COMP 4211 Final Report
Juyoung BAE 20378582
Taek-Jung KIM 20383240

Anomaly Detection using Self-supervised Convolutional Autoencoder

Abstract

Machine learning in computer vision is in increasing trend. Advancement of this technology has enabled computers to have better accuracy in the classification of images. In this project, we explore the application of convolutional autoencoder and noise removal techniques to find the defective area in the images. We apply similar convolutional structures as Alexnet which was the winner of ILSVRC in 2012.

Introduction

Humans are evolved to be good at recognizing patterns, similarities, and differences. The studies have shown that cognitive recognition of humans exceeds all the other animals and these abilities were one of the most important traits that helped humans to survive[1]. Even though humans are the best at finding anomaly, manual inspection of human is not practical due to its expense and possible human error. Thus, machine learning systems seem to be the solution to this issue.
The current algorithm uses supervised learning technique where it trains the system from both non-defective and defective images. To effectively train the system, it requires a large dataset that demands huge efforts preparing masks for the defected segments of the images beforehand. In our project, we instead use one of the self-supervised (unsupervised) learning techniques, Convolutional Autoencoder. Instead of training the system with non-defective and defective images, we only feed the system with defect-free images. Autoencoder encodes the image into feature data and decodes it as similar to the original image. Studies have shown that architectures based on Convolutional Autoencoders are having sufficient performance in practical uses, such as U-net used in biomedical sector[2]. Anomaly detection has many real-world implications. As mentioned above, it can be used in the biomedical sector where autoencoders can diagnose patients' illnesses by looking at CT, MRI, and x-ray images. Industrial manufacturing sectors is one of the areas where anomaly detection is being used. A detailed inspection is required for quality control. Thus, anomaly detection can find deformed products accurately to improve the quality of the product.

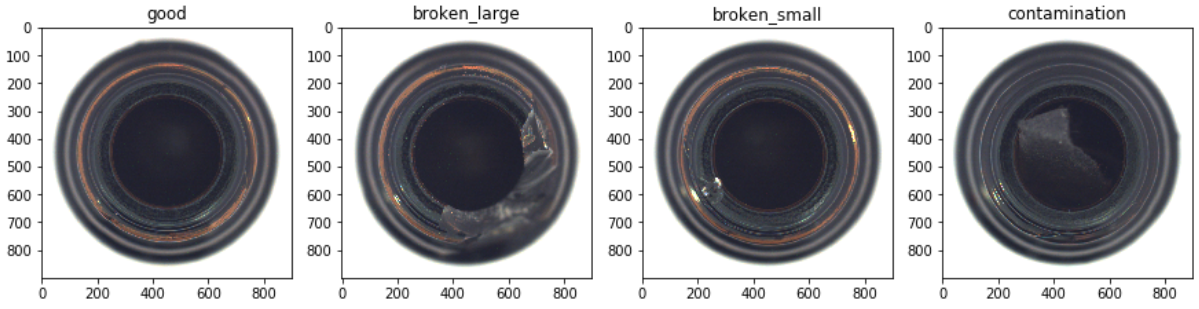

Figure 1: Sample non-defective(good) and detective bottle images

Dataset

MVTec Anomaly Detection dataset can be classified into two major groups, texture and object. The texture group has 5 categories (carpet, grid, leather, tile, and wood) and the object group has 10 categories (bottle, cable, capsule, hazelnut, metal nut, pill, screw, toothbrush, transistor, and zipper). From training the texture data (leather), our model encountered limitations, the disappearance of the texture feature, and the darkening of the image. Since the texture pattern of each leather image is slightly different, the model treated texture as a noise deleting these features. Also, since texture patterns are darker than the surface, it darkened the reconstructed image reducing performance. From training the object (hazelnut), another issue emerged. Since the orientation and size of the hazelnuts were different for each image, the output image was not accurately reconstructed.
Due to these limitations, we decided to use the bottle data set for our project. The image has a dimension of 900 x 900 pixels and is acquired as RGB format. The training set of bottles only contains defect-free images and the test set contains both non defective and defective images (broken large, broken small, contamination) with annotations (Figure 1).

Augmentation

The training set only provides 209 defect-free images which is not sufficient to train the model. Thus, we have to acquire sufficient data to train the model. One method is by randomly duplicating images. By creating a duplicate image and concatenate them into the training set, we can produce enough number of the training set. However, since it is random, some of the images can appear more than others. This might cause the model to be biased to one of the images. Even, if the number of duplicates per image is the same, this is exactly the same as the increasing number of epochs when training. Thus, we have decided to use image augmentation when duplicating the image. Not any augmentation can be used since augmentation should not demolish the consistency of the data set. Since the bottle is placed in the center of the image and has a circular shape, the orientation is vertically and horizontally symmetry. We have applied a horizontal flip and vertical flip to increase the size of the dataset. Eventually, 627 training set was acquired for this project.

Pre-processing

To improve the performance and efficiency of the training, few pre-processing methods are applied. Since the dataset is an image, it is consisting of a number between 0 and 255. While training with large numeric values, due to the predetermined value of the parameter, the data might collapse to zero or infinity. Thus, it is better to reduce the range of values. We have first considered using normalization (involves division by standard deviation). However, since the bottle image is consisted of mostly black, this produced values compacted to zero. Since small values can also cause data to collapse, we decided to under scale(normalize) to value between 0 and 1. This is beneficial in the evaluation stage since the activation function sigmoid produces values between 0 and 1 as well. Thus, it is straightforward to compare the original image and the reconstructed image.
The image was transformed into grayscale before the training. To use structural similarity loss (to be explained in detail in Structural Similarity section of the report), it is required to use a grayscale image. Also, by using grayscale, it reduces three channels into one channel. Thus, the number of the input variables is one third, less computation is needed.
The image was resized to 227 x 227. Since our model use a similar structure as Alexnet, which inputs image 227 x 227. Thus we also have reduced the size of the image when training the model.

Convolutional Autoencoder

For this project, convolutional Autoencoder was used. Autoencoder is a type of artificial neural network that enables to encode images to lower dimensionality and decode these values to reconstruct original data[3]. If the encoding function is **W** and the decoding function is **U**, where input is **x**.

$$x' = U(Wx)$$
$$\text{Where } x' \approx x$$

The encoding function and decoding function is the inverse function to each other. Since autoencoder compares the output with its input, it is self-supervised learning (unsupervised learning).
Unlike normal autoencoder using a feed-forward neural network, convolutional autoencoder uses convolutional operations to achieve output identical to the input. A convolutional autoencoder is more suitable for image processing since a convolutional operation is usually faster than a feed-forward neural network since matrix multiplication requires more computational power than convolutional operation. Also, a convolutional operation is better at extracting features of the images.
In convolutional autoencoder, encoding is done by two major operations, convolution layer and max-pooling layer. Convolutional layers are also known as filters, where it sums elementwise matrix multiplication between the value of the filter and values of the image under the filter. This enables systems to learn features, such as lines, curves, edges, and patterns. Max pooling is an operation where it finds the maximum value of the image under the pooling frame, pass it on to the output matrix. The maximum value represents all the values under the pooling frame and other values will be discarded. By using strides to these layers, it reduces the dimension of the input image, encoded through the process.

Decoding is done by the inverse of these operations, convolutional transpose, and upsampling. Convolutional transpose is inverse of a convolutional layer where it extracts one value from the input, multiplying with the filter to create multiple values. This operation extracts feature vectors to predict the original form. Upsampling is inverse of max pooling where it instead extracts one value and duplicates to multiple values. These operations with strides increase the dimensionality the same as the corresponding encoding stage.
Anomaly detection uses denoising properties of autoencoders to find the deformed area. When reconstructed, autoencoder is trained to not produce output identical to input but gives a general form of the image. Thus, even though the input image has defects, autoencoder generates output of an object that does not have defects. By using the difference between input and output, we are able to detect the anomaly (explained later).

Environment

We used TensorFlow Keras API to implement the model, and Nvidia GTX 1070 and Tesla P100 (Google Colab Pro) GPU to meet our computational needs.

Architecture

Our design of autoencoder architecture is largely based on the AlexNet convolutional neural network. AlexNet came out in 2012 as a revolutionary advancement. Winning the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) competition It proved itself to be one of the most capable models for detection tasks with notable features such as overlapping pooling and Relu activation. The contraction path of our model directly reflects the AlexNet model, while the expansion path is an inverted configuration of the same model. To implement the bottleneck part of the autoencoder, we introduce a modification. AlexNet has two dense hidden layers with 4096 neurons, followed by one output layer that performs the classification into 1000 classes. The first attempt was to use the two hidden dense layers of original AlexNet as the bottleneck. However, such implementation of the bottleneck yielded an extremely large number of parameters, leading to significant overfitting as well as inefficiency in training. Thus, we reduced the size of the bottleneck to a single fully connected layer of 2048 neurons, after which our autoencoder model enters its expansion path.

For our model, we have decided to use convolutional transpose instead of using upsampling. As explained above, convolutional transpose is inverse of convolutional, and upsampling is inverse of max pooling. Since our model has three max-pooling layer on the encoding stage, three upsampling layers on the decoding stage should be applied. However, we instead used a convolutional transpose layer to inverse a max-pooling layer. For upsampling, it only duplicates one feature value to many, where convolutional transpose uses a filter to multiply values. Since a max-pooling layer discarded many features that might be useful during the operation, using convolutional transpose seemed more appropriate to restore these losses.

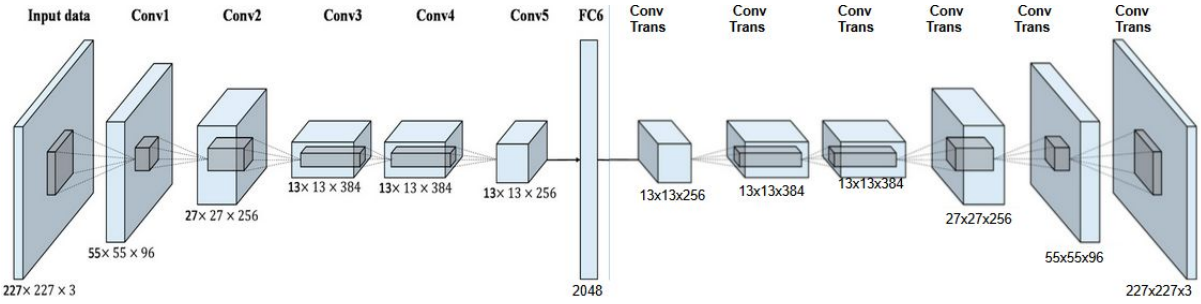

Figure 2: Finalized custom convolutional autoencoder model

One of the aspects that we have tried implementing on our model was using an average layer. By looking at U-net model, it is known to use a concatenate layer at each decoding stage. It concatenates the current decoding layer with the corresponding encoding layer, which produces better reconstruction of the original image. However, a concatenate layer increases the number of parameters hence we instead inserted an average layer. By using an average layer instead, it has resulted similar effect with the same number of parameters. However, on the test set, it has reproduced output so similar to the original image, which output also had anomaly and had loss close to zero. Since the reconstructed image was too accurate, we have decided to use only one average layer at one of the decoding stages. During this experiment, we have found some relationships between the details of the reconstructed image and the location of the average layer. For having an average layer close to the bottleneck, the effect of the average layer was negligible. Having an average layer at the second decoding stage resulted output image to have a similar outline as the input image (including defect segments). For the third layer, it produced output image to have shadows and brightness more similar to the original image. fourth and fifth had reproduced image almost identical to the original input. However, on our final model, we have decided to remove the average layer to improve the performance. Having an average layer at the first decoding stage was negligible, just consuming computation, and having an average layer at the second decoding stage resulted worse performance since it has produced an outline of the anomaly on the output image. If the layer was deeper, having an average layer might have helped to improve the performance. However, with only five layers, it was not beneficial.

Structural Similarity

Our project features the concept of structural similarity (SSIM) metric (Wang et al., 2004)[4] as an alternative to per-pixel l2 metric for evaluating the difference between two images. With its simplicity and computational inexpensiveness, per-pixel L2 metric is widely considered as a default choice in many applications. However, as simple as it is, per-pixel L2 distance metric only considers the absolute difference between individual pixel values. In other words, it only considers the respective color discrepancy to detect the differences. Therefore, in applications where the difference between the input and the target images are heavily inclined to more subtle discrepancies than explicit color, per-pixel L2 distance metric

is not a reliable approach. On the other hand, SSIM metric defines a distance measure between two images by taking into account of their similarity in luminance, contrast, and structure. Our project aims to detect any kind of anomaly in the image that deviates from its reconstructed counterpart. Thus, it is imperative that our model:

1. Learn all the subtle aspects of the defect-free images during training to yield a clear reconstruction that reflects those features, and

2. Precisely detect the defect characterized not only by explicit color difference but also by more wide range of criteria when comparing between the anomalous test images and their reconstructions.

Thus, as a part of the experiment, we introduce the concept of structural similarity (SSIM) metric to two of the following applications to satisfy above requirements and compare the performance to approach with per-pixel l2 alternative.

1. As a loss function to guide our training

2. As an evaluation metric to measure the reconstruction error between the original test images and their reconstructed variants.

Input Handicap

Other than the anomaly detection task introduced in this work, noise removal is another widely accepted application of self-supervised autoencoder model. Self supervised noise removal task first involves duplicating the original, noise-free training images. Each of the training images in one version of the training set is introduced with random noises, and the other version remains as noise-free. For the training of the autoencoder model, noised version of the training images is fed as input, and the corresponding noise-free training images serve as a target. This arrangement of self-supervised learning allows the autoencoder model to learn and identify the inconsistency in the input image introduced by the noise. Consequently, when given a noised test image, the model can reproduce the noise-free counterpart.

In this experiment, we attempt to incorporate the same principle used in the noise removal task to our self-supervised autoencoder model. Every images of non-defective object in the MVTec dataset – and consequently in real-world applications – are not identical to each other. Although they are all equally labeled as non-defective, because every non-defective object is of different samples, there must be some minor inconsistency even among the non-defective images. Unlike in the pure noise removal task, during the training of the autoencoder model used in this work, such embedded inconsistencies cannot be manually removed in the target image. In other words, the inconsistency is present in both the input and the target; consequently, it will be treated as being part of the “non-defective” features during the training and learnt accordingly. As a result, we speculated that such embedded inconsistencies among the non-defective training images learned during training can be reproduced in the reconstruction of test image. This is not desirable because such

embedded inconsistencies of training images are not part of the feature that constitutes the “non-defective” criteria and therefore should not be considered during the testing.

To counter this problem, we attempted to artificially amplify such inconsistency during training by introducing various input handicaps to images used as training input, while keeping the corresponding target counterparts as handicap-free. By amplifying the inconsistency to the training input, we speculated that the model can be aware of the presence of inconsistency among the input training images. Also, by introducing such handicap only to the input training images, the original embedded inconsistencies that were already present among different non defective images are slightly altered in the input. In other words, such embedded inconsistencies are no longer present in the same form in the input and the target.

Thus, in theory, we speculated that the autoencoder would not be able to distinguish whether the certain inconsistency found in the input training image is part of the artificially added input handicap, or part of the pre-existing embedded inconsistency that are in fact present in the target as well. As a result, regardless of the case, the autoencoder will learn to treat it as an inconsistent feature that needs to be disregarded during the reconstruction of test images. As a part of the experiment, we investigated whether amplifying the inconsistency through input handicap can indeed demonstrate what we speculated and yield more robust performance.

Experiment - Training Phase

Prior to the experiment, all images are processed to input size of 227 x 227, gray scaled to single channel image, and normalized to range of 0~1. Data augmentation is performed to increase the number of available training images. Each transformed versions of the entire training images are directly concatenated to the original training set and shuffled.

To study the effect of input handicaps to overall performance of the model, we apply two input handicaps to each image in the augmented and shuffled training image set. Handicaps include:

- Add gaussian noise of range 0 to 0.05, sampled once per pixel from a normal distribution
- Apply a black cutout of size 15% of the input image at random location.

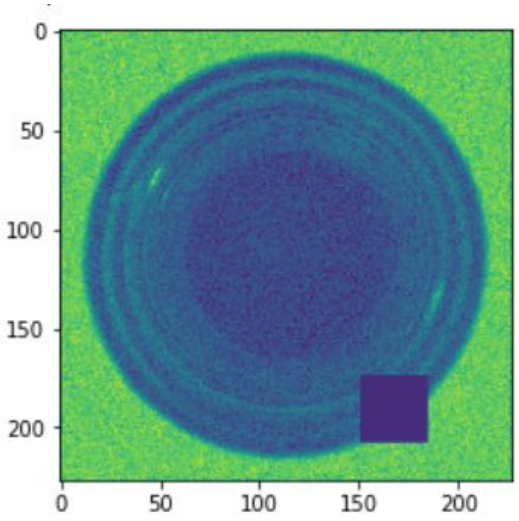

Figure 3: Sample input training image with input handicaps applied

Processed training images serve the roles of both the input and target to the model. Therefore, to ensure that the output images have the same dimension and range, the final convolutional layer needs to have single filter to yield single channel output and a sigmoid activation to enforce the range to 0~1 as well.

We employed two different loss functions – SSIM Loss and Mean-Squared (L2) Loss – as an experiment to compare their reconstruction capability and a its effect on actual anomaly detection performance. Unlike the mean squared loss, SSIM Loss is not defined in the Keras library for immediate use in the parameter. Thus, we defined the SSIM Loss accordingly.

By definition, loss function returns a smaller value as the output image becomes closer to the target. However, SSIM metric returns a higher value for higher similarity. Thus, in order to utilize SSIM metric as a loss function, our custom defined SSIM loss returns its additive inverse. Parameters used for the SSIM metric are set as default settings provided by Tensorflow image library.

	Training Loss	Input handicap
Model 0 (Baseline)	SSIM	x
Model 1	Mean-squared	x
Model 2	SSIM	o

Table 1: Summary of model variations to be experimented

Table 1 summarizes all the model variations to be used in this experiment. First model is defined as baseline. Training the baseline model involves SSIM train loss, and handicaps are not applied to input image. Second model uses a conventional pixel-wise mean squared error loss function instead of SSIM loss function. Other features are identical to the baseline model. Third model applies handicap to the input images. Likewise, other features are identical to the baseline model.

We implement a grid search approach to determine the Optimizer, learning rate, and batch size that suit best for the experiment.

Optimizer	**Adam**		RMS Prop	SGD with Momentum
Learning rate	5e-5	**1e-4**	5e-4	1e-3	5e-3
Batch size	4	**8**	16	32	64

Table 2: Summary of entries in hyperparameter grid search

Through grid search, we observed that Adam optimizer, learning rate of 1e-4, and a batch size of 8 yielded best performance for majority of the variations introduced in this experiment. We label this hyperparameter setting as ‘default’ setting to be consistently used throughout the experiment. For small number of variations, other hyperparameter settings yielded a slightly better performance; however, the improvement was not only inconsistent, but also significantly small compared to using a default hyperparameter setting. Therefore, for the sake of consistency, we use the default hyperparameter setting for all variations of the experiment.

We train the model for maximum 150 epochs with early stopping of 10 epoch-patience. Early stopping is an effective approach that discourages potential overfitting from excessive training. 5% of the augmented training images are reserved for training validation to be used for loss evaluation after the end of after every epoch. If the validation loss does not improve for 10 consecutive epochs, we cease the training and return the most recent model with lowest training validation loss.

Experiment - Evaluation phase

The first and most fundamental step for evaluating the performance of self supervised detection task is to **compare** the original test image and its reconstructed counterpart. The output of the comparison can then be processed and utilized for every possible evaluation criterion. In other words, with different approach for comparison, the evaluation results generated from each comparison metric are fundamentally different.
For this project, we adopt following two image comparison metrics:

- SSIM difference
- Per-pixel l2 difference

For each evaluation criteria, we generated two independent result, each obtained from two different comparison metrics. We aimed to observe how different comparison metric directly affects the evaluation. Thus, we ensured that the evaluation criteria that we discuss in this experiment to be consistent across different comparison metrics. For implementing the SSIM difference metric, we used default parameters settings provided by skimage library.

In this experiment, we introduce two different evaluation criteria – qualitative and quantitative – to evaluate the performance of each variations of the experiment. To qualitatively evaluate the performance, we generated a difference map that highlights the region of discrepancy between the test image and its reconstructed image. For each [test image, reconstructed counterpart] pair, we visualized two independent difference maps generated from SSIM and per-pixel l2 comparison metric, respectively. The objective of this qualitative approach is to visually evaluate whether the defective region is correctly highlighted in the generated difference map, while non-defective region is not.

Although not included in this project, one possible extension to this visual approach would be to introduce an intensity threshold to the difference maps. These threshold-applied difference map can then be evaluated on how much it overlaps with the ground truth anomaly mask, which is also provided in the MVTec dataset. The result of this extension can be used to precisely quantify the detection performance.

On the other hand, quantitative approach aims to utilize the comparison results and perform a **classification** on whether the object shown in the test image is defective or not. To achieve this, we introduced ‘difference score’ to quantify the difference between the original test image and reconstructed counterpart. Difference score is simply an average of all the pixel values of the difference map generated for the qualitative approach. For each [test image, reconstructed counterpart] pair, we considered two comparison metric – SSIM and per-pixel L2 – to generate the maps. Hence, we calculated the two difference scores per pair, each obtained from corresponding difference maps.

Theoretically, high reconstruction error of the test image necessarily means large difference between the test image and its reconstructed counterpart. The region that yields such difference is marked with large pixel values in the difference map. Therefore, difference map generated by reconstructing a defective image is more likely to have larger average pixel values than a difference map generated by reconstructing a non-defective image. To perform a classification on the obtained score, a threshold needs to be determined. For this purpose, we further split the given test set to ‘threshold validation set’ and ‘absolute test set’. (For clarification, threshold validation set mentioned in this section is unrelated to the training validation set used during training.) Threshold validation set is obtained by randomly sampling the difference scores, 30% each from defective and non-defective test set, respectively. Remaining difference scores in the test set is reserved as absolute test, which is used only for evaluating the classification performance.

Using the difference scores of images in threshold validation set, we determined the threshold as a value that yields least number of misclassifications. With the obtained

threshold, we perform the classification on the difference scores of the absolute test set. In the data set, the number of non-defective test image is not equal to the number of defective test images. In other words, cost of misclassification of defective test image is not the same as the cost of misclassification of non-defective test images. In fact, in typical real-life industrial applications, number of non-defective products is significantly larger than that of defective products. We took consideration of this property of the task and adopted F1 score as a metric to evaluate the classification performance.

Experiment Result

We discuss the result from two different approaches:

- Qualitative evaluation using the visuals of the test image, reconstructed counterpart, and two different difference images.
- Quantitative evaluation of classifying the presence of the defect using difference scores each obtained from respective difference images.

For each model variations of the experiment, sample visuals used in this section for discussion are manually selected to be the best representative of all test images.

1. Choice of loss function

To study the effect of using different training loss functions, we compared the performance between the baseline model that uses SSIM Loss function and its variation that uses the conventional mean squared loss instead. For reconstruction capability, we can observe that the baseline model trained with SSIM Loss significantly outperforms the model trained with the conventional mean squared loss.

Figure 4: Visuals of result from sample **defective test image** obtained from the **baseline model**. From left to right, each sub-figure is labeled Figure 4-1, 4-2, 4-3, 4-4, and 4-5, respectively

Figure 5: Visuals of result from sample **non-defective test image** obtained from the **baseline model**.

From left to right, each sub-figure is labeled Figure 5-1, 5-2, 5-3, and 5-4, respectively

Figure 6: Visuals of result from sample **defective test image** obtained from **model trained with MSE Loss**
From left to right, each sub-figure is labeled Figure 6-1, 6-2, 6-3, 6-4, and 6-5, respectively

Figure 7: Visuals of result from sample **non-defective test image** obtained from **model trained with MSE Loss**
From left to right, each sub-figure is labeled Figure 7-1, 7-2, 7-3, and 7-4, respectively

Second sub-figures of Figure 4 and Figure 6 (Figure 4-2 and Figure 6-2) illustrates the reconstructed version of same defective bottle image, each from a baseline model with SSIM loss and mean squared loss. Figure 4-2 successfully ignored the cracked defect present in the original image and replaced the defective region with non-defective features in the reconstructed image. However, Figure 6-2 exhibits a major distortion of the region where the defect is present in the original image. During training, mean squared loss can only quantify the loss as an absolute pixel difference. In other words, the focus of the training is only to match the pixel colors of the output to those of target. As a result, during testing, when the model receives an input with an unseen structure difference that deviates from the non-defective training images, the model faces difficulty to recover the exact structure only with information of each pixel's color, thus resulting in distorted reconstruction. On the other hand, SSIM Loss quantifies the loss as a difference of more sophisticated features of luminance, contrast, and structure. Thus, during training, model learns those in-depth aspects of the non-defective training image, and when given a test image with the same defect, it can perform more reliable reconstruction.

Such advantage of SSIM Loss over mean squared loss can be also discussed in reconstruction capabilities of non-defective test images. Third and Forth sub-figures of Figure 5 (Figure 5-3 and Figure 5-4) are SSIM and L2 difference images, respectively, between non defective test image and its reconstruction predicted from a model trained with SSIM Loss. Third and Forth sub-figures of Figure 7 (Figure 7-3 and Figure 7-4) are the difference images generated by identical SSIM and L2 metric but are compared to reconstructed image predicted from a model trained with mean squared loss instead of SSIM Loss. Theoretically ideal difference image between a non-defective image and its reconstruction should not contain any highlighted regions. More highlighted regions in a difference image necessarily means poor reconstruction performance. We observed that highlighted intensity of Figure 5-3 and Figure 5-4 are both significantly lower than that of

corresponding Figure 7-3 and Figure 7-4. In other words, even for reconstruction of non-defective image, a model trained with SSIM Loss is more competent in predicting a reconstruction closer to the original image than a model trained with mean squared loss. This result demonstrates that model trained with SSIM Loss yields more robust prediction and consequently a better detection compared to model trained with mean squared loss.

	Baseline: SSIM Loss		Model 1: Mean Squared Loss	
Comparison Metric	SSIM Metric	L2 Metric	SSIM Metric	L2 Metric
F1 Score	0.72	**0.82**	**0.79**	0.74

Table 3: Classification result in F1 score of baseline model and model trained with MSE Loss. Higher F1 score for each comparison metric over different loss function is highlighted in bold

In theory, reconstruction capability is necessarily related to the classification performance. First step to determining whether the object in the image is defective or not is to compare the original image to the reconstruction. The magnitude of such difference obtained from the comparison is then used to perform the classification. As a result, a better reconstruction capability yields more accurate comparison and thus more reliable classification.

First, as compiled in Table 3, classification based on L2 comparison metric achieved higher F1 score for model trained with SSIM Loss. This result is consistent to the higher reconstruction performance of SSIM-guided training discussed earlier in this section. However, the result of classification does not completely correlate to the better reconstruction capability of model trained with SSIM Loss.I fact, as recorded in Table 3, classification based on SSIM metric yielded better performance for model trained with mean squared loss. Possible reason behind this finding is the presence of distortion in reconstructed images obtained from a model trained with mean squared loss. As discussed earlier in this section, Figure 6-2 exhibits high distortion at the defective region of the reconstructed image. Distortion is an outcome of relatively poor reconstruction performance of mean squared loss-guided model when given a defective test image. However, difference score is obtained by computing the magnitude of difference present between the reconstructed image and the original test image. In particular, SSIM metric seeks to output a comparison result based on features beyond the simple color difference. To the perspective of the SSIM metric, such distortion in reconstructed image is considered as a major difference that encompasses all the features that the metric evaluates. As a result, the presence of distortion in reconstructed image contributed to high difference score when given a defective test image.

Figure 4-4 and Figure 6-4 are SSIM difference images obtained by comparing the defective original test image to the reconstructed image obtained from a model trained with mean squared loss and SSIM loss, respectively. Although the quality of reconstruction from SSIM loss guided model is superior, due to the high presence of distortion, we observed misleadingly highlighted regions that falsely contribute to large difference scores. Granted, to

achieve better classification, we want a defective test image to have higher difference score than non-defective test image. However, we assume that any defect in the test image that deviates from the non-defective training images to be ignored and not reproduced in its reconstructed counterpart. Thus, if the high difference score is an outcome of a distortion in reconstructed image caused by the defect, then it defeats the underlying assumption of the methodology used in this project.

However, results obtained from L2 comparison metric does not seem to suffer from this phenomenon as much as the results from SSIM comparison metric. Figure 4-5 and Figure 6-5 are difference images obtained from L2 comparison metric instead of SSIM comparison metric. They are direct counterparts of Figure 4-4 and Figure 6-4, respectively. Consistent to the fact that L2 metric only captures the color difference, both Figure 4-5 and Figure 6-5 successfully identified the part of the defective region that contains stark color difference that deviates from the expected non-defective features learnt during training. However, we can observe that the intensity of the region highlighted by L2 difference is stronger in Figure 4-5 – which is obtained by comparing reconstructed image of satisfactory quality – than in Figure 6-5, obtained by comparing distorted reconstructed image. In fact, the distortion in the reconstructed image is caused due to the presence of the defect in the original test image. In other words, the distortion in the reconstructed image reflects the defect if the original test image to some extent. Consequently, the remnant of the defect reproduced in the reconstructed image caused weaker L2 difference. In the end, we can conclude that higher SSIM metric-based classification performance of model trained with mean squared loss should be considered trivial, while higher L2-metric based classification performance of model trained with SSIM loss is a result of superior reconstruction capability that prevented reproduction of the defect present in the original test image.

2. Effect of input handicap

As introduced in the Input Handicap section, we hypothesized that introducing an inconsistency to the input images during training would improve the robustness of the overall performance by preventing the innate inconsistencies that are present in the non-defective training images from being reproduced in reconstruction of test images. In fact, this speculation is grounded on the assumption that introducing an input handicap would not impair the reconstruction capability in the first place. However, according to the result obtained, adding an input handicap led to poor reconstruction capability.

Figure 4: Visuals of result from sample **defective test image** obtained from the **baseline model**.

Repeated for convenience

Figure 5: Visuals of result from sample **non-defective test image** obtained from the **baseline model**.

Repeated for convenience

Figure 8: Visuals of result from sample **defective test image** obtained from **model trained with input handicap**

From left to right, each sub-figure is labeled Figure 8-1, 8-2, 8-3, 8-4, and 8-5, respectively

Figure 9: Visuals of result from sample **non-defective test image** obtained from **model trained with input handicap**

From left to right, each sub-figure is labeled Figure 9-1, 9-2, 9-3, and 9-4, respectively

Second sub-figures of Figure 4 and Figure 8 (Figure 4-2 and Figure 8-2) are the reconstructed counterparts of the same defective bottle image, each from a model trained without and with the input handicap, respectively. In Figure 8-2, we can observe that the reconstructed image from a model trained with input handicap contains a noticeable distortion, which is not present in the reconstructed image from the baseline model, as can be seen in Figure 4-2. This result effectively defeats the underlying assumption of our speculation. In fact, to some extent, the performance of classification was consistent to the reduced reconstruction capability of the model trained with input handicap.

	Baseline: Without Input Handicap		Model 2: Input Handicap Applied	
Comparison Metric	SSIM Metric	L2 Metric	SSIM Metric	L2 Metric
F1 Score	**0.72**	0.82	0.69	**0.83**

Table 4: Classification result in F1 score of baseline model and model trained with Input Handicaps. Higher F1 score for each comparison metric over different loss function is highlighted in bold

As recorded in Table 4, baseline model trained without the input handicap achieved slightly higher SSIM metric based classification F1 score than the model trained with input handicap. In the preceding section about the experiment result regarding the effect of varying loss function, we discussed that presence of distortion in reconstructed image can misleadingly improve the classification result obtained from SSIM comparison metric. However, the distortions created by the model trained with input handicaps were not as extreme as distortions created by model trained with mean squared error. In other words, distortions from adding input handicaps are not drastic enough to cause high SSIM difference score. Thus, we concluded that better SSIM metric based classification performance of model without input handicap is not trivial and is direct consequence of better reconstruction capability. On the other hand, classification result obtained from L2 comparison metric was higher in model trained with the input handicap, but in extremely trivial amount, which is likely due to the randomness of sampling the test image. In fact, the difference of reconstruction capabilities between baseline model and Model 2 is not as extreme as that between the baseline model and Model 1. Thus we concluded that unlike the SSIM comparison metric, L2 comparison metric is not sensitive enough to accurately reflect the slightly superior reconstruction capability of the baseline model.

Conclusion

In this experiment, we investigated how implementing SSIM as an alternative to L2 metric yield different results. When used as loss function, model trained with SSIM loss yielded significantly better reconstruction capabilities and consequently detection performance than a model trained with per-pixel L2 loss. Consequently, when used as a comparison metric to compare the original test images and its reconstructed counterparts, SSIM metric was more sensitive and penalizing to even a slightest difference that a conventional L2 comparison metric couldn't detect. In fact, SSIM comparison even penalized the minor reconstruction errors that are not part of the defect. As a result, classification result obtained from L2 comparison metric was generally better than the classification result obtained from SSIM comparison metric. However, we cannot hastily conclude that L2 comparison metric is better than SSIM comparison metric. As sensitive and penalizing it is, SSIM metric can be considered to be more reliable and accurate in evaluating the difference between two images.

For the input handicap, the experiment failed to bear fruitful result. In fact, adding input handicap for the training undermined the reconstruction capability, which was against our speculation that it would in fact improve the overall performance. One possible reason behind this result could be that the handicap we introduced to the input training images was too extreme and effectively contaminated the actual features that the model was supposed to learn. In future work, experimenting with different intensities and types of the handicap could possibly yield more insightful findings.

References

1. Mattson, M. Superior pattern processing is the essence of the evolved human brain, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4141622/

2. Livne, Michelle, Rieger, Jana, Aydin, Utku, O., . . . I., V. A U-Net Deep Learning Framework for High Performance Vessel Segmentation in Patients With Cerebrovascular Disease, from https://www.frontiersin.org/articles/10.3389/fnins.2019.00097/full

3. Paul, S.. Convolutional denoising autoencoder for images, from https://srinjaypaul.github.io/Convolutional_autoencoders_for_images/

4. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli.Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing,13(4):600–612, 2004.

Appendix

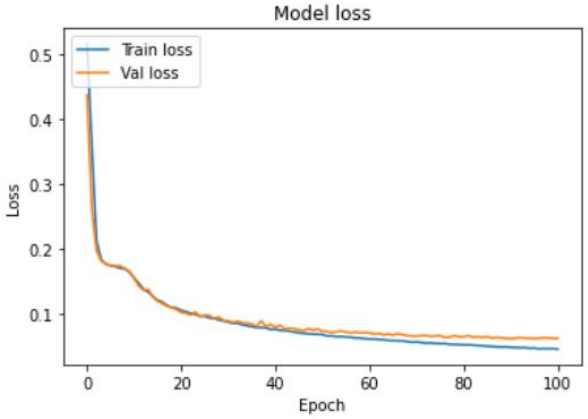

Figure 10: Loss curves of baseline model

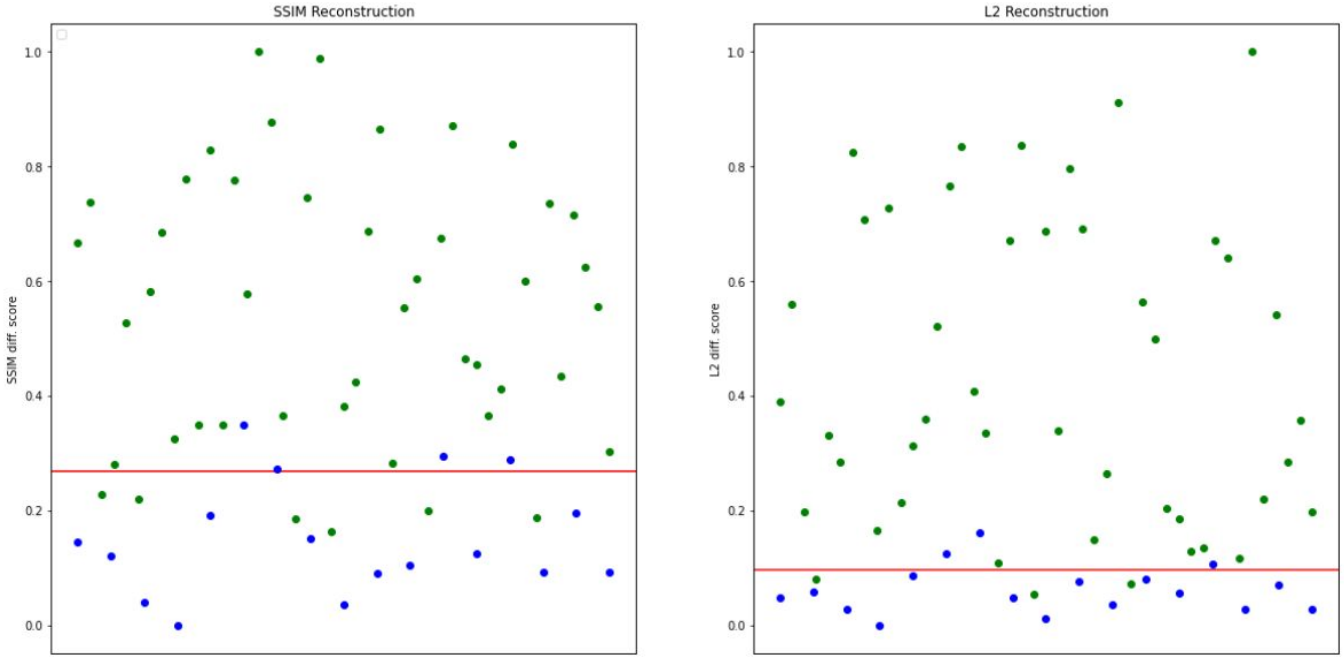

Figure 11: Visual classification result of test images obtained from baseline model using SSIM difference metric (Left) and L2 comparison metric (Right)
Blue dots represent non-defective test images and Green dots represent defective test image

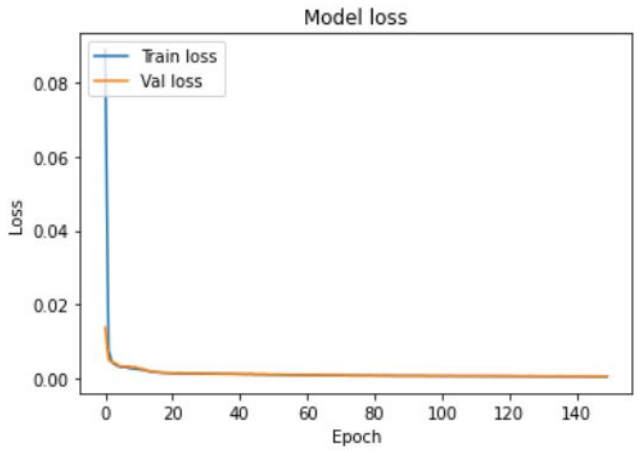

Figure 12: Loss curves of Model 1

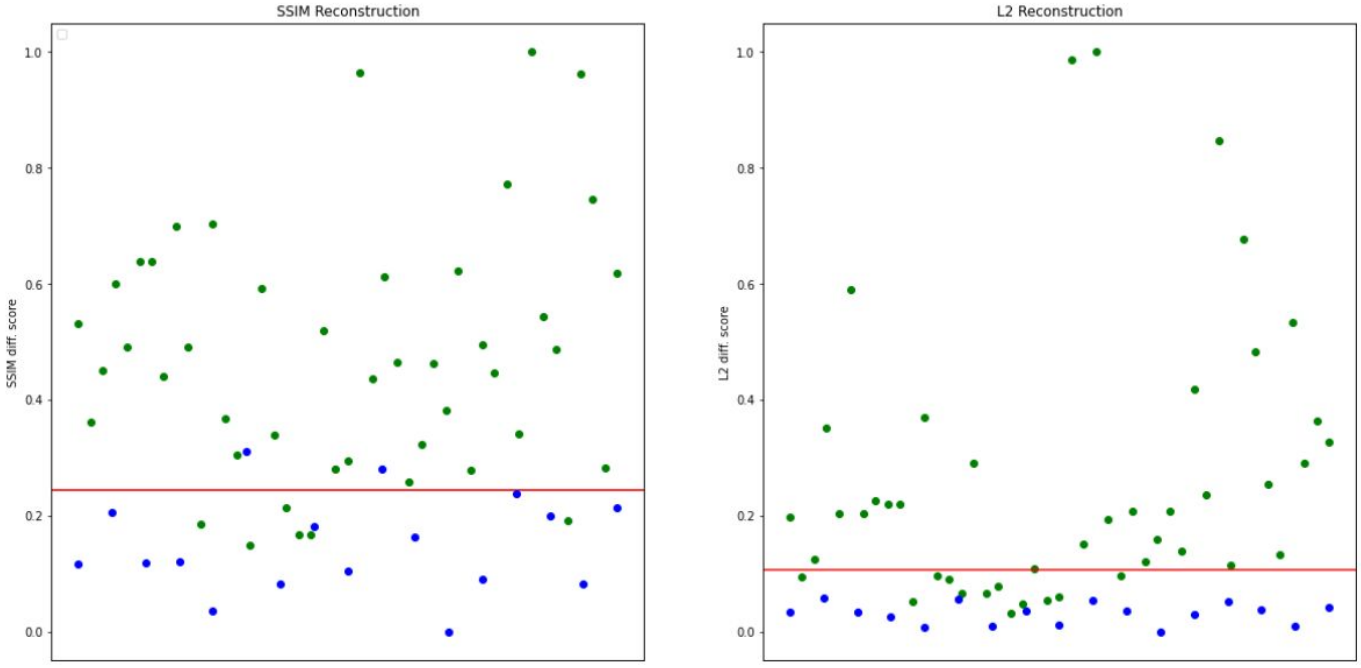

Figure 13: Visual classification result of test images obtained from Model 1 using
SSIM difference metric (Left) and L2 comparison metric (Right)
Blue dots represent non-defective test images and Green dots represent defective test image

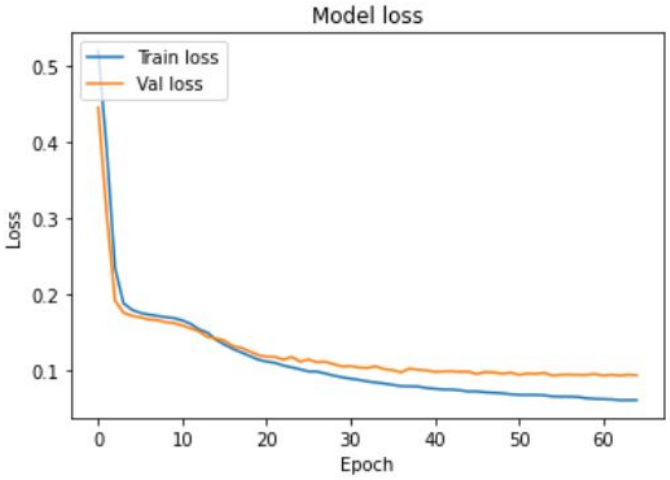

Figure 14: Loss curves of Model 2

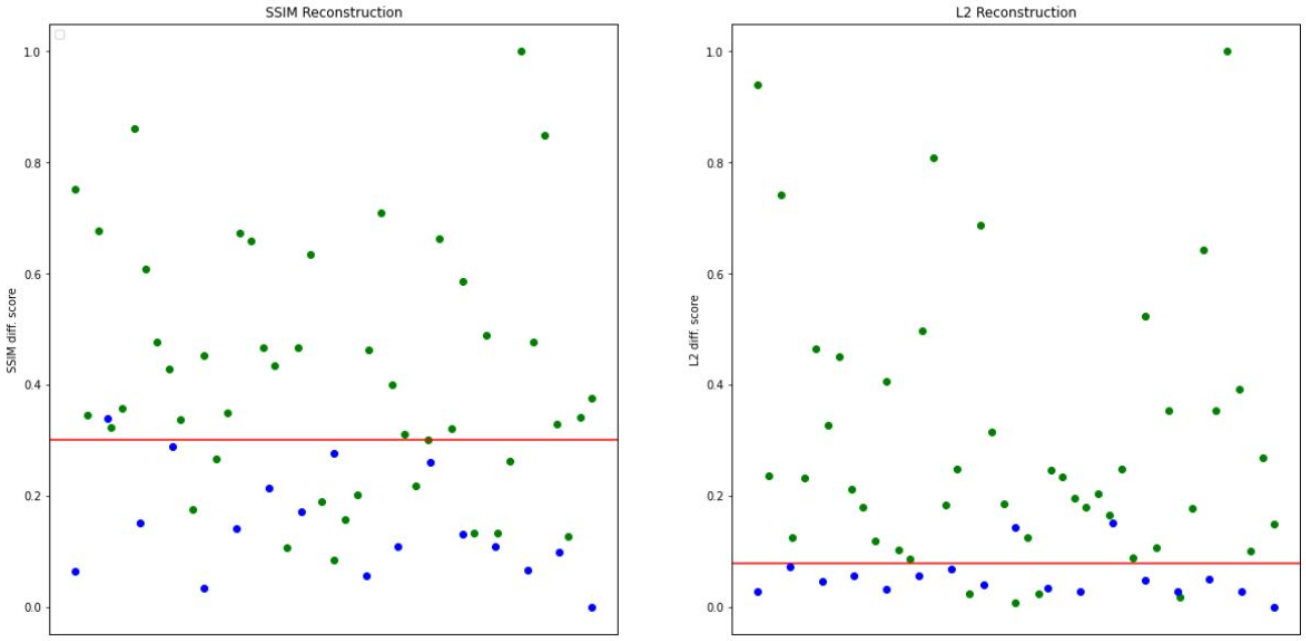

Figure 15: Visual classification result of test images obtained from Model 2 using
SSIM difference metric (Left) and L2 comparison metric (Right)
Blue dots represent non-defective test images and Green dots represent defective test image

Presentation

https://youtu.be/__y4-nH4hlU

Division of Labor

Report

Abstract: Taek Jung Kim
Introduction: Taek Jung Kim
Dataset: Taek Jung Kim
Augmentation: Taek Jung Kim
Preprocessing: Taek Jung Kim
Convolutional Autoencoder: Taek Jung Kim
Architecture: Taek Jung Kim
Structural Similarity: Juyoung Bae
Input Handicap: Juyoung Bae
Experiment-Training Phase: Juyoung Bae
Experiment-Evaluation Phase: Juyoung Bae
Result- Choice of Loss Function: Juyoung Bae
Result- Effect of Input Handicap: Juyoung Bae
Conclusion: Juyoung Bae

Code

Initial Backbone & Debugging: Taek Jung Kim
Refactoring & Improvement: Juyoung Bae