Deep Learning based Optical Inspection with Centralized Analysis for High Volume Smart Manufacturing

Guoyi Li, Chao Feng, Addishiwot Woldesenbet, Bruce King, Hamid Hadavi, Vikasini Moku, Kurt Loken and Gary Kunkel

Seagate Technology, Bloomington, Minnesota, United States

{guoyi.li, chao.j.feng, addis.woldesenbet, bruce.king, hamid.hadavi, vikasini.moku, kurtis.d.loken, gary.j.kunkel}@seagate.com

ABSTRACT

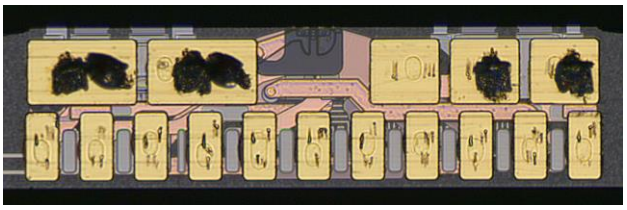
Increased capabilities in data storage and exploration provide significant insights for quality assurance in a high volume manufacturing environment. However, these opportunities are associated with great challenges in analytical model development, application deployment, system throughput and reliability assurance. While no commercial software system fully meets the needs of recording head factories in Seagate, a novel strategy named optical inspection with centralized analysis (OPICA) has been developed to detect defects of trailing edges of the recording heads of hard disk drives, and fail the parts when necessary.

Leveraging the robust state-of-the-art artificial intelligence technologies, a deep learning based semantic segmentation engine is built using convolutional neural networks for optical inspection. It has shown an improved accuracy to that of visual inspection performed by human. Meanwhile, a high performance computation engine has been built as a Kubernetes cluster with multiple GPU and CPU units. It is able to achieve the target throughput of three million high-resolution images in each day (i.e., 12 TB image data and 35 images per second). With the high fidelity offered by a Kubernetes cluster, the developed applications (image inference engine, preprocessor, postprocessor, etc.) serve as containerized microservices independently. Such an architecture ensures the vertical and horizontal scalabilities according to the computation of each individual deployment, while all deployments communicate through an Advanced Message Queuing Protocol (AMQP) cluster without human interference. This analytic framework enables Industry 4.0 recording head manufacturing by integrating advanced AI technologies with a robust edge computation architecture.

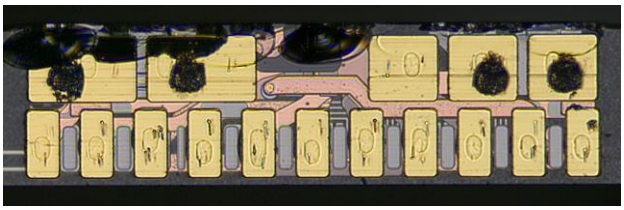
1. INTRODUCTION

The exponentially increasing demands of device storage in industries such as communications, cloud computation and autonomous vehicle provide a unique opportunity for high volume hard drive manufacturing (Yang et al., 2017; Ramanujam, 2018). As a hard drive manufacturer, the assurance of product quality becomes ubiquitously critical and challenging due to the boosted productivity on a daily base. Therefore, the methodologies and best practices to achieve an effective inspection system for high volume manufacturing has attracted more and more attention in the past decades (Sadeghian, Koster & van den Dool, 2013; Bonam al., 2016). As an example, the recording heads factory at Seagate requires a peak throughput of approximately three million images of the recording heads per day, which greatly challenges the traditionally developed metrology and analysis.

In this paper, as no commercial software or system fully meets the needs of availability and flexibility, we present the development of a novel optical inspection system in a high volume smart manufacturing, named optical inspection with centralized analysis (OPICA). This system is designed for defect detection and classification of trailing edges (TE) of the recording heads in hard disk drives (HDDs). For a demonstrative purpose, Figure 1 presents two optical TE images captured by high-speed cameras, and the dimensions of each image are 1591×491×3 in pixel with a size of approximate 2.2 MB in bitmap (BMP) format. Figure 1(a) refers to a qualified image that is expected to go through next inspection/test stage. The image in Figure 1(b), on the other hand, presents an image with one type of defect (details of defects will be discussed in the later sections). It should be noted that the specifications of defects vary in different images with one or more defects. More importantly, the appearance of defect(s) does not necessarily indicate the failure or rejection of the particular image. Instead, the decision of rejection or acceptance for an image depends on if the specifications of the defect (i.e., types, sizes and locations) exceed defined criteria/thresholds.



(a)



(b)

Figure 1. Optical images of TE in recording head including (a) accepted image and (b) defected image.

For analyzing the TE optical images, two strategies have been considered including rule-based computer vision approaches and machine learning based models. Rule-based approach generally involves the analysis of image specifications such as RGB colors, contrast, sharpness, spatial configurations, etc., with the empirical knowledge to make decisions (Khan, 2014). The machine learning based approach relies on building a trainable model to make predictions based on the available training examples without empirical knowledge. Specially, for analyzing the target high-resolution images, the effectiveness of convolutional neural networks (CNNs) has been widely proven, since the convolutional filters are specially suitable for compressing the information in the images (Krizhevsky, Sutskever & Hinton, 2012; Zhang et al., 2015). However, either rule-based or machine learning based approach could not fully meet the mentioned requirement of the desired inspection system. First, rule-based inspection strategies are limited by the requirement of domain knowledge including computer vision, which might not be realistic in a factory environment. On the other hand, as the evolutions of product and inspection criteria, the machine learning models, especially the deep learning based CNNs, will consume significant amount of time and labor including retraining, hyperparameter tuning, image labelling, model validation, etc. In addition, the number of examples with defects may not be sufficient for training the model to provide decisions with an acceptable accuracy. In order to overcome these challenges, the OPICA system is designed to be a combination of CNNs model and rule-based classifier. The CNN based semantic segmentation model works as a computer vision engine. It provides pixel-wise labels for the optical images (Moeskops et al., 2016) without making predictions for the final decisions (rejection/acceptance). Based on the predicted masks, the rule-based algorithm could incorporate the defined inspection criteria, and provide the predicted decision on each image.

The next challenge is how to deploy the trained deep learning model as a factory tool. A factory workstation could not meet the required throughput, while the trained CNN model has additional supportive components such as the rule-based classifier. It indicates that the inspection system needs functional, vertical and horizontal scalabilities (Bernstein, 2014) in the computation hardware. Moreover, the communications between subcomponents of the system should be automatic, which requires a robust message queuing system (Vinoski, 2006). In order to achieve these, a high performance computation engine is built as a Kubernetes cluster with multiple GPU and CPU units. In the cluster, the concept of application containerization is applied, which is a virtualization method used to deploy and execute services without launching an entire virtual machine for each application (Scheepers, 2014). Therefore, the number of replicated applicable pods (i.e., the number of replicas in a ReplicaSet) could be scaled for each application based on its need of computation power and corresponding hardware limitations. With this computation architecture, the developed applications such as image segmentation engine, preprocessor, postprocessor and rule-base classifier are containerized as Docker containers and running as multiple microservices independently. Their communications are built via a high availability RabbitMQ cluster without human interference, which supports the Advanced Message Queuing Protocol (AMQP).

The rest of this paper is organized as follows. Section 2 introduces the selection and evaluation of the deep learning CNNs that used to build the inference engine, followed by a discussion of currently achieved model's accuracy. Then, the computation architecture and corresponding deployment strategy are introduced in section 3 including the current system level throughput. A summary with key observations are presented in section 4.

2. TRAINING AND EVALUATION OF DEEP LEARNING MODELS

This section introduces the detailed investigation on the performance of the chosen state-of-the-art CNNs as semantic segmentation engines for the defect inspection in TE images. Then, leveraging the widely accepted model assessment strategy and the demands of the factory, the customized evaluation metrics are introduced in details followed by corresponding results and observations.

2.1. Investigation of convolutional neural networks

CNN based image segmentation has been applied to an extensive range of applications and its effectiveness has been widely proven (Minaee et al., 2020). An image segmentation model generally has two main neural components, encoder and decoder. The encoder compresses the input information into a latent space, while the decoder decompress the information to a defined size with pixel-wise labels. For the

application of defect detection, semantic image segmentation is a suitable candidate because it provides prediction for each pixel with aspects of defect type (Huang, 2018). This research considers three widely accepted neural networks in recent years to explore their effectiveness on TE defect detection, including DeepLab, Mask R-CNN and SegNet. DeepLab (Chen et al., 2017) is a hybrid neural network system combines deep convolutional nets, atrous convolution and fully connected conditional random fields. Based on the authors' observations and the consideration of computation efficiency, the CNNs of DeepLab system in this research is built based on the ResNet-50 (He et al., 2016). The Mask R-CNN (He et al., 2017) is an extension of Fast R-CNN (Girshick, 2015) with adding the functionality of predicting the object mask with a bounding box recognition, which is a unique feature in the family of CNN based semantic image segmentation. In addition, the SegNet CNN was developed by Badrinarayanan, Kendall, and Cipolla, R. (2015), whose encoder part was built based on VGG16 (Simonyan & Zisserman, 2014). Based on the authors' recommendation, this research implements the first four encoders of VGG16 network. Meanwhile, Badrinarayanan, Handa & Cipolla (2015) reported that the implementation of *upsampling* operation in the decoders, with the *maxpooling* indices used in corresponding encoders, could improve the accuracy of the SegNet model. This strategy is applied in this research by incorporating customized layers in the CNN model and the improvement has been observed. It should be noted that, for effectively training these three models, the pre-trained weights obtained by the ImageNet through Keras Application Programming Interface (API) (Chollet, 2015) are loaded as the initial weights of encoders.

2.2. Design of evaluation metrics

It is critical to establish a reliable evaluation methodology to ensure that the chosen model possesses an optimal performance among the introduced models, while its accuracy meets the requirement of the recording head factory. In this aspect, two evaluation metrics have been designed.

To compare the performance of the introduced semantic segmentation models on the prediction of defects, the F1-score criterion, as known as dice coefficient, is implemented. One of its original formulation can be expressed as:

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{1}$$

where the TP, FP and FN are true positive, false positive and false negative, respectively. However, it is difficult to apply this equation to the current application, because

1) There are multiple types of defect but this formulation is formed as a binary approach (a TE image could has one or more types of defects or no defect).

2) This formulation is limited by the situation that the TP, FP and FN are zeros and the true negative (TN) equals to total number of pixels. This situation occurs when calculating F1 score for defect classes ("good" pixels would be in the negative class), and the image has no defect while the model provides a perfect prediction (i.e., all pixels are predicted to be "good").

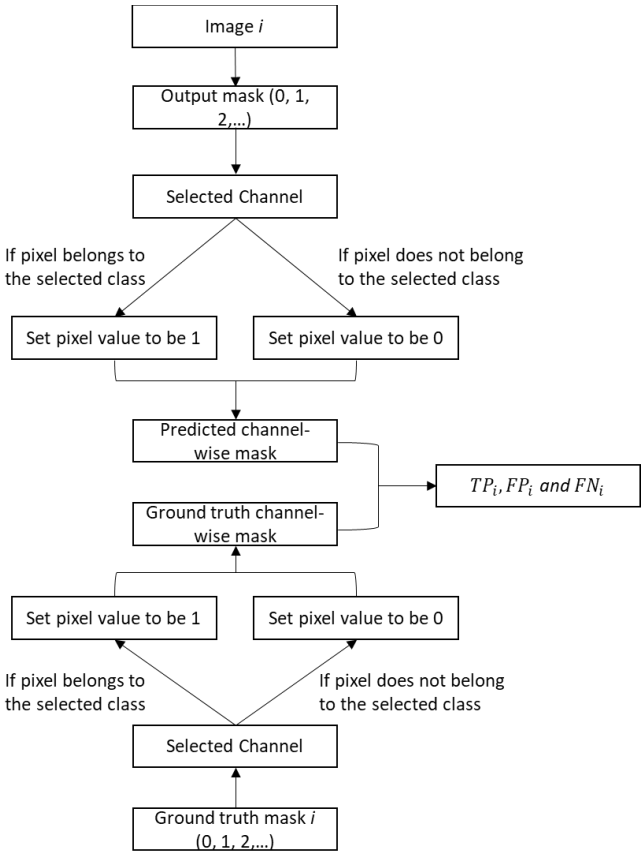


Figure 2. Schematic of workflow to calculate the F1 score for image *i*.

The second issue can be address by setting the F1 score to be 1 when TP, FP and FN are zeros. To address issue associated with multiclass prediction, the F1 score should be calculated for each defect (i.e., each output channel of the semantic segmentation model). Meanwhile, a cumulative calculation scheme is necessary, i.e., the F1 score should be calculated based on a dataset that has images with all possible types of classes instead of calculated with individual images then conducting averaging. Figure 2 shows the evaluation workflow for one channel/class of image i as an example. It is worth mentioning that one of the key steps is to transfer the multiclass mask to a binary class based on the target class that the F1 score is calculated for. For example, when calculating the F1 score for defect type one, all pixels belong to defect type one are regarded as positive class, while all others are set to be negative class. Therefore, for a selected class, the F1 score can be calculated as

$$F1_{class} = \frac{2\sum_{i=0}^{n} TP}{2\sum_{i=0}^{n} TP + \sum_{i=0}^{n} FP + \sum_{i=0}^{n} FN} \quad (2)$$

where n refers to the number of images to be evaluated.

The factory's inspection standard is, however, the accuracy of final decision made by the rule-based classifier using the predicted masks. Even the F1 score is convincible in comparing different CNN models, it does not necessarily indicate that the selected model achieves the same level of performance as human. Therefore, a second metric named attribute gage (AG) is defined for ensuring the accuracy of final decision. To be clear, this evaluation is conducted after the completion of CNN model comparison. Since the objective of the presented research is to implement the deep learning based computer vision instead of the factory labors, the set standard is that the AG of the trained model is at least equivalent to that of trained human. To design an unbiased evaluation, the following steps are taken. First, a test dataset is built with evenly distributed images for each class to ensure the data balance. Then, a subject matter expert (SME) reviews the images and give the decision on rejection or acceptance for each image. If a decision were rejection, the SME would provide the reason of rejection (i.e., which criterion and defect type the decision of rejection are made based on). Such aspects are regarded as the ground truth of the decision. Following the same procedure, multiple human are trained on rejection/acceptance criteria of TE images and provide their decisions for the same dataset. Finally, the trained CNN model is used to provide a mask for each image, and the rule-based classifier provides the decision of rejection or acceptance based on the predicted masks. Based on these steps, the AG of the developed system can be calculated as the percentage of the classifier's decisions that are agree with the ground truth in relation to the total number of made decisions. Similarly, the AG of one person is the percentage of number of decisions made by this person that are consistent with ground truth in relation to the total number of made decisions. The final AG of human is the averaged value of AGs calculated by each individual.

2.3. Results and discussion

2.3.1. Model Comparison

To compare F1 scores of the trained DeepLab, Mask-RCNN and SegNet, a dataset is built with 788 images (563 as training images and 252 as test images). There are roughly 150 and 50 images for each class (six classes in total) in training and test sets, respectively. The datasets include one "good" class and five different types of defects. The five defects include AChipA, AChipB, Contam, Wirebond and CopperExp. The models are built and trained using Keras API modules with the Tensorflow running at the backend (Abadi et al., 2016). The results obtained from the three models are listed in Table 1. In order to keep a consistent input size, all the images are resized to 224×896×3 before fitted into the CNNs. It can be seen that the DeepLab has higher F1 score in the "good" class; the three models achieve very similar F1 scores of AChipA, AChipB and Contam; SegNet's F1 scores of Wirebond and CopperExp are better than those of the other two models'.

The results suggest that, regardless which model to use, the F1 scores of the first four defect types are generally higher than CopperExp's. Based on the analysis of images, it is found that the size of CoppeExp are relatively small and its color is more likely to be mixed with the background (normal features). The demonstrative images are shown in the section 2.3.2 (Figure 4). Table 1 also concludes that the values of F1 scores are in the range of those reported in the literature (Kumar, 2018; Vuola, Akram & Kannala, 2019; Bertels, 2019), which indicates that the models are reasonably built and trained. Based on the comparison, SegNet is chosen because it performs better than the other two models in detecting the Wirebond and CopperExp.

Models / Defects	DeepLab	Mask-RCNN	Segnet
Good	0.92	0.82	0.88
AChipA	0.85	0.86	0.85
AChipB	0.83	0.83	0.85
Contam	0.77	0.76	0.79
Wirebond	0.59	0.71	0.72
CopperExp	0.46	0.53	0.56

Table 1. Comparison of F1 score on the test dataset.

2.3.2. Attribute Gage Investigation

This section introduces the modifications made on the chosen SegNet model, which is deployed to the edgeline computation environment and used for the AG study reported in this paper. For reader's reference, the detailed training/test procedures are also discussed in detail. In the end of this section, the AG study of comparing the performance of developed model with the factory operators is presented.

Based on the rejection/acceptance criteria, the AChipA and AChipB are two different defects. Their specifications, however, are very similar to each other; the only distinguishable configuration is that they locate at different regions of the recording heads. Therefore, we combine these two defects into one, named Chip, to further boost the performance of the SegNet model. In this scenario, the factory still maintains the two defects separately without any interruption, since the rule-based classifier can distinguish them easily based on their locations predicted by the semantic

segmentation model. The SegNet used for AG study is trained and validated with 2117 and 481 images, respectively, using a Linux machine with an Nvidia's GeForce RTX 2080 Ti GPU. Based on the GPU's capacity, the batch size is set to be four. Before training, a pixel level standardization is applied to all the images using the training dataset. Because of the multi-class segmentation nature, the loss function is set to be the categorical cross-entropy, while the Adam methodology is implemented as the optimization scheme.

The initial learning rate is set to be 0.0001. It would be reduced by one magnitude if the validation loss were not reduced in 5 epochs (the minimal learning rate is set to be 10^{-10}). In order to avoid overfitting, the early stopping criteria is also applied, i.e., the training process would be terminated if the validation loss were not reduced in 15 epochs. The training and validation losses with respect to the progress of epochs are shown in Figure 3. The training process stops after 96 epochs and the chosen model is achieved at epoch 81. The approximate computation time for each epoch is 264 seconds, indicating the total time of seven hours. In addition, while the F1 score is shown in Table 2, Figure 4 shows representative images in validation set including four types of defects with their corresponding predicted and ground truth masks. It indicates that the predictions possess good agreement with the ground truth masks. Checking the image with CopperExp, the model possesses an even more reliable prediction than the ground truth (marked by engineer): the trained model marks the area missed by the engineer on the third top pad.

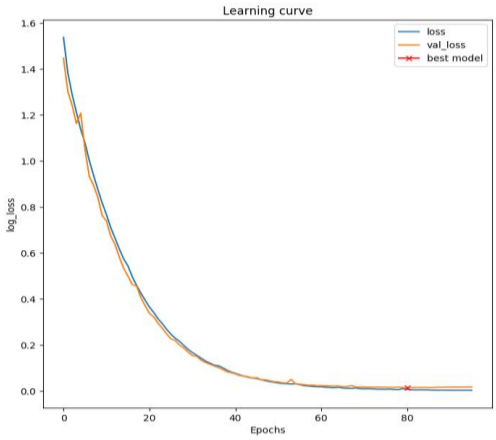


Figure 3. Learning curve of SegNet for AG study.

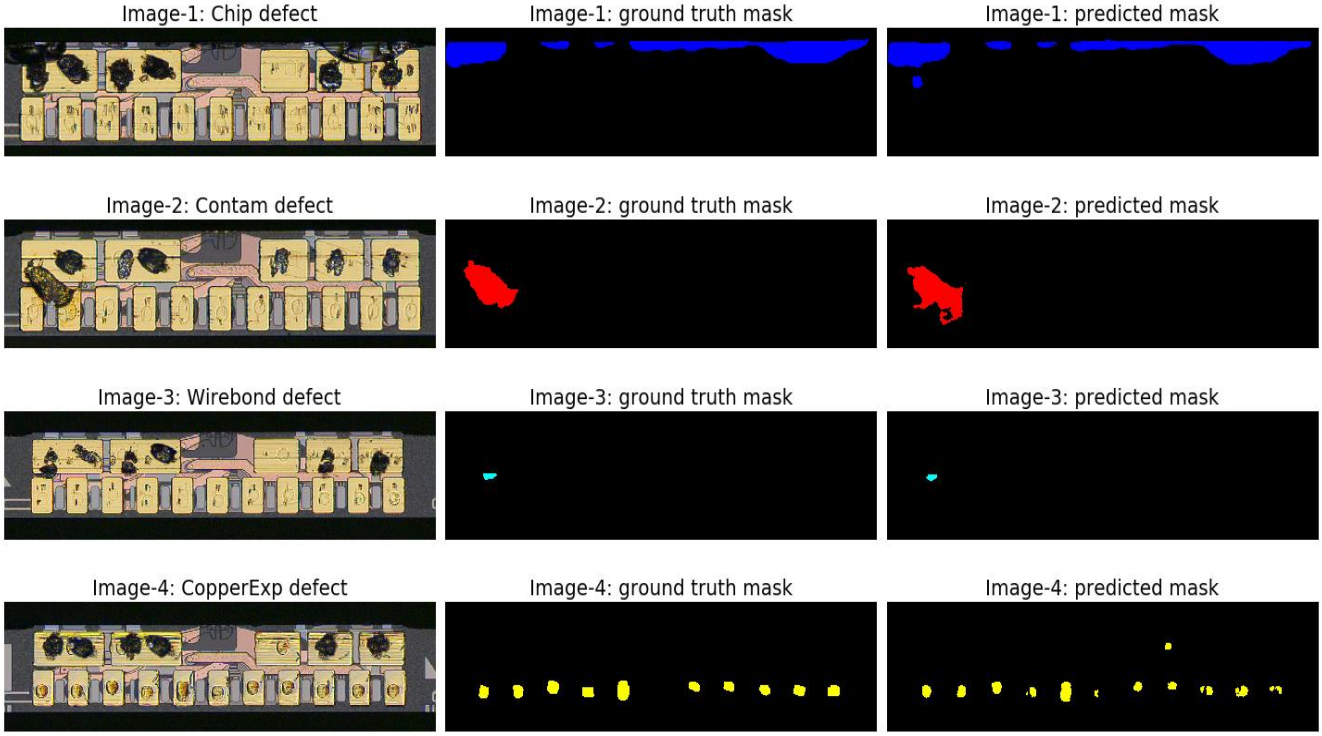


Figure 4. Images with four defined defect classes: Chip (blue; refer to chip on the TE), Contam (red; refer to contamination on the TE), Wirebond (cyan; refer to connection of two pads on the TE), CopperExp (yellow; refer to expose of copper on the TE) and background "good" class (black).

To evaluate the AG, a test dataset, also named golden standard dataset, is built with 40 images, which contains eight images for each class (five classes in total). The trained human and SME review these images firstly. Based on the introduced method, the averaged AG of the trained human is 79%. Then, the rule-based classifier analyzes the predicted masks obtained from the trained SegNet model to get the aspect of rejection and acceptance for each image. Comparing with the ground truth from SME, the AG of SegNet model is found to be 92.5%. A significant improvement is found in the AG of SegNet, which proves the robustness of the developed model. The human independency of the developed system is one of the factors that lead to the premium performance; the accuracy of the system would not be impacted by the human fatigue, random mistake and personal bias. In addition, the rejection/acceptance criteria are easier to be applied by the rule-based classifier than visual checking. In more detail, the rule-based classifier is able to strictly apply the defined criteria while human can only use the "feeling" of the define criterion if there are no optical measurement tool available. In addition, it should be noted that this research does not necessarily indicate that the trained SegNet model is the only/best solution for the current and similar applications. The decision of choosing the SegNet model is because it meets the set evaluation metrics. Moreover, with the high fidelity computation architecture introduced in the next section, it is not difficult to update or change the current applications if better CNNs are found.

	F1 score
Good	1.00
Chip	0.88
Contam	0.71
Wirebond	0.74
CopperExp	0.79

Table 2. F1 score of the SegNet deployed in the computation edgeline.

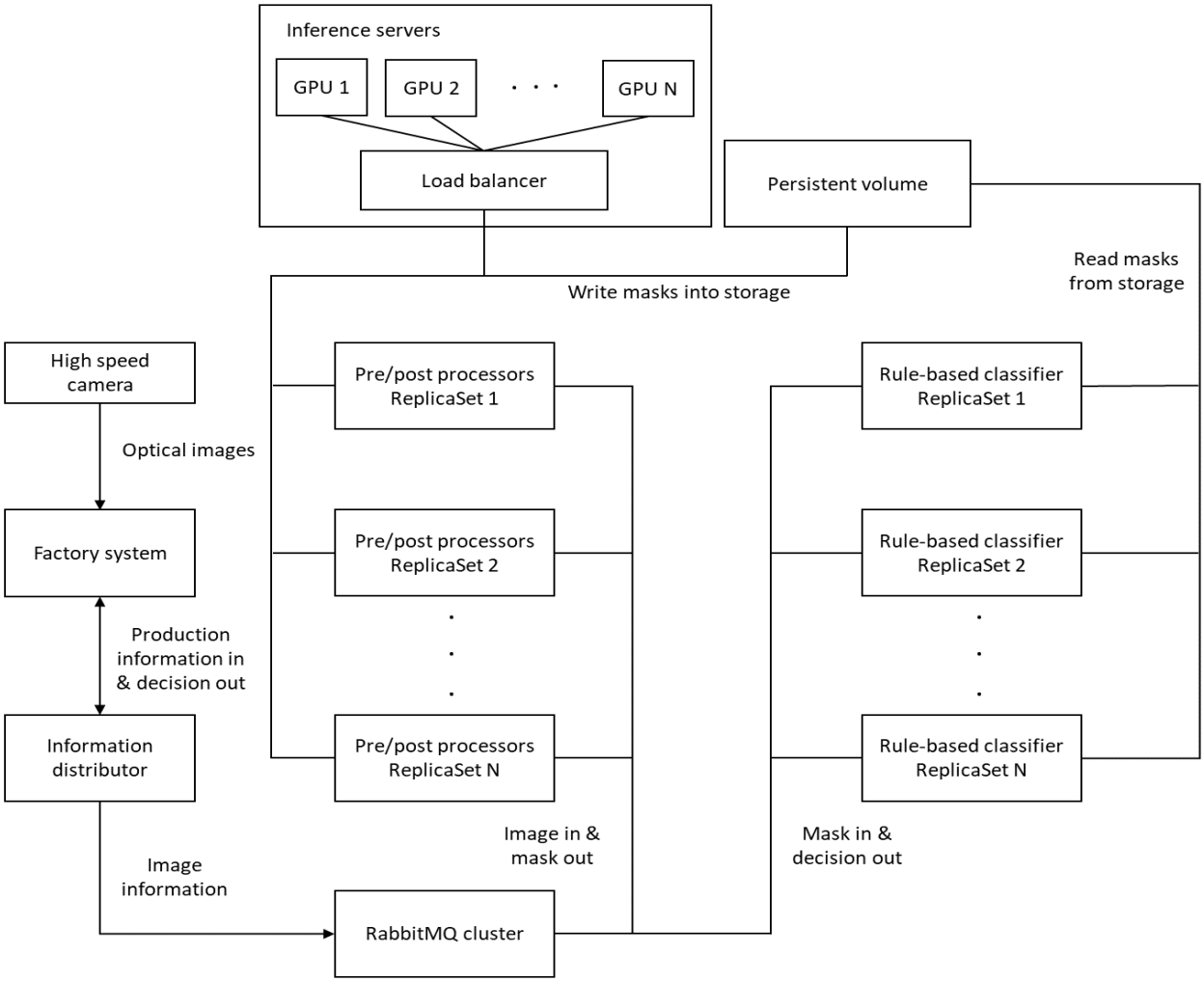


Figure 5. A schematic of OPICA system

3. COMPUTATION ARCHITECTURE DEVELOPMENT

Based on the throughput target, laboratorial laptops or workstations could not afford the required computation load. Therefore, a high performance computation engine is built as an edgeline located at the factory. This edgeline contains four Nvidia's Tesla T4 GPU with memory of 16 GB, and 32 Intel Xeon D-1587 CPUs are attached to each GPU. In addition, three CPU worker nodes are added to the edgeline with 16 Intel Xeon Gold 6148 CPUs on each node. A Kubernetes cluster is built with three master nodes and seven worker nodes on the edgeline that are used for cluster level management and computation, respectively. As introduced in the previous sections, all the applications are containerized as microservices using Docker containers and a Docker Trust Registry (DTR, which is made of three additional worker nodes) on the cluster manages the built Docker images for version control. A network file system (NFS) is also attached to the cluster as a persistent volume, which is the physical storage of necessary files for all applications including the Docker images. Figure 5 shows a schematic of the developed system, and the next two subsections will discuss the detailed configurations.

3.1. Inference Services

This section focuses on the details of inference services that are key components for optical image analysis, referring to the pre/post processors, inference servers (i.e., Tensorflow services) and rule-based classifiers. These applications are programmed using Python, and the messages used for microservices' communication are in JSON format. The information flow is as follows.

The pre/post processors obtained the saved directory of each image by consuming the corresponding message in the RabbitMQ server. Then, they read the images from the persistent volume (or any servers where the images are hosted) into the designated memory, and conduct necessary preprocessing including resizing and standardization. The preprocessed information will be sent to the inference servers, which are built by the trained SegNet model with the Tensorflow serving scheme (known as a gRPC service). After the pre/post processors receive the predicted results from the inference server as *softmax* channels, they will conduct *argmax* operations and write the predicted masks into the persistent volume. Then, a message including the path to the saved masks will be sent back to another queue in the RabbitMQ server. On the other side, the rule-based classifiers listen to this message queue, and consume the incoming messages instantaneously. With the directories of masks, the classifiers will read the masks into the memory, and apply the defined aspects of rejection/acceptance to each mask based on the location, type and size of the defects if any. The decisions of rejection or acceptance for the images will be sent back to a queue in the RabbitMQ server so that the factory can analyze the information easily.

In the current architecture, the inference servers are the only applications running on the GPUs under the supports of Tensorflow and Nvidia's CuDNN toolkits; other applications are running on the CPU worker nodes. Due to one trained CNN model occupies almost the entire memory of a GPU, the current edgeline system could afford at most four services running in parallel. However, the computation of pre/post processors are always intensive requiring more than four replicas in the CPU nodes. To optimize the utilization of the computation resource, a load balancer (Bowman-Amuah, 2003) is implemented as the connection mechanism between the pre/post processors and inference servers. Specifically, the load balancer evenly distributes the computation load from the pre/post processors to the running Tensorflow severs to avoid overwhelming a single GPU.

3.2. Other Supportive services

To enable a seamless connection between the developed system and factory's hardware, other supportive services are also essential. In this section, two main supportive systems are briefly discussed including the information distributor (named as OPICA Bridge) and the factory system (named as OPICA Dashboard) (see Figure 5).

The OPICA Dashboard is a web-based tool that informs the OPICA edgeline system for any new inspection requests. Once a series of recording heads scanned by the optical cameras, a unique name will be assigned to each optical image. The optical images will be then saved to either the NFS storage or a HTTPS server. Other important functions of the OPICA Dashboard include dataset management and model version control. With a scheduled human review strategy (i.e., human will review a certain proportion of the inspected images), the dashboard is able to visualize the accuracy of the current deployed inspection system. Moreover, operators and engineers can visualize specific images with predictions made by the SegNet model and rule-based classifier. For the images with relatively poor prediction accuracies, reviewers are able to provide annotations using the dashboard, and these images and their corresponding annotated masks (i.e., ground truth) will be reserved for the next round of model training or tuning.

The OPICA Bridge is a group of applications developed for processing the information from the factory then distributing corresponding messages to the RabbitMQ server. It acts as a "bridge" between the factory system and computation edgeline. More specifically, the OPICA Bridge builds messages with image locations, parameters for rule-based classifications, SegNet model names and versions, system status and recording head identification numbers based on the information sent by the factory system. Such messages are in a format that could be consumed by the pre/post processors directly. Another function of the bridge is to gather the predicted decision by the rule-based classifier for each image. It then sends this information to the factory so that the images

and their corresponding predictions are able to be visualized in the factory system. It also acts as a "security guardian": it checks the information from both sides (factory system and computation edgeline), and informs engineers if anything in the message is found to be inappropriate such as missing attributes and wrong formats. In such a way, unnecessary system level interruption can be effectively avoided.

3.3. System Throughput

The current microservice architecture is built based on the system level deployment trials and experimentations. In order to maximize the system level throughput, four inference serving pods are deployed (one pod per GPU). For readers' reference, a pod is generally known as a Kubernetes smallest deployable unit. The pre/post processing is found to be the most resource consuming applications on the CPUs, and 32 replicas are deployed to the cluster's CPU worker nodes. In addition, the number of replicas of rule-based classifier is set to be eight. It should be noted that the OPICA Bridge is running on the Kubernetes cluster as well. Each of its application possesses up to eight replicas, since these applications are not computational intense. A dataset with 615,000 images is used to evaluate the throughput of the entire developed OPICA system with an end-to-end system level throughput test. The current highest throughput that the system can achieve is approximate 36.7 images per second. It indicates the current system can afford a daily throughput of approximately 3.2 million high-resolution optical images per day, which is well aligned with the set throughput target.

4. CONCLUSION

This paper presents the development of OPICA system for TE defect inspection in recording head factories at Seagate. Two main topics are discussed including the investigation of CNN models and the deployment of the trained model to a Kubernetes cluster. First, in order to achieve a deep learning model with an acceptable accuracy, three state-of-the-art CNN based semantic segmentation models have been evaluated with specifically designed F1 score formulation. The selected model, SegNet, is then qualified by an AG study and shows an improved performance to the trained human, which fully meets the objectives of deep learning model development. Second, a Kubernetes cluster is built as a high-performance computation engine to enable high speed and high fidelity inspection including a message queuing system (AMQP supported). The current throughput is found to be 36.7 images per second, which meets the expected system peak throughput (three million images per day, i.e., 35 images per second).

The presented research is expected to be a useful reference for developing and deploying deep learning models for high-volume smart manufacturing. The designed evaluation metrics are examples of aligning deep learning model assessment with factory's manufacturing standard. In addition, the computation edgeline shows promising ability to handle the massive information generated by the high volume manufacturing environment. The developed system has been launched to the recording head factory and it is currently under production level validation. Meanwhile, RabbitMQ configuration and CPU allocations are under further optimization, which show promising in further boosting the system level throughput.

ACKNOWLEDGEMENT

This work is supported by Seagate Technology PLC.

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., (2016). Tensorflow: A system for large-scale machine learning. In *12th Symposium on Operating Systems Design and Implementation* (pp. 265-283).

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *39*(12), 2481-2495.

Badrinarayanan, V., Handa, A., & Cipolla, R. (2015). Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*.

Bernstein, D. (2014). Containers and cloud: From lxc to docker to kubernetes. *IEEE Cloud Computing*, *1*(3), 81-84.

Bertels, J., Eelbode, T., Berman, M., Vandermeulen, D., Maes, F., Bisschops, R., & Blaschko, M. B. (2019). Optimizing the Dice score and Jaccard index for medical image segmentation: Theory and practice. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 92-100). Springer, Cham.

Bonam, R., Tien, H.Y., Chou, A., Meli, L., Halle, S., Wu, I., Huang, X., Lei, C., Kuan, C., Wang, F. and Corliss, D., (2016). EUV mask and wafer defectivity: strategy and evaluation for full die defect inspection. In *Extreme Ultraviolet (EUV) Lithography VII* (Vol. 9776, p. 97761C). International Society for Optics and Photonics.

Bowman-Amuah, M. K. (2003). *U.S. Patent No. 6,578,068*. Washington, DC: U.S. Patent and Trademark Office.

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, *40*(4), 834-848.

Chollet, F. (2015). keras.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Huang, H. W., Li, Q. T., & Zhang, D. M. (2018). Deep learning based image recognition for crack and leakage defects of metro shield tunnel. *Tunnelling and Underground Space Technology*, *77*, 166-176.

Khan, M. W. (2014). A survey: image segmentation techniques. *International Journal of Future Computer and Communication*, *3*(2), 89.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

Kumar, P., Nagar, P., Arora, C., & Gupta, A. (2018). U-Segnet: fully convolutional neural network based automated brain tissue segmentation tool. In *2018 25th IEEE International Conference on Image Processing (ICIP)* (pp. 3503-3507). IEEE

Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., & Terzopoulos, D. (2020). Image Segmentation Using Deep Learning: A Survey. *arXiv preprint arXiv:2001.05566*.

Moeskops, P., Wolterink, J. M., van der Velden, B. H., Gilhuijs, K. G., Leiner, T., Viergever, M. A., & Išgum, I. (2016). Deep learning for multi-task medical image segmentation in multiple modalities. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 478-486). Springer, Cham.

Ramanujam, M. (2018). *U.S. Patent No. 10,023,231*. Washington, DC: U.S. Patent and Trademark Office.

Sadeghian, H., Koster, N. B., & van den Dool, T. C. (2013). Introduction of a high throughput SPM for defect inspection and process control. In *Metrology, Inspection, and Process Control for Microlithography XXVII* (Vol. 8681, p. 868127). International Society for Optics and Photonics.

Scheepers, M. J. (2014). Virtualization and containerization of application infrastructure: A comparison. In *21st Twente Student Conference on IT* (Vol. 21).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Vinoski, S. (2006). Advanced message queuing protocol. IEEE Internet Computing, 10(6), 87-89.

Vuola, A. O., Akram, S. U., & Kannala, J. (2019). Mask-RCNN and U-net ensembled for nuclei segmentation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)* (pp. 208-212). IEEE.

Yang, J., He, S., Lin, Y., & Lv, Z. (2017). Multimedia cloud transmission and storage system based on internet of things. *Multimedia Tools and Applications*, *76*(17), 17735-17750.

Zhang, W., Li, R., Deng, H., Wang, L., Lin, W., Ji, S., & Shen, D. (2015). Deep convolutional neural networks for multi-modality isointense infant brain image segmentation. *NeuroImage*, *108*, 214-22.