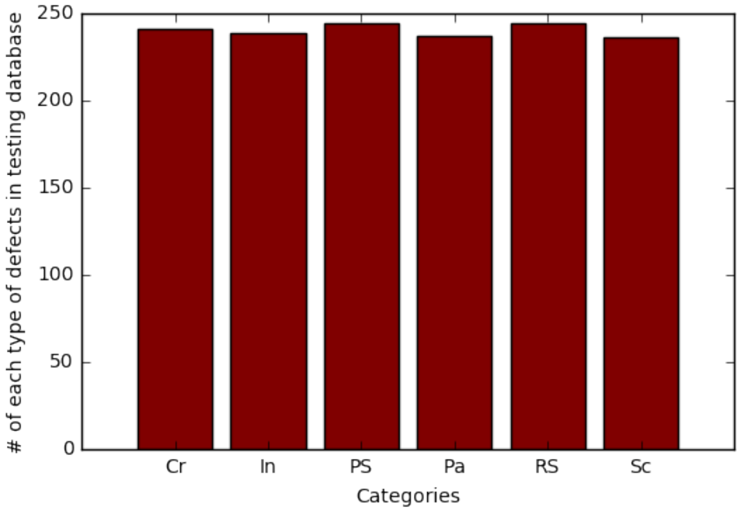


Fig. 7: # of occurrence of each category in testing dataset

- *Data augmentation*
  - *<u>Rotation</u>: rotate inputs randomly*
    - rotation_range = 40
  - *<u>Shift</u>: horizontal/vertical shift inputs randomly*
    - width_shift_range = 0.2
    - height_shift_range = 0.2

- *<u>Shear</u>: shear angle in counter-clockwise direction as radians*

- *<u>Zoom</u>: zoom in/out inputs randomly*

- *<u>Flip</u>: horizontal/vertical flip inputs randomly*
    - horizontal_flip = True

- *<u>Fill Mode</u>: points outside the boundaries of inputs are filled according to the given model*
    - fill_mode = 'nearest'

      mode: {'constant', 'nearest', 'reflect', 'wrap'}

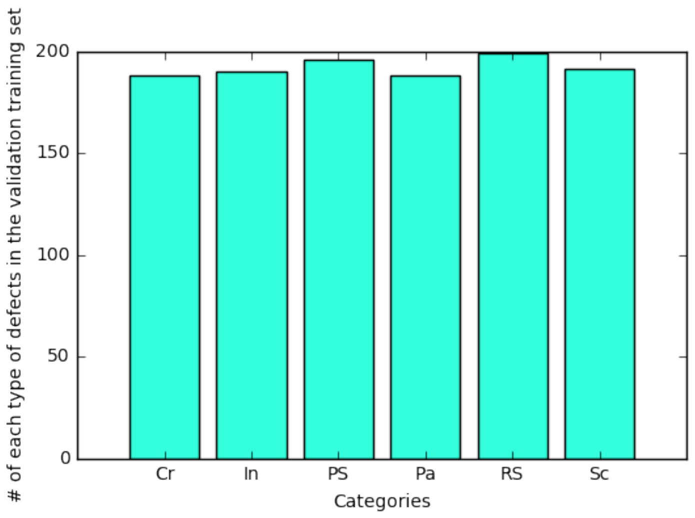


Fig. 8: # of occurrence of each category in validation training dataset

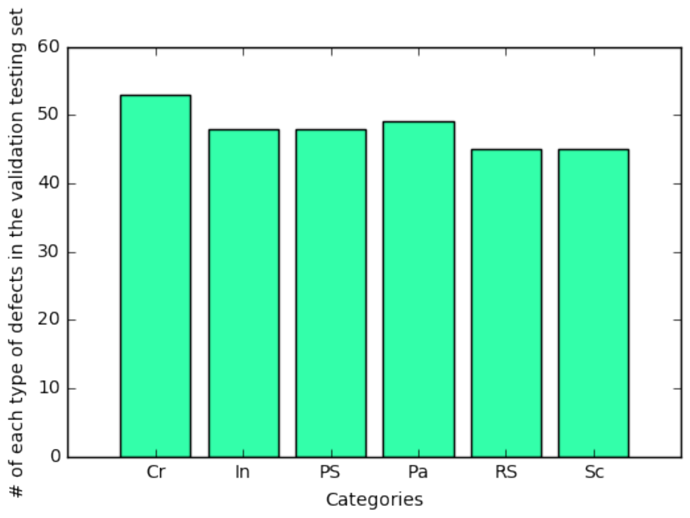
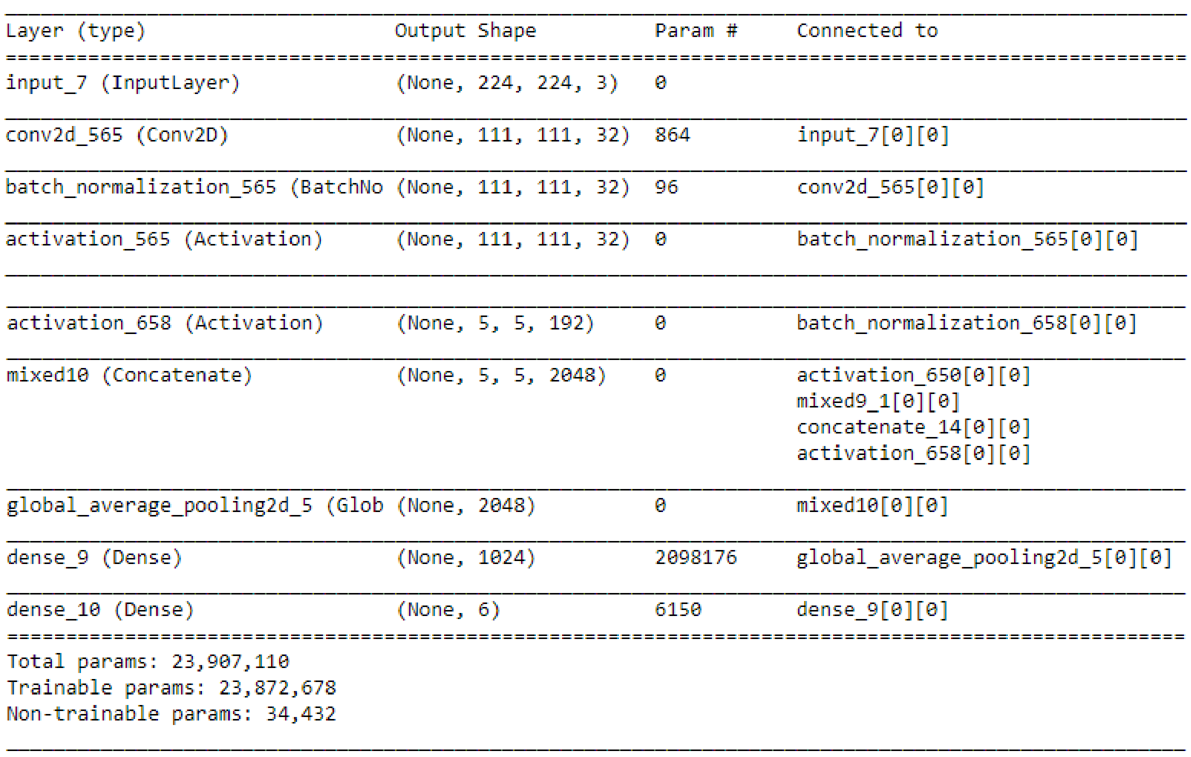


Fig. 9: # of occurrence of each category in validation testing dataset

<u>Implementation</u>

The classifiers was trained on the preprocessed training data, and the process can be further divided into the following steps:

1. Load whole defect images in NEU-CLS database
2. Preprocess them as described in the previous part
3. Build CNN classifiers
   - *Inception-V3*
     - Load the architecture and weights of pre-traind model based on ImageNet dataset
     - Modify the network architecture shown in Fig. 10 for matching the number of classes we want to predict
     - Define the optimizer, loss function and accuracy
     - Retrain the weights of this model



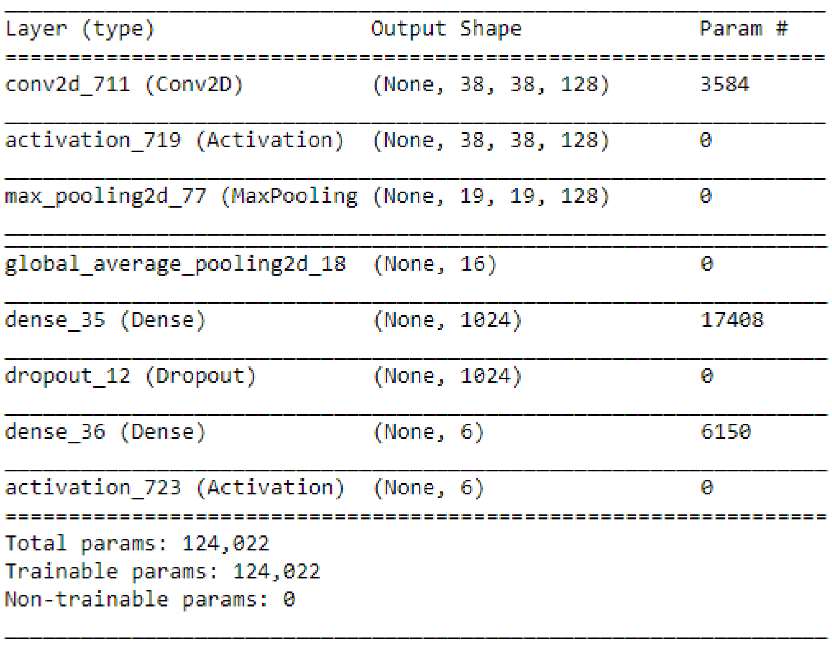
```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_7 (InputLayer)             (None, 224, 224, 3)   0
____________________________________________________________________________________________________
conv2d_565 (Conv2D)              (None, 111, 111, 32)  864         input_7[0][0]
____________________________________________________________________________________________________
batch_normalization_565 (BatchNo (None, 111, 111, 32)  96          conv2d_565[0][0]
____________________________________________________________________________________________________
activation_565 (Activation)      (None, 111, 111, 32)  0           batch_normalization_565[0][0]
____________________________________________________________________________________________________

____________________________________________________________________________________________________
activation_658 (Activation)      (None, 5, 5, 192)     0           batch_normalization_658[0][0]
____________________________________________________________________________________________________
mixed10 (Concatenate)            (None, 5, 5, 2048)    0           activation_650[0][0]
                                                                   mixed9_1[0][0]
                                                                   concatenate_14[0][0]
                                                                   activation_658[0][0]
____________________________________________________________________________________________________
global_average_pooling2d_5 (Glob (None, 2048)          0           mixed10[0][0]
____________________________________________________________________________________________________
dense_9 (Dense)                  (None, 1024)          2098176     global_average_pooling2d_5[0][0]
____________________________________________________________________________________________________
dense_10 (Dense)                 (None, 6)             6150        dense_9[0][0]
====================================================================================================
Total params: 23,907,110
Trainable params: 23,872,678
Non-trainable params: 34,432
____________________________________________________________________________________________________
```

Fig. 10: Inception-V3 model

   - *New designed CNN*
     - Define the network architecture shown in Fig. 11 and Fig. 12 and training parameters
     - Define the optimizer, loss function and accuracy
     - Start training the network
     - Redefine the network if the accuracy is not high enough

4. Save and freeze the network with the highest accuracy



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_711 (Conv2D)          (None, 38, 38, 128)       3584      
_________________________________________________________________
activation_719 (Activation)  (None, 38, 38, 128)       0         
_________________________________________________________________
max_pooling2d_77 (MaxPooling (None, 19, 19, 128)       0         
_________________________________________________________________
_________________________________________________________________
global_average_pooling2d_18  (None, 16)                0         
_________________________________________________________________
dense_35 (Dense)             (None, 1024)              17408     
_________________________________________________________________
dropout_12 (Dropout)         (None, 1024)              0         
_________________________________________________________________
dense_36 (Dense)             (None, 6)                 6150      
_________________________________________________________________
activation_723 (Activation)  (None, 6)                 0         
=================================================================
Total params: 124,022
Trainable params: 124,022
Non-trainable params: 0
_________________________________________________________________
```

Fig. 11: New designed CNN model

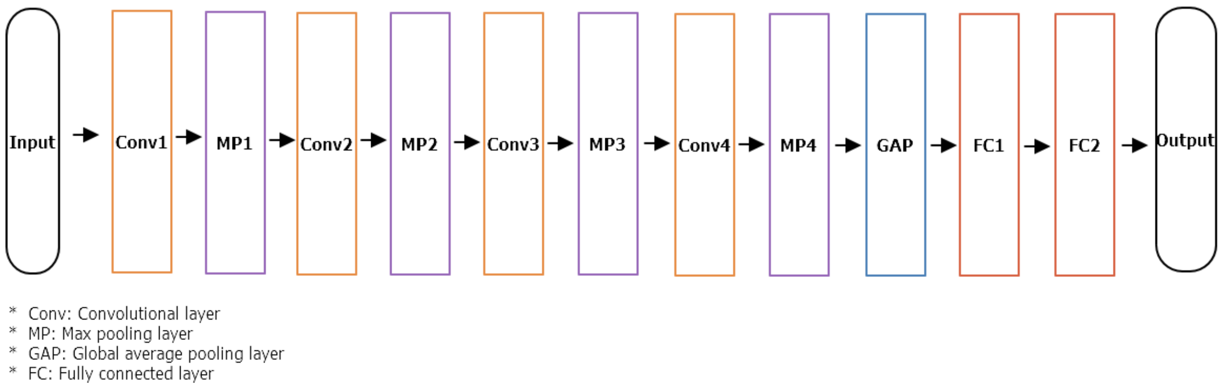

Input
Conv1
MP1
Conv2
MP2
Conv3
MP3
Conv4
MP4
GAP
FC1
FC2
Output
* Conv: Convolutional layer
* MP: Max pooling layer
* GAP: Global average pooling layer
* FC: Fully connected layer

Fig. 12: Architecture of new designed CNN

Refinement

As mentioned in the Benchmark section, the Inception-V3 model without data augmentation gets a not bad accuracy, around 39.44%. Though data augmentation technique, the accuracy is improved up to 76.11%.

However, we got the initial accuracy about 17% by our new designed CNN model with the original NEU-CLS dataset. To improve the effectiveness of defect image classification, the following techniques are used:

- The filters of convelutional layer
  - (32, 64, 128) → (128, 64, 32) → (128, 64, 32, 16)
- Resize image:
  - 200 * 200 → 40 * 40

- The number of epochs:
    - 10 → 25
- The type of input images:
    - RGB images → gray-scale images
- Data augmentation
    - Generate more training data

The final new designed model are derived by training with above techniques, and it has an accuracy of 87.50% shown in Table 2.

Results

<u>Model Evaluation and Validation</u>

Before starting to train models, whole dataset is divided to three subsets: training set, validation set, testing set. The validation set is used to avoid overfitting when training CNN model.

Though different adjustments which include model architecture, parameters, the type of input images and so on, our new designed CNN model arrives high accuracy with 87.50% and even exceed to our benchmark model who has an accuracy of 76.11%.

In Table 2, it shows the experiment results with varying size, epochs and whether data augmentation or not. The best result is 87.50% and it is along with the following list:

- The inputs are gray-scale images and all images are resized to 40 * 40, which not only improve the accuracy but also reduce the computing complexity.
- The shape of the filters of the convolutional layers = 3 * 3.
- The first convelutional layer learns 128 filters, the second learns 64 filters, the third learns 32 filters, the forth learns 16 filters.
- The activation functions are all adopted ReLU.
- The max-pooling size is 2 * 2 and it's stride is 2, so the resolution of the output matrices is half the resolution of the input matrices.
- The first fully connected layer has 1024 outputs, and the second has 6 outputs (the classes of we want to predict have 6 different types).

- The training runs for 25 epochs.

	Resize Image	Data Augmentation	Epochs	Gray/ Color	Accuracy
Inception-V3 (Benchmark)	224 * 224	X	10	Color	39.44%
	224 * 224	O	10	Color	76.11%
New Designed CNN	200 * 200	X	10	Color	17.78%
	40 * 40	X	10	Color	40.00%
	40 * 40	O	10	Color	69.17%
	40 * 40	X	10	Gray	63.05%
	40 * 40	O	10	Gray	70.28%
	40 * 40	O	25	Gray	87.50%

Table 2: Experiment Results

However, I think our new designed CNN model could be more robust on defect image classification. In next section, I will show which images could be recognized well and which images may be predicted poorly. Though the analysis, we could improve our model in the future work.

Conclusion

<u>Visualization</u>

		Prediction					
		Cr	**In**	**PS**	**Pa**	**RS**	**Sc**
Ground Truth	**Cr**	43	0	0	0	16	0
	In	0	54	7	0	1	0
	PS	1	14	39	0	2	0
	Pa	1	0	0	62	0	0
	RS	0	4	1	0	51	0
	Sc	0	6	2	0	2	54

Table 3: Matrix of ground truth and prediction

Shown in Table 3, we discover that Cr (crazing), Pa (patches), Sc (scratched) could be distinguished precisely, but In (inclusion), PS (pitted surface), RS (rolled-in scale) is easily confused by our designed CNN model. Nevertheless, I display these images in Table 4. Though observing them, I find several interesting points as following lists:

- The prediction result is impacted with varying lighting conditions lightly.
- Some images have similar features but they belong to different classes, i.e., Img. 0 and Img. 1. To solve this issue, we should clearly define each class, and make sure the data without wrong ground truth.

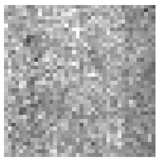
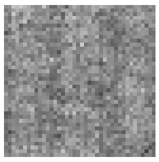
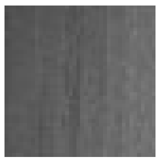
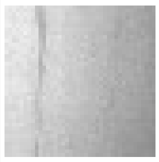
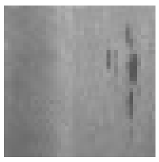
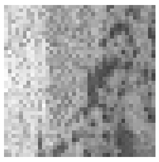
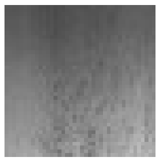
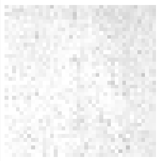
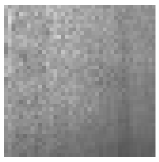
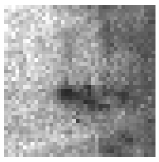
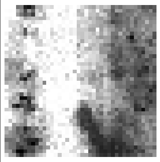
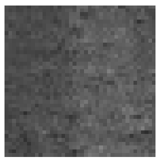
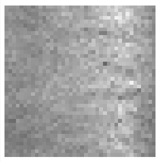
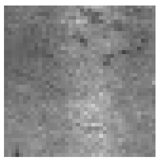
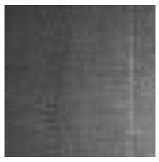
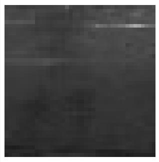
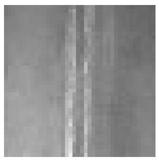
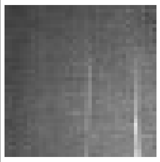
		Prediction					
		Cr	**In**	**PS**	**Pa**	**RS**	**Sc**
Ground Truth	**Cr**	 Img. 0				 Img. 1	
	In		 Img. 2	 Img. 3		 Img. 4	
	PS	 Img. 5	 Img. 6	 Img. 7		 Img. 8	
	Pa	 Img. 9			 Img. 10		
	RS		 Img. 11	 Img. 12		 Img. 13	
	Sc		 Img. 14	 Img. 15		 Img. 16	 Img. 17

Table 4: Defect images of ground truth and prediction

<u>Reflection</u>

The process used for this project can be summarized as below steps:

1. Initial a problem and search relative papers
2. Find out the issues from the past methods
3. Download the dataset and explore it
4. Preprocess the dataset
5. A benchmark classifier is created
6. Design a new classifier and train it using the preprocessed dataset
7. Find out why the classifier could not get a good accuracy
8. Retrain the classifier

The most difficult part in this project is step 7, because I need to think every possible causes and try different methods to prove them. Fortunately, I finally build a robust model with a high accuracy using the deep learning knowledge learned from Udacity to classify these defect images.

<u>Improvement</u>

To achieve a more precise classification, a hybrid model could be tried. Each model with different architectures or algorithms always have an aspect which they are good at. If we could combine their advantage, maybe the accuracy would be improved.

Furthermore, I think that defect images have similar features by which we could distinguish them. As a result. I hope could recognize other defect images using this model, and let it auto-retrain by transfer learning technique.