Synthesizing Hard Training Data From Latent Hierarchical Representations

Høj, Benjamin Jendresen; Møgelmose, Andreas

Published in:
Scandinavian Conference on Image Analysis 2023

Publication date:
2023

Document Version
Accepteret manuscript, peer-review version

Link to publication from Aalborg University

Citation for published version (APA):
Høj, B. J., & Møgelmose, A. (2023). Synthesizing Hard Training Data From Latent Hierarchical Representations. I *Scandinavian Conference on Image Analysis 2023*

Synthesizing Hard Training Data From Latent Hierarchical Representations

Benjamin J. Høj and Andreas Møgelmose[0000−0003−0328−382X]

Visual Analysis and Perception Lab, Aalborg University, Denmark
{bejeho,anmo}@create.aau.dk

Abstract. This paper introduces a framework for creating augmented hard samples, which are new images created to resemble those that a classifier will struggle to classify. This is used for data from an automatic visual defect inspection system, specifically images of vials with and without chipped glass. The hard samples were found by training ConvNeXt classifiers and using the confidences of the classifiers on the training dataset. VQ-VAE2 was used to obtain the latent representations of the hard samples, and a PixelSnail model was used to create new high-frequency details while retaining low-frequency details. The PixelSnail model was pre-trained on a large amount of non-defect images. The augmentation method was applied to a dataset of 200 images and was evaluated by training classifiers to test the effect of using the augmented hard samples. Introducing the augmented hard samples into the dataset was found to improve classifier performance, measured in Area Under Curve (AUC) of the Receiver Operator Characteristic (ROC) curve, from 0.953 to 0.973. The method was tested with augmenting both defect and non-defect images, and the somewhat surprising conclusion is that while using augmented defect images didn't yield improvements, augmenting non-defect images did.

Keywords: Machine Learning · Automatic Visual Inspection · Data Augmentation.

1 Introduction

Manufactured goods can in some cases have certain defects that makes them unsuitable for use. Finding the defective products before they're shipped is particularly important in the pharmaceutical industry as a defective product can cause harm to the patient. Some of these inspection tasks are today performed by humans, but humans, in addition to often finding the job tiresome, might not discover all defects, which means an automatic visual inspection system might be able to improve the detection rates. Some defects can be quite rare and does therefore not yield a lot of data for training a deep learning model and results in a significant class imbalance between defect and non-defect images. Manufacturing these defects can be both expensive to do and hard to make representative of real defects.

This paper presents an approach to synthesizing images of hard samples for an automatic visual inspection system in order to improve classifier performance when trained on an imbalanced dataset with a limited amount of samples of one class. Specifically, we introduce altered images of glass vials used for medicine, as seen in Figure 1, into a classifier training set. This is done to allow classifiers to train on more images with features that they tend to misclassify. Figure 2 shows the overall process of creating new samples from a training set using the proposed method. The process uses the hiearchical latent representation of the VQ-VAE2 architecture from [9] for augmenting hard samples. We define hard samples as training samples with a high loss. These augmented samples are then added to the training set, and a new iteration begins.

This work presents two contributions. The first is a method for creating augmented images using a hierarchical latent space. The method utilizes the large amount of non-defect data to create images, the augmented hard samples, that will allow classifiers trained on them to improve their performance as described in Section 5. The second is how this method can be implemented in an iterative process to improve performance by continuously adding these synthesized training samples. The implementation in this paper can be used for datasets with large class imbalances with few pictures of one class.[1]

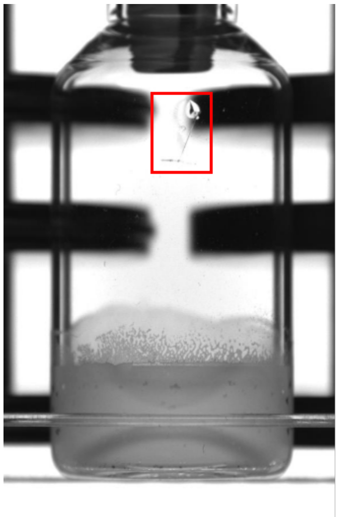

Fig. 1. Medicinal glass bottle (vial) with a chip highlighted in a red square.

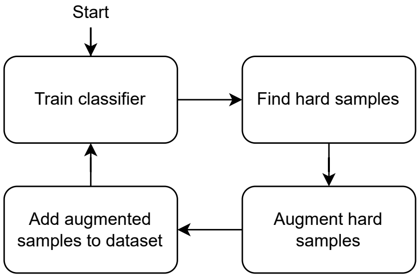

Fig. 2. Overview of the iterative augmentation process.

[1] The code for this project is available at https://github.com/xxxxxx

2 Related Research

Augmentation methods are often used in order to improve the performance of deep learning models. This project has a focus on creating new details in images for dataset augmentation as a complement to classical augmentation, and this section will therefore give an overview of other methods that have done similar work. Many algorithms [1, 11, 8, 12, 10] use some form of GAN for generating/augmenting data. Some [4, 2, 3] use classical methods but attempts to find policies for which augmentations to use and for how augmented the images should be. Others [5, 6] combine multiple samples in order to create a new training sample. These methods will be described in further detail below.

Antreas Antoniou et al. [1] proposed a Generative Adversarial Network (GAN) model for data augmentation. The model obtains a latent representation for an image and combines it with a sample from a standard Gaussian distribution, which is passed through a generator to obtain a new sample. Jun-Yan Zhu and Taesung Park et al. [11] propose the CycleGAN architecture, which use 2 GANs that transforms images between two domains, for example switching winter landscapes to summer landscapes or vice versa, using unpaired images for training. In [8], Guim Perarnau et al. propose an architecture that contains two encoders for obtaining a latent representation of an image and an attribute vector. A conditional GAN is then trained on both vectors, and an image can then be edited by Yezi Zhu et al [12] created a framework for synthesizing training images where a conditional GAN takes a mask as input and then fills out the mask. In [10] by Simon Vandenhende et al., a conditional GAN was trained in order to create hard samples. The GAN had, in addition to its discriminator, also a classifier. The GAN was trained towards making samples that were hard for the classifier to classify, but its images were kept realistic by the discriminator.

Ting-Yao Hu[4] created an augmentation framework that ranks the samples based on their loss and uses more significant augmentations on samples with low loss. [2] by Ekin D. Cubuk et al. presents a reinforcement learning approach where the model learns to predict an augmentation policy. The policy include probabilities for using different augmentations and their magnitudes. The policies are learned by assessing the validation accuracy of a trained model using the augmentation policies. In [3] by Chengyue Gong et al., saliency maps are used to guide classical augmentation methods such as cutout, so that they don't remove the most important high-frequency details for the classification.

In [5], Lemley et al. uses a network to learn how to generate an augmented image by combining multiple images of the same class. This is done using the error of a classifier and a loss function that causes the output of the generator to be similar to other images of that class. Han Liu et al. [6] uses a VQ-VAE to interpolate between two samples in the latent representation. They do this by obtaining the latent representation in two medical images of different medical conditions. These are then linearly combined and passed through the decoder. To obtain realistic images for the interpolation points, they train the generator to generate realistic images for combined latent representation.

There are many methods for data augmentation, including several that combine images to obtain new images or others that looks into how to best apply classical augmentation. To the best of the authors' knowledge, no other research has been done to augment images by obtaining a hierarchical latent representation and then augmenting by resampling a shallow representation to obtain new high-frequency details. The work in this paper is based on the VQ-VAE2 model [9] by Ali Razavi, Aäron van den Oord and Oriol Vinyals, and we opted to go with VQ-VAE2 rather than GANs due to more promising results in preliminary testing.

3 Data

The data used in this project consists of medicinal vials with freeze-dried product. There are many different kinds of defects that can happen on the production line, such as plastic particles in the product, loose packaging components, etc. This project focuses on vials with chipped glass due to it being one of the defects with the most data for testing, which means more data can be used for evaluating the performance of the proposed method. Our dataset contains 66 defect vials with 30 images of each vial as it rotates, each with a resolution of 576x892. The chips aren't visible in all of the images, so the ones showing no defects were removed. The 66 vials were randomly split up into 8 for training (100 images), 10 for validation (170 images) and 48 for testing (833 images). We deliberately limit the amount of training data to resemble the training data quantities for defects with less data. There are 250k images from vials without defects for training and 30k for validation and testing, so images were randomly sampled to have a 1:1 ratio of images with defects to images without for all defect images. The test and validation sets will remain static. Due to the large class imbalance, the non-defect images will be used for pre-training as both classes are identical except for the chipped area of the glass. An example of an image with a chip can be seen in Figure 1. The defect vials used were chipped by hand (again due to the real defects being exceedingly rare), but were judged to be realistic by a human inspector. This does not impact the academic evaluation of the proposed method as all data came from the same population and were randomly split into training, validation and testing sets.

4 Choosing and Augmenting Samples

This section will cover the implementation and details of the proposed solution. Section 4.1 will go into the framework for implementing the solution and selecting images to augment. Section 4.2 will then go into the VQ-VAE2 and PixelSnail implementation for the augmentation.

4.1 Framework

The implementation followed four repeating steps:

1. Train 20 classifiers
2. Find images in the training set with the highest loss across classifiers
3. Synthesize images from the hard samples found in step 2
4. Augment dataset with the synthesized images

The ConvNeXt tiny architecture presented by Zhuang Liu et al in [7] was used as the classifier due to having a low inference time, which is good for training many classifiers, and due to high performance. Its output was softmaxed for obtaining confidences. It was trained for 150 epochs using Stochastic Gradient Descent with a learning rate of 0.001, momentum of 0.9 and a batch size of 32. Hard samples were found by using the confidences of the trained classifiers on the training set and then choosing the samples with the lowest confidences in the true label. The chosen samples would be augmented with new high-frequency features, added to the training dataset, and the process would then repeat. Two synthesized images are added to the dataset per iteration, chosen due to having two GPUs for synthesizing images. All models in this project were trained by using the images resized to 512x512.

4.2 Synthesizing Augmented Images

Creating new samples was done by using the VQ-VAE2 and PixelSnail models using the implementation from Kim Seonghyeon.[2] The VQ-VAE2 architecture uses two encoders in order to obtain two latent images, one for global structure of the image (structural map) and another for more local details (local map). The pixels of these latent images consists of vectors of dimension 64. The local map have a size of 128x128 while the structural map have a size of 64x64. There is a codebook of learned vectors for each latent space, and the vectors in the latent images are then mapped to the nearest vector in the codebook measured in L2-space. The new latent images, with each pixel vector replaced by the nearest codebook vector, will then be passed to the decoders, which will then recreate the image. The architecture has three loss terms. The first is a simple recreation loss which in the original VQ-VAE2 paper uses the L2 loss, but in this paper the L1 loss is used instead and was found to give a lower recreation loss than the L2. The second loss function is using exponential moving averages to learn the codebook vectors, which will get closer to the encoder outputs. Lastly, there is an encoder loss term, which trains the encoders to output vectors closer to those in the codebook by using the L2 loss between them.

Sampling from the VQ-VAE2 architecture was done using a PixelSnail autoregressive model like in the original VQ-VAE2 paper. The PixelSnail model in this project would only sample the local map and would be conditioned on the structural map. This allows the model to use the structural map to recreate the image but with new high-frequency details. An augmented image can be seen in Figure 3, which also shows the augmentation process.

[2] Code at https://github.com/rosinality/vq-vae-2-pytorch

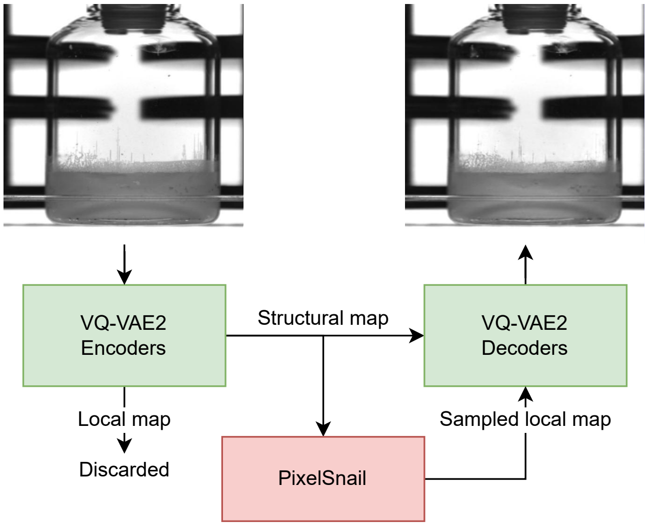

Fig. 3. The augmentation process. The VQ-VAE2 latent representations are acquired. The local map is then discarded, and another conditioned on the structural map is created. The augmented image is then created using these latent representations.

5 Evaluation

The proposed augmentation was assessed by training classifiers as the quality of the dataset is tied to the performance of models trained on it. The Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) curve was used as the metric for this performance, where higher is better. For each iteration of the training set, 20 different classifiers were trained to get a more stable performance metric and to select hard samples. The final metric to evaluate the dataset at that iteration is then the mean of the AUCs for the 20 models. A baseline mean AUC was calculated using the defect images from the classifier training set and randomly sampling from the 250k non-defect images to utilize the large amount of data available. Sampling from the 250k data was done every time a non-defect sample was loaded such that the model was trained with as much varied data as possible. This yielded a mean AUC of 0.953 compared to a mean AUC of 0.951 when trained on only the training dataset of 100 defect images and 100 non-defect images. 0.953 was chosen as the baseline result to beat due to being the best AUC obtained by simple training using the data available.

Two experiments were done using the same trained VQ-VAE2 and PixelSnail model. The training is described in Section 5.1. Though two images were sampled for each iteration, only the loss was used for finding hard samples rather than choosing one image from each class, which would keep class balance. This was done to focus more on the images that the classifiers struggled to correctly classify. The first experiment augmented the two hardest samples regardless of them being defect or non-defect images and is detailed in Section 5.2. The second experiment in Section 5.3 only augmented hard non-defect images.

5.1 Model Training

The VQ-VAE was first trained on the 250k good images for 11 epochs such that it would learn to recreate a large amount of vial variance. It was then trained for 258 epochs on the classifier training dataset (see Section 3) where training was terminated once the recreation loss started stagnating. The PixelSnail model was trained for 3 epochs on the 250k images before it was trained on the classifier training dataset for 112 epochs, where overfitting started after 28 epochs. Training the PixelSnail on the 250k non-defect images was shown in a test to be important as it would otherwise fail to produce an image without severe artifacts. As the augmented image should retain a high amount of the same features as the unaugmented image, we opted to use an overfitted model. The PixelSnail architecture allows for sampling only an area of the latent map, so an area with a chip was resampled for each epoch and qualitatively inspected. The training parameters after 66 epochs were chosen as it would preserve most of the features with small variations and was therefore chosen for the augmentation process.

5.2 Augmenting Both Classes

For this test, 15 dataset iterations were done. The mean AUC for each iteration can be seen in Figure 4. It can be seen that the mean AUCs have a general increasing trend for the iterations but that some of the iterations also include some dips, particularly iteration 14 that had a mean AUC lower than what was found for the initial dataset. The augmented images included in the training dataset for this iteration were both defect images and also showcases how drastic an effect few images can have on classifier performance. The highest AUC obtained was 0.971 at iteration 13, which is an increase of 0.18. In most of the iterations that improved the AUC, non-defect images were augmented, which lead to the experiment in Section 5.3 where only non-defect images were considered for augmentation. A likely reason for decrease in performance when creating augmented defect images is that the defect images chosen for augmentation had very small or barely visible chips, meaning that small changes could make them no longer resemble actual chips. The framework also had a tendency to reaugment already augmented defect images, likely for this reason. As a test, the images added to the datasets for iterations 2, 3, 5, 11 and 13 were manually selected and added to the original dataset for training 20 new classifiers. These were selected as their augmented images had improved the mean AUC compared to all previous iterations. The images consist of 3 augmented defect images and 7 augmented non-defect images, and was done in order to see how models trained on these would perform. The experiment yielded a mean AUC of 0.966 and was compared to a similar experiment but without the augmented defect images. This gave the same mean AUC of 0.966, indicating that the augmented defect images do not improve the classifier performance.

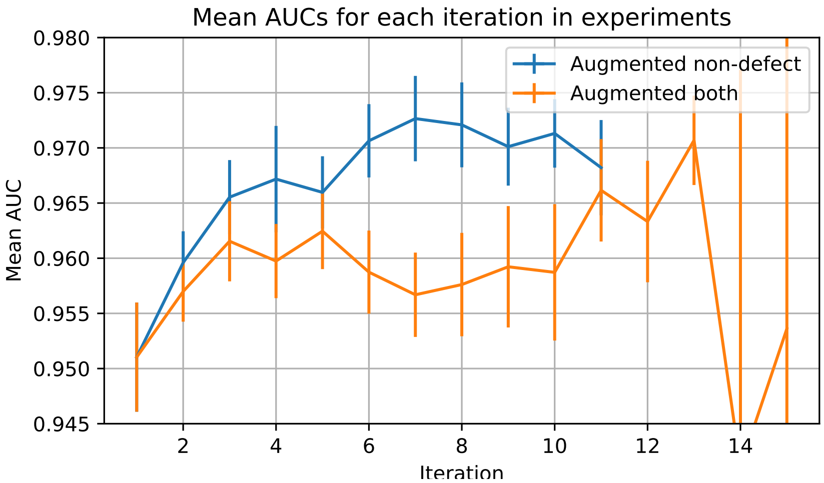

Fig. 4. Mean AUCs for both experiments. For the first iteration, no new images were introduced into the dataset. The 95% confidence interval is shown for each iteration. The last two iterations for the graph, with both classes augmented, contain outlier AUCs, resulting in a very large confidence interval.

5.3 Augmenting Non-defect Class

This test was done due to the results from Section 5.2 showing that the augmented defect images didn't improve classifier performance and was run for 11 iterations due to time constraints. The graph in Figure 4 shows that the mean AUC rose higher than it did when both classes were considered for augmentation. The highest mean AUC found was at iteration 7, which yielded a mean AUC of 0.973, a clear improvement from the 0.953 baseline. Another experiment was made in order to see if the improvement would also happen by simply copying the all the unaugmented versions of the images in the iteration that gave a mean AUC of 0.973 to the dataset. This means that if image A had been augmented twice up to iteration 7, then the classifier would see this same image thrice in one epoch. The result was a mean AUC of 0.963, which is an improvement over the baseline by 0.01 but only half the improvement from introducing the augmented images into the dataset.

6 Discussion

This paper has presented a method for going into the latent space using the VQ-VAE2 architecture and augmenting an image therein using a PixelSnail model. The training data used were 250k non-defect images and 100 defect images. A test was done to see if the initial training on the 250k non-defect images were important, but without first training on these images, the model failed to augment images without very significant artifacts. A shortcoming of the model is that it requires a human to subjectively decide which PixelSnail checkpoint to use for generating local information, which might not correspond to the best checkpoint that would create the biggest dataset improvements. The experiment augmenting samples from both classes showed that augmented hard defect images didn't improve the mean AUC. This could be because the PixelSnail model was pretrained on so many non-defect images that it could use that prior knowledge to augment non-defect images properly but didn't have enough prior knowledge on chips. Despite the shortcoming on creating augmented images with chips, the method was successfully used to increase the mean AUC of the trained model to 0.973 from 0.953 by constraining the augmentation procedure to non-defect images. The work done also shows that rather than training on many varied images, such as when establishing the baseline, it can be better to augment selected images or train more on some, as shown in Section 5.3. This could mean that it could potentially be applied to bigger datasets as well to improve performance of classifiers.

7 Future Work

The main challenge of the proposed method is that it's unclear when to stop the overfitting for the PixelSnail as it is currently a subjective choice and might not reflect what would best increase the classifier performance. Additionally, there is no metric used for evaluating the created images except for the performance on the training when added to the training set. As some images seemingly lower the model accuracy, the model could likely benefit from some kind of filter to decide on whether to use or discard a synthesized image.

8 Conclusion

This work introduces a method for utilizing the VQ-VAE2 architecture for creating augmented images that can be introduced into a training set for increasing performance. Specifically, the method leverages the inherent split between high-frequency and low-frequency details, such that the general shape of training images is maintained, while the details will be changed. This helps steer the process away from too drastic changes. A baseline without this dataset augmentation delivered a mean AUC of 0.953, improved by our method to 0.973 when applied to a rather small classifier training dataset with 2 classes, each with 100 images. This makes the method highly suitable for cases where only a few pictures from one class can be obtained. It is worth noting, though, that the method requires a significant number of samples for training the PixelSnail model to properly augment the images without inducing significant artifacts and thus requires a significant number of samples of the augmented class.

References

1. Antoniou, A., Storkey, A.J., Edwards, H.: Data augmentation generative adversarial networks. ArXiv **abs/1711.04340** (2017)
2. Cubuk, E.D., Zoph, B., Mané, D., Vasudevan, V., Le, Q.V.: Autoaugment: Learning augmentation policies from data. ArXiv **abs/1805.09501** (2018)
3. Gong, C., Wang, D., Li, M., Chandra, V., Liu, Q.: Keepaugment: A simple information-preserving data augmentation approach. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1055–1064 (2021)
4. yao Hu, T., Shrivastava, A., Chang, J.H.R., Koppula, H.S., Braun, S., Hwang, K., Kalinli, O., Tuzel, O.: Sapaugment: Learning a sample adaptive policy for data augmentation. ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) pp. 4040–4044 (2021)
5. Lemley, J., Bazrafkan, S., Corcoran, P.M.: Smart augmentation learning an optimal data augmentation strategy. IEEE Access **5**, 5858–5869 (2017)
6. Liu, H., Zhao, Z., Chen, X., Yu, R., She, Q.: Using the vq-vae to improve the recognition of abnormalities in short-duration 12-lead electrocardiogram records. Computer Methods and Programs in Biomedicine **196**, 105639 (2020). https://doi.org/https://doi.org/10.1016/j.cmpb.2020.105639, https://www.sciencedirect.com/science/article/pii/S0169260720314723
7. Liu, Z., Mao, H., Wu, C., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s (2022)
8. Perarnau, G., van de Weijer, J., Raducanu, B., Álvarez, J.M.: Invertible conditional gans for image editing. ArXiv **abs/1611.06355** (2016)
9. Razavi, A., van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with vq-vae-2. ArXiv **abs/1906.00446** (2019)
10. Vandenhende, S., Brabandere, B.D., Neven, D., Gool, L.V.: A three-player gan: Generating hard samples to improve classification networks. 2019 16th International Conference on Machine Vision Applications (MVA) pp. 1–6 (2019)
11. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. 2017 IEEE International Conference on Computer Vision (ICCV) pp. 2242–2251 (2017)
12. Zhu, Y., Aoun, M., Krijn, M., Vanschoren, J.: Data augmentation using conditional generative adversarial networks for leaf counting in arabidopsis plants. In: BMVC (2018)