Fast and Accurate Convolution Neural Network for Detecting Manufacturing Data

Youcef Djenouri, Gautam Srivastava, and Jerry Chun-Wei Lin*

***Abstract*—This paper introduces a technique known as Clustering with Particle for Object Detection (CPOD) for use in smart factories. CPOD builds on regional based methods to identify smart object data using outlier detection, clustering, particle swarm optimization (PSO), and deep convolutional networks. The process starts by removing noise and errors from the images database by the Local Outlier Factor (LOF) algorithm. Next, the algorithm studies different correlations from the set of images in the database. This creates homogeneous, and similar clusters using the well known k-means algorithm, and the FastRCNN (Fast Region Convolutional Neural Network) uses these clusters to design efficient and more focused models. PSO is used to optimize the different parameters including, the number of neighbors of LOF, the number of clusters of k-means, the number of epochs, and the error learning rate for FastRCNN. The inference process benefits from the knowledge provided by training. Instead of considering a complex single model of the whole images database, we consider a simple homogeneous model. To demonstrate the usefulness of our approach, intensive experiments have been carried out on standard images database, and real smart manufacturer data. Our results show that CPOD when compared to baseline object detection solutions is superior in terms of runtime and accuracy.**

***Index Terms*—Object Detection, Clustering, Particle Swarm Optimization, Smart Factory, Deep Learning.**

I. Introduction

Intelligent Manufacturing has been gaining popularity in the last decade [1], in particular, numerous computer vision systems [2], [3] have been implemented in smart factory environments. Object detection [4] is one of the hottest research topics in smart manufacturing, where the aim is to identify objects from smart factory images. Identifying smart manufacturer objects can help for automatic surface inspection. Traditional machine vision methods have been widely used for automatic surface inspection to date. Common techniques include edge detection [5], and histogram-based features [6]. There are various features that can be extracted from an image and they need to be carefully crafted for specific tasks [7]. In recent years, several deep learning methods [8]–[12] have been used for accurately identifying smart factory objects. Solutions to object detection for smart factory data [8], [10] are known to be high in time complexity with very low accuracy. Even region based solutions [10] give strong accuracy compared to single-pass based solutions [8]. However, the overall process still suffers both from high runtime as well as low accuracy. The main reason for these issues is that all previous solutions are required to build complex models with a high number of parameters being fixed. This research work continues in this direction, and a new intelligent algorithm is proposed to efficiently, and accurately identify objects in smart factory environments.

Y. Djenouri is with the Department of Mathematics and Cybernetics, SINTEF Digital, Oslo, Norway. Email: youcef.djenouri@sintef.no

G. Srivastava is with the Department of Mathematics & Computer Science, Brandon University, Canada. Email: Srivastavag@brandonu.ca and also with the Research Centre for Interneural Computing, China Medical University, Taichung, Taiwan.

J. Chun-Wei Lin is with the Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway. Email: jerrylin@ieee.org (*Corresponding author)

A. Motivation

Solutions to object detection algorithms for smart manufacturing data suffer from the detection rate, which uses the entire images database in the learning process. Moreover, it is not straightforward to tune the hyper-parameters of deep learning models. Motivated by the success of clustering in many industrial applications [13], the research presented in this work splits the images database into homogeneous clusters to create simple and more focused models. Each model learns from homogeneous and similar images. This allows for the reduction of runtime compared to the existing solutions. Moreover, motivated by the success of metaheuristics in tuning the parameters of deep learning models [14], this research work incorporates particle swarm optimization (PSO) to not only tune the parameters of deep learning models but also to set the parameters of the entire process which includes the pre-processing step. Compared to previous work, this paper employs several innovations to improve training and inference speed while also increasing detection accuracy.

B. Contributions

In this paper, we propose the Clustering with Particle for Object Detection (CPOD) framework with the aim to create small and focused learning model for identifying objects in smart manufacturer data. The images database is first cleaned by removing noise in images. Next, it is divided into several small clusters that act as independently as possible. The training model is learned from each cluster of images. Next, during the inference step, instead of exploring the whole images database, we investigate only the effort in learning from the most similar clusters in the image query. With this in mind, the main contributions of this work are as follows:

1) LOF (Local Outlier Factor) algorithm is used to remove outliers from the feature images extracted using the convolution neural network (CNN).

2) An adapted k-means algorithm is proposed for clustering the feature images into similar clusters, and minimizes the number of the shared image features among clusters.
3) FastRCNN algorithm [4] is used on each cluster of feature images. During the inference process, we use the centroids information to derive the most similar clusters to the image query.
4) PSO (Particle Swarm Optimization) algorithm is used to tune the parameters of the different steps of CPOD framework, including the number of neighbors of LOF, the number of clusters of k-means, the number of epochs, and the learning error rate of FastRCNN.
5) CPOD is thoroughly evaluated by extensively analyzing its computational time as well as its accuracy with the baseline object detection algorithms on different image databases (standard, and smart manufacturer data). The results reveal the superiority of CPOD against FastRCNN [4], and the work of Park *et al.* [10] in both runtime and accuracy.

II. Related Work

Wang *et al.* [8] proposed the use of FastRCNN model for auto-sorting of object detection in the robotic arms environment. The approach aims to identify the top view object detection, various view angle object detection and various view angle plus occlusion object detection. Yuanbin *et al.* [9] proposed a CNN-based approach for detecting elementary objects in complex manufacturing system. The approach incorporates both a cloud-edge computing environment, and convolution neural network to identify elementary manufacturing objects in a reasonable amount of time. Li *et al.* [15] proposed a supervised deep neural network approach for image reconstruction of electrical resistance tomography problem. Different convolution neural layers are developed to accurately perform the feature extraction, the image reconstruction, the residual connection; and the jump connection.

Park *et al.* [10] proposed a hybrid deep neural network model for defect detection problem. A preprocessing and data enhancement are used to improve the deficiency detection accuracy of manufacturer objects in a smart-phone environment. Yuanbin *et al.* [11] used both faster regional convolutional neural network; and the cloud-edge computing environment to accurately locate small defects in geometrically complex products. The use of cloud-edge computing environment has greatly improved the computational time of the FastRCNN implementation in a complex manufacturing environment. Huang *et al.* [12] suggested the use of a deep neural network with auto-encoder to transform the object detection problem into a classification problem. This approach solves the real-value manufacturer data issue, however it highly suffers from the detection accuracy ratio. Kyeong-Beom *et al.* [16] Combining deep learning with augmented reality to detect 3D objects, and enable performing more realistic manufacturing jobs in complex environment. In addition, two real cases studies have been conducted, the first one aims to map a virtual object in a real world environment, and the second one aims to maintain and inspect a real 3D smart manufacturer objects.

Fangwei *et al.* [17] studied the effect of data volumes of 3D images to estimate the cost of a manufacturing process. It used the convolutional neural network in all processing stages including the feature learning, the object recognition, and the cost estimation. Jiang *et al.* [18] proposed an intelligent algorithm for autonomous gesture recognition, by combining objected detection, camera calibration, and gesture 3D information extraction. He *et al.* [19] proposed a classification priority network, which reverses the order of bounding boxes estimation, and prediction, to improve the defect detection process in analyzing the surface quality of steel products. Wen *et al.* [20] suggested the use of the data-driven model to identify objects in robotic environment. It combines several deep learning models to efficiently realize the object prediction, and then to control the behaviour of the different robots. Andrew *et al.* [21] proposed an intelligent system for object detection in process automation. A set of bank filters are first used to normalize the training 3D images by employing non-linear operations. Object matching is then investigated to derive the similar objects to the current video 3D frame.

Liu *et al.* [22] introduced a new unsupervised encoder layer to discover local 3D point-wise features, which improves both the feature discrimination, the robustness of the detection process. Kara *et al.* [23] developed a novel framework called HealthFog which can be used for integrating ensemble deep learning in Edge computing devices to automatically identify heart disease from medical images data. Tuli *et al.* [24] proposed the use of convolution neural network for table detection form the datasheet images in a supply chain applications. Liang *et al.* [25] proposed an edge computing-based convolution neural network model to deal with smart manufacturer objects in the internet of things.

Through this short yet informative literature review, solutions to object detection algorithms for smart manufacturer data suffer from weak detection rate since the whole images database is considered/used in the learning process. Moreover, it is not straightforward to tune the hyper-parameters of deep learning models. Clustering techniques has attracted high research interest in industrial applications [13]. These approaches use the well-know "divide and conquer" strategy, where the main purpose is to split data into several clusters and explore the generated clusters separately. This allows a high benefit in terms of computation time. In addition to clustering, PSO has been extensively explored in tuning CNN models [14]. These approaches consider the possible values of hyper-parameters as solution space, and develop intelligent operators to efficiently explore the entire solution space in order to find optimal hyper-parameters of deep learning models. Motivated by the success of cluster-based algorithms in handling industrial applications as well as the success of swarm intelligence approaches in tuning CNN models, in the next Section, we propose our hybrid algorithm, which combines clustering, PSO, and CNN to efficiently explore the images database, and accurately detect objects from smart manufacturer data.

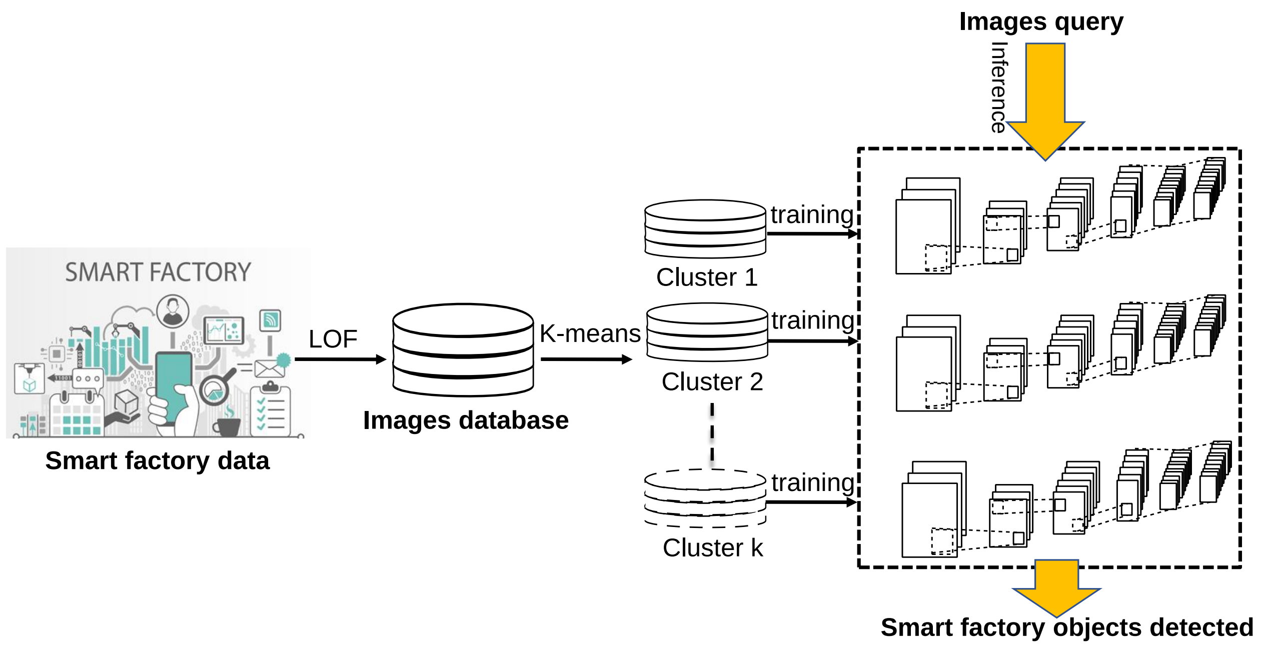

Fig. 1. CPOD Framework

III. CPOD: Clustering with Particle for Object Detection

A. Principle

This section presents the proposed CPOD framework, which integrates, deep learning and particle swarm optimization for identifying objects in smart factory environments. The framework considers as input smart manufacturing image data, and as output the smart factor object detected in these images. As shown in Fig. 1, CPOD consists of two stages:

1) **training**: which aims to first collect the manufacturer image data. The process starts by recording video frames from cameras. The frames are then transformed to images, where different distortion techniques such as mapping, resizing are used to correct the images. We then pre-process the whole set of images by removing noises and outliers. We then split the cleaned images into clusters, where each cluster contains similar images. The object detection process is applied on each cluster, where a particle swarm optimization is used to learn the hyper-parameters of the networks.
2) **inference**: which aims to identify the objects of the given image using the trained models of the previous step. This step benefits from the knowledge extracted in the previous step, where only considering the models of the most similar clusters to such image. The detail explanation of each stage is given in the following subsections.

B. Preprocessing

The aim of this step is to remove outliers, and noise from manufacturer's data. Several outlier detection methods have been used in the literature. Some algorithms used neighborhood computation, others used density estimation. Experimental evaluation reported in [26] revealed the success of density-based solutions against the neighborhood-based solutions, in particular the LOF algorithm. Therefore, we used LOF to identify anomalies from images. Instead of handling points in the classical LOF, our LOF deals with complex objects represented by the features of the images, where the difficulty behind this approach is how to determine the density of each image feature. The set of feature vectors from training images are computed using CNN. Our adapted LOF alongside a slightly more complex density estimation, compares the density estimate for each feature f with the density estimates of the kNNs of f. The density-estimate used in LOF is the so called *local reachability density* (lrd). It is the estimated distance at which a feature can be found by its neighbors, if a neighbor were to reach out lrd value distance in any direction, it would be likely/ most optimal to find that individual feature. It is the count of the items in the k nearest neighbor set, over the reachability-distance of the feature to all the values in its set, More formally, we have:

$$\text{lrd}(f) = 1 \bigg/ \frac{\sum_{p \in k\text{NN}(f)} \text{reach-dist}_k(f,p)}{|k\text{NN}(f)|} \quad (1)$$

where the reachability-distance (reach-dist) with parameter k is the maximum value of the k^{th} nearest neighbor of the feature and the distance of between the feature and its neighbor, more formally, it is given by:

$$\text{reach-dist}_k(f,p) = \max\{k\text{NN-}dist(p), dist(f,p)\} \quad (2)$$

with some distance measure $dist$ and kNN-$dist(p)$ being the distance between the feature p and the k^{th} nearest neighbor of p. The final outlier score of a feature is the product of the sum of the lrd of all the features in its k nearest neighbors and the the sum of the reachability-distance of all the features of the same set, to this feature, more formally, we have:

$$LOF(f) = \frac{1}{|k\text{NN}(f)|} \times \sum_{p \in k\text{NN}(f)} \frac{\text{lrd}(p)}{\text{lrd}(f)} \tag{3}$$

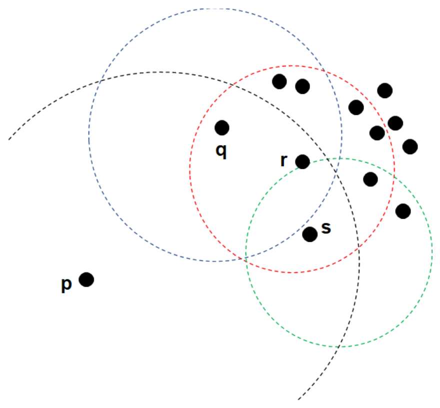

Fig. 2. Outlier detection of feature vectors of image data

Continuing on the example given in Fig. 2, the local density estimate for p (i.e., lrd(p)) would not be compared with *all* other local density estimates but only with the density estimates for its k nearest neighbors (i.e., lrd(q), lrd(r), and lrd(s) in the case of $k = 3$). The global ranking of all points according to the outlierness is then based on their *relative* (estimated) density, as compared to the densities of their k nearest neighbors (Eq. 3). This relation to the local characteristics of the dataset makes the method *local*. The features less than 1 are considered as outliers, and are not considered in next steps.

C. Training

This includes two main stages:

1. Clustering The aim of this step is to divide the whole images database into k clusters, $C = \{C_1, C_2 ... C_k\}$, where each cluster $C_s = \{I_1^{(s)}, I_2^{(s)} ... I_{|C_s|}^{(s)}\}$ is the subset of the images I. The overlapping features among the clusters should be minimized, and the overlapping features of images within each cluster should be maximized. More formally, we have to optimize the two following functions:

$$C^* = \begin{cases} \arg\min_{C} |\bigcup_{i=1,j=1}^{k} (\mathcal{F}^+(C_i) \cap \mathcal{F}^+(C_j))|, i \neq j \\ \wedge \\ \arg\max_{C} |\bigcup_{C_s}^{C} (\mathcal{F}(I_i^{(s)}) \cap \mathcal{F}(I_j^{(s)}))| \forall (i,j) \in [1..|C_s|]^2 \vee \\ i \neq j \end{cases} \tag{4}$$

Note that $\mathcal{F}(I_i)$ is the set of features of the image I_i, and $\mathcal{F}^+(C_i)$ is the set of features of the cluster C_i, it is the union of all features of images belonging to C_i.

Many clustering algorithms have been proposed in literature. Among them, it is well-known that k-means is one of the best partitioning algorithms which can satisfy the requirements mentioned in Eq. 4. Therefore, in this research work, we present an adaptation of k-means for clustering manufacturer images data by proposing the following concepts,

1) **Similarity computation**: The similarity measure between two image features $\mathcal{F}(I_i)$ and $\mathcal{F}(I_j)$ is computed as:

$$D(\mathcal{F}(I_i), \mathcal{F}(I_j)) = \frac{|\mathcal{F}(I_i) \cap \mathcal{F}(I_j)|}{\max(|\mathcal{F}(I_i)|, |\mathcal{F}(I_j)|)} \tag{5}$$

2) **Centroids updating**: Let us consider the set of image features of the cluster $C_i = \{\mathcal{F}(I_1)^{(i)}, \mathcal{F}(I_2)^{(i)}, ..., \mathcal{F}(I_{|C_i|})^{(i)}\}$. The aim is to find a gravity center of this set which is also an image feature. Inspired by the centroid formula developed in [27], we compute the centroid μ_i. The frequency of each item is calculated for all the image features of the cluster C_i. The length of the image features center is denoted by l_i, and corresponds to the average number of features of all image features in C_i as:

$$l_i = \frac{\sum_{j=1}^{|C_i|} |I(\mathcal{F}(I_j)^{(i)})|}{|C_i|} \tag{6}$$

Afterwards, the features of the images in C_i are sorted according to their frequency, and only the l_i frequent features are assigned to μ_i, as

$$\mu_i = \{j | j \in \mathcal{F}_{l_i}\} \tag{7}$$

Note that $\mathcal{F}_{l_i}$ denotes the set of the l_i frequent features of the cluster C_i.

We first compute the regional and global features using CNN for each image in the database. The main reason of computing regional and global features with CNN instead of local features in this stage is that the images database may contain dissimilar images, which should be assigned to different clusters, computing local features with SIFT extractor such as corners do not allow to determine such dissimilarities in an efficient way. The feature vectors of the images database are then grouped into several clusters using k-means algorithm. The process starts by assigning randomly the images to the k clusters and a centroid is computed for each cluster. Then, each image is assigned to a cluster whose centroid is the closest to that image using Eq. 5, 6, and 7. This process is repeated until there is no further assignment of the images to the clusters.

2. Object Detection The aim of this step is to build the object detection models, one for each cluster of images. We used the regional convolution neural network [4]. For every image in each cluster, regions of interests are determined and passed onto the hidden layer where the Relu activation function is performed and the same process as CNN is followed. In FastRCNN, the model performs better and more quickly as the regions of interest are found using a selective search method and all the regions of interests are found at once for an image, unlike CNN which finds ROI (regions of interest) and applies Relu on each ROI separately which is more time-consuming and slower. In practice, the training images have a high resolution, from 5.000 pixels to 100.000 pixels. As a result, a millions to billions of region proposals have been generated, which became the whole system very high time and memory consuming, and in some cases, the system will be bluntly blocked after several days and weeks of processing. To deal with this issue, we propose a strategy to prune, and filter the number of bounding boxes. Two manufacturer data of different categories, and allocated in the same frame should not close from each other, for consequent two bounding boxes in the same image should not be close to each other. Therefore, we compute the similarity between each new bounding box generated and the bounding boxes already generated. The similarity between two bounding boxes is determined by the number of pixels that separate these bounding boxes. We only keep the bounding boxes, which gives high diversity of the image. It means the minimal set of bounding boxes which cover the maximum of pixels in the image. In addition, to well optimize the hyper-parameters of the FastRCNN algorithm, we used the particle swarm optimization in the whole process. We chose PSO thanks to the efficient balancing between the diversification and the intensification issues, which are both important in hyper-parameter optimization. In the following we present the main components of the PSO for solving the hyper-parameter optimization problem.

a) **Population Initialization**: The initial population of the particles are generated randomly from the entire solution space, which represents the possible configurations of the number of neighbourhoods of LOF, the number of clusters of k-means, and the number of epochs, the error learning rate for FastRCNN algorithm.

b) **Particle Updating**: Considering a swarm with P particles, there is a position vector X$_{ti}$= $(x_{i1}x_{i2}x_{i3}\dots x_{in})^T$ and a velocity vector $V_{ti} = (v_{i1}v_{i2}v_{i3}\dots v_{in})^T$ at a t iteration for each one of the i particle that composes it. The particles update their positions in the solutions using the velocity formula as follows:

$$V_i^{t+1} = w \times V_i^t + c1 \times (p^t - X_i^t) + c2 \times (p^* - X_i^t) \quad (8)$$

and

$$X_i^{t+1} = X_i^t + V_i^{t+1}, \quad (9)$$

where i = 1, 2, ..., P. From Eq. 8, it shows that two factors c_1, and c_2 contribute to the movement of a particle in an iteration. p^t is the position of the best particle at iteration t, and p^* is the position of the best particle of all iterations. Furthermore, Eq. 9 is used to update the position of a particle. The parameter w is a positive constant value. This parameter is important for balancing the global search, also known as exploration (when higher values are set), and local search, known as exploitation (when lower values are set).

c) **Fitness Computing**: The evaluation function of the solution S will be the sum of both the classification, and the regression rate of the FastRCNN algorithm on the configuration made by S. The aim is to maximize the following function:

$$Fitness_{max}(S) = C(S) + R(S), \quad (10)$$

where $C(S)$ is the classification rate of the FastRCNN on the configuration made by S, and $R(S)$ is the regression rate of the fast FastRCNN on the configuration made by S.

D. Inference

The inference has the goal to detect the objects from the input image. It benefits from the knowledge extracted in the previous step, the features of the image query are extracted using CNN algorithm, we then determine the similarity between each centroid, and the features of such image. The similarity between two image features f_i and f_j, denoted as $d(f_i, f_j)$, is defined by their symmetric difference, i.e., the number of all feature points in the two image features minus the number of shared feature points in the two image features. Formally:

$$d(f_i, f_j) = |f_i| - \{|(p_{il}, p_{jl})|, \forall\ l \in [1..|f_i|]\} \quad (11)$$

Instead of using the whole models in the inference process, we explore the clusters of the images according to their similarity to this image. The search starts by exploring the images of the most similar cluster to the image query, and then we used the model learned from this cluster to detected the object of the input image. In this step, different kind of inference are generated, inference from computers, which we send the trained model to the computers, and infer the model from such computer for each new trajectories image data. We can also use smartp hones which support Andorid, and GPU computing to infer the group of trajectory outliers in real time processing. In this context, several technologies could be integrated such as TensorflowLite[1].

Algorithm 1 CPOD Algorithm

```
1:  Input: $I = \{I_1, I_2, \dots, I_n\}$: the set of images. $q$: the input
    image for inference. $IMAX$: the maximum number of
    iterations of PSO.
2:  Output: $O$: the set of the detected objects of $q$.
    ************Training*****************
3:  $I \leftarrow CNN(I)$;
4:  $I \leftarrow LOF(CNN(I))$;
5:  $C \leftarrow kmeans(CNN(I))$;
6:  $models \leftarrow \emptyset$;
7:  InitializationParticules();
8:  $iteration \leftarrow 1$;
9:  while iteration $\leq$ IMAX do
10:     for s=1 to k do
11:         $model_s \leftarrow FastRCNN(C_s)$;
12:         $g_s \leftarrow centroid(C_s)$;
13:         $models \leftarrow models \cup \{model_s, g_s\}$;
14:     end for
15:     EvaluateParticules();
16:     UpdateParticules();
17:     $iteration \leftarrow iteration + 1$;
18: end while
    ************Inference*****************
19: for s=1 to k do
20:     $model' \leftarrow Similarity(q, models)$;
21: end for
22: $O \leftarrow Searching(model', q)$;
23: return $O$;
```

[1]https://www.tensorflow.org/lite

Algorithm 1 presents the pseudo-code of the CPOD algorithm. Note that `LOF()` is the local outlier factor algorithm to remove outliers from images. `CNN()` is the feature extractors from images using the convolution neural network. `kmeans()` is the the $k-$means function which groups the set of features images in k clusters. `FastRCNN()` is the function to identify objects from each cluster of imag
remark that the training is the high time consumin
which includes several loops, and several scanning
images database. However, the inference contains on
loop, and needs scanning only the images for the r
clusters to the user query. However, the training is perf
only once, independently from the number of image q
The complexity cost of the inference depends to the n
clusters visited during the search process, noted l. Assur
the cost of FastRCNN is $\mathcal{O}(n)$, then the complexity
CPOD is $\mathcal{O}(\frac{l}{k}n)$. Ideally, only a single cluster is ex
which costs $\mathcal{O}(\frac{1}{k}n)$, and in the worst case, all clust
explored, which costs $\mathcal{O}(n)$.

IV. Performance Evaluation

Extensive experiments have been carried out to e
the performance of the proposed approach (CPOD) usi
challenging standard image databases `VOC 2012`, and
smart manufacturer data. Details of these image databa
given in the following:

1) `VOC 2012`: The challenging `VOC 2012` i
database [28] was used which contains $17,125$ ima
different objects, each image is represented by di
number of pixels, where the resolutions of the i
are high and more than 200×200 pixels per
This dataset contains 20 different classes. Each class represents one category of objects such as person, bird, cat, cow, dog, horse, sheep, aeroplane, bicycle, boat, bus, car, motorbike, train, bottle, chair, dining table, potted plant, sofa, tv/monitor.
2) Smart Manufacturer Data: A motorcycle wheel production line dataset was used as described in [9], this dataset contains $4,000$ different images, each image contains a wheel, each of which consists of three elements a rim, a hub and spokes. There is 33 different types of wheel in totals.

To evaluate the detected objects, computational runtime, and accuracy represented by mAP (mean Average Precision) are used. mAP is widely used metric to evaluate the object detection systems, and it is defined as follows,

$$mAP = \frac{\sum_{i=0}^{n} AvgP(i)}{n}, \qquad (12)$$

where n is the number of all objects to be detected, and $AvgP(i)$ is the precision at rank i, i.e., the first i ranked objects is considered while the remaining objects are ignored.

All implementations are executed on a computer with i7 PC, coupled by a GPU graphics card, `GeForce GTX 1070`. First, the training step is analyzed by tuning the parameters of the particle swarm optimization. The best configuration of CPOD is then compared to the state-of-the-art object detection solutions, by varying both the number of images in the database, and the number of images to be detected.

A. Parameters Setting

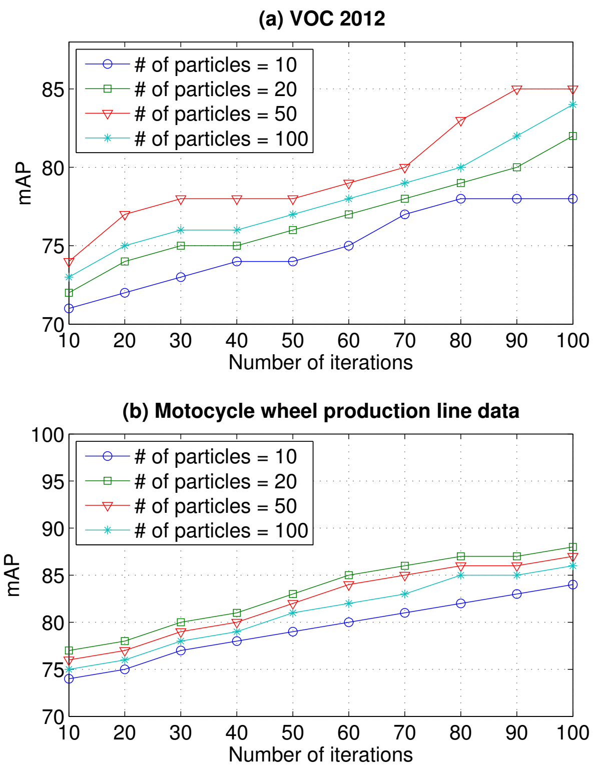

Fig. 3. Parameter Setting of CPOD

The aim of this experiment is to tune the parameters of the CPOD framework. Fig. 3 presents the quality of the objected detection of CPOD algorithm using both `VOC 2012`, and motorcycle wheel production line data. By varying the number of iterations of the particle swarm optimization from 1 to 100, and the number of particles from 10 to 100, the detection quality of CPOD is increased, for both databases. The best parameters of CPOD for both databases are illustrated in Table I.

TABLE I
Best Parameters of the CPOD.

Parameters	`VOC 2012`	motor. prod.
Number of Iterations	100	90
Number of Particles	20	50
Number of Neighbors	10	15
Number of Clusters	25	20
Number of Epochs	950	420
Error Learning Rate	0.005	0.002

B. CPOD Vs State-of-the-art Object Detection Algorithms

The aim of this experiment is to compare CPOD with three baseline algorithms, FastRCNN [4], Park *et al.* [10], and Yolo [29] in terms of accuracy and runtime. Fig. 4 present

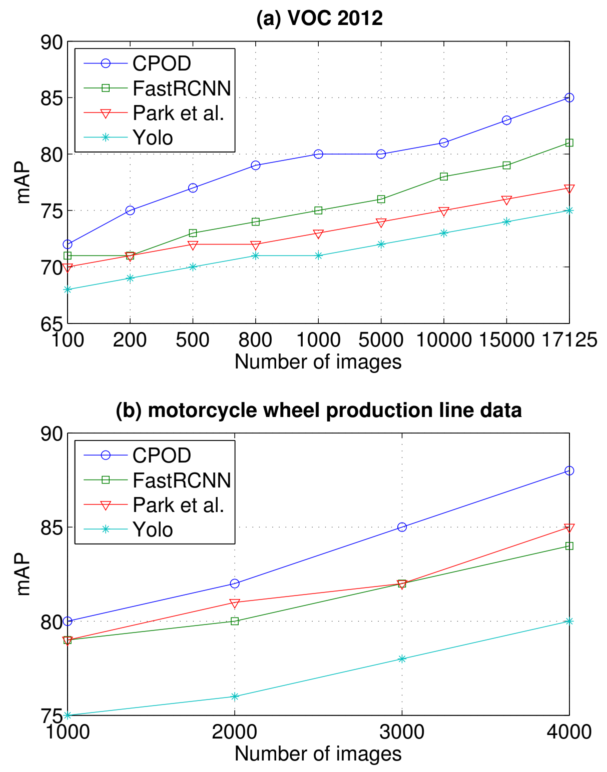

Fig. 4. CPOD Vs State-of-the-art Object Detection Algorithms: Accuracy

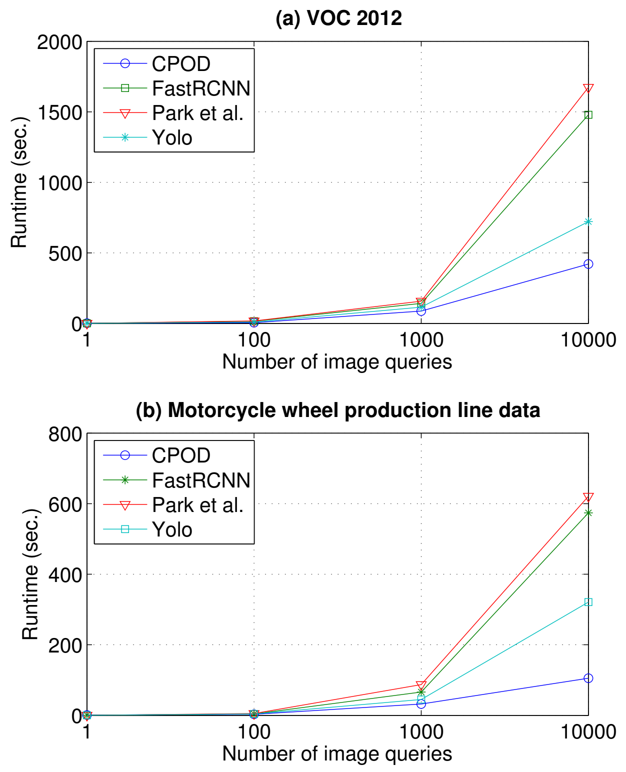

Fig. 5. CPOD Vs State-of-the-art Object Detection Algorithms: Runtime

the accuracy of the CPOD approach on `VOC 2012`, and motorcycle wheel production line image databases, compared with FastRCNN [4], Park *et al.* [10], and Yolo [29]. By varying with the number of images used as input, CPOD outperforms the two baseline algorithms in terms of accuracy, determined by mAP measure. In fact, the accuracy of CPOD exceeds 84%, where the accuracy of the other algorithms does not reach 80%. This is explained by the fact that the CPOD cleans the database before the learning process, the learning is highly focused on similar clusters, and the parameters are well selected using the particle swarm optimization. Fig. 5 present the runtime of the CPOD approach on `VOC 2012`, and motorcycle wheel production line image databases, compared with FastRCNN [4], Park *et al.* [10], and Yolo [29]. By varying with the number of image queries from 1 to $10,000$ queries, CPOD outperforms the two baseline algorithms in terms of runtime. In fact, the runtime of CPOD does not exceed 250 seconds with handling 10,000 queries, where the runtime of the other algorithms reach $1,700$ seconds with handling the same number of user queries. This is explained by the fact that the CPOD explores only the similar clusters for each image query, where the whole images database is used by the state-of-the-art object detection solutions, and for this for dealing each image query.

TABLE II
CPOD VS STATE-OF-THE-ART OBJECT DETECTION ALGORITHMS: TRAINING TIME (IN HOURS).

Algorithms	`VOC 2012`	motor. prod.
CPOD	5.26	1.74
FastRCNN	6.75	2.38
Park et al. [10]	7.79	3.19
Yolo	5.71	2.05

Table II compares training time (hours) between CPOD, FastRCNN, Park *et al.* [10], and Yolo. CPOD processes images faster than FastRCNN, Park *et al.* [10], and Yolo algorithms. This is mainly explained by the fact that CPOD efficiently eliminates hundreds to thousands of bounding boxes in the pre-processing step. Moreover, small number of similar images are trained by each learner.

V. DISCUSSION AND FUTURE PERSPECTIVES

This section discusses the main findings from the application of CPOD to the object detection problem.

- The proposed framework is not only able to identify the objects from the images database, but studying the different correlations and similarities between the images and find out disjoint groups among them. In the context of object detection, we argue that considering the clustering techniques in the preprocessing step allows to quickly identify the objects.
- From a deep learning research standpoint, CPOD is an example of combining different methods to optimize the learning process. In our specific context, outlier detection, clustering, particle swarm optimization, and convolutional neural network meet the object detection for exploring the smart manufacturer data. This adaptation is implemented in different phases, such as removing noises, find similar clusters, learning process, and hyper-parameters optimization.

- Another finding of this study is that the learning process benefits from the data preprocessing by using outlier detection, clustering. Thus, each model learns from clean and similar images, this accelerates the learning process, by creating simple smaller models. Each model is designed for identifying objects from homogeneous, and similar images.
- The last observation is that the framework is generic and could be applied in any computer vision problem, contrary to the other algorithms, which can deal only a particular computer vision problem. Object detection problem illustrated in this paper is just an example of application of our framework. Other computer vision problems such as classification and others may be solved by our framework.

Motivated by the promising results shown in this paper, different directions may be investigated:

1) **Improving the clustering step.** k-means has been used as clustering technique in this research work to split the images database into different homogeneous and similar clusters. Additional techniques can possibly be used for improving the clustering process, and then reduce the number of shared features among the images of different clusters. Thus, an interesting topics for future work is to integrate other clustering techniques into the CPOD framework, such as intelligent hierarchical, overlapping, or methods from other fields such as entity resolution and/or record linkage. Another thing that can be done is to find an appropriate mechanism to automatically fix the number of clusters. Using several runs to find the best value of the number of clusters is not very efficient in practice. One way to address this issue is to create a knowledge base containing each training images database, with the best value of the number of clusters, and then study the correlation between the meta-features of the images databases (number of features, number of images, luminous pixel values, etc.), and the best values of the number of clusters. This can help to automatically predict the best value of the number of the clusters of the new images database.
2) **Improving the learning step.** We plan to boost the performance of the CPOD and apply it to more complex computer vision applications in the smart manufacturer environment by exploiting the high-performance computing tools such as GPUs, supercomputers and cluster computing. In this context, the aim is to create independent job for each cluster of images by respecting the high-performance computing challenges including threads divergence, synchronization, communication, memory management, and load balancing. In this context, strategies to deal with load balancing seems promising. One way to address this issue is to develop clustering strategies allowing to find out equitable clusters in terms of number of images per cluster. Another way is to develop new strategies for repairing clusters, to find clusters with approximately the same number of images. Applying CPOD on MapReduce is also an alternative approach for improving both the training, and the inference step.
3) **Case studies.** In this paper a case study of an application of CPOD in the smart manufacturer data is shown. Motivated by the promising results shown in this first case study, we plan to extend CPOD for solving domain-specific complex problems requiring learning from big data. This can be found, for instance, in the context of intelligent transportation applications or in the context of medical data. Other potential use is the learning from sensor data, notably for realtime applications related to Internet of things and cyber-physical systems such as road traffic management and related services, energy management in smart buildings and smart grids, where the learning process is required to be performed within a very short latency [30].

VI. Conclusion and Future Work

This paper presents the CPOD (Clustering with Particles for Object Detection) framework. CPOD is used for solving the object detection problem in smart factory environments. The approach starts by cleaning the outliers using the local outlier factor, which considered as noises on the learning step. Next, the approach explores the different correlations between the images and learn from similar clusters, which allows to create more focused models, one for each cluster. Particle swarm optimization is used as well to optimize the parameters of CPOD during the preprocessing as well as the learning steps. To evaluate the CPOD framework, intensive experiments have been carried out on two image retrieval databases; one standard database (`VOC 2012`) as well as one real smart factory dataset (motorcycle wheel production line data). The results achieved are very promising compared to baseline object detection solutions in terms of both a faster runtime as well as higher accuracy. As future avenues of research, we plan to investigate other intelligent techniques in order to improve the CPOD framework and make it competitive and applicable for real-world applications. In the future, the modified CPOD prototype should be designed to deal with complex scenarios in smart manufacturer data, where objects may move continuously within short video frames.

References

[1] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, “Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges,” *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[2] J. Wan, J. Li, M. Imran, and D. Li, “A blockchain-based solution for enhancing security and privacy in smart factory,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3652–3660, 2019.

[3] J. Wan, S. Tang, D. Li, M. Imran, C. Zhang, C. Liu, and Z. Pang, “Reconfigurable smart factory for drug packing in healthcare industry 4.0,” *IEEE transactions on industrial informatics*, vol. 15, no. 1, pp. 507–516, 2018.

[4] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[5] H. Kong, J. Yang, and Z. Chen, “Accurate and efficient inspection of speckle and scratch defects on surfaces of planar products,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1855–1865, 2017.

[6] Ç. Aytekin, Y. Rezaeitabar, S. Dogru, and I. Ulusoy, "Railway fastener inspection by real-time machine vision," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 7, pp. 1101–1107, 2015.

[7] D. Weimer, B. Scholz-Reiter, and M. Shpitalni, "Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection," *CIRP Annals*, vol. 65, no. 1, pp. 417–420, 2016.

[8] T. Wang, Y. Yao, Y. Chen, M. Zhang, F. Tao, and H. Snoussi, "Auto-sorting system toward smart factory based on deep learning for image segmentation," *Ieee Sensors Journal*, vol. 18, no. 20, pp. 8493–8501, 2018.

[9] Y. Wang, K. Hong, J. Zou, T. Peng, and H. Yang, "A cnn-based visual sorting system with cloud-edge computing for flexible manufacturing systems," *IEEE Transactions on Industrial Informatics*, 2019.

[10] J. Park, H. Riaz, H. Kim, and J. Kim, "Advanced cover glass defect detection and classification based on multi-dnn model," *Manufacturing Letters*, vol. 23, pp. 53–61, 2020.

[11] Y. Wang, M. Liu, P. Zheng, H. Yang, and J. Zou, "A smart surface inspection system using faster r-cnn in cloud-edge computing environment," *Advanced Engineering Informatics*, vol. 43, p. 101037, 2020.

[12] H. Huang, L. Zhao, H. Huang, and S. Guo, "Machine fault detection for intelligent self-driving networks," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 40–46, 2020.

[13] P. Jia, X. Wang, and K. Zheng, "Distributed clock synchronization based on intelligent clustering in local area industrial iot systems," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 3697–3707, 2019.

[14] F. Li, X. Cai, L. Gao, and W. Shen, "A surrogate-assisted multiswarm optimization algorithm for high-dimensional computationally expensive problems," *IEEE Transactions on Cybernetics*, 2020.

[15] F. Li, C. Tan, F. Dong, and J. Jia, "V-net deep imaging method for electrical resistance tomography," *IEEE Sensors Journal*, 2020.

[16] K. B. Park, M. Kim, S. H. Choi, and J. Y. Lee, "Deep learning-based smart task assistance in wearable augmented reality," *Robotics and Computer Integrated Manufacturing*, vol. 63, p. 101887, 2020.

[17] F. Ning, Y. Shi, M. Cai, W. Xu, and X. Zhang, "Manufacturing cost estimation based on a deep-learning method," *Journal of Manufacturing Systems*, vol. 54, pp. 186–195, 2020.

[18] D. Jiang, Z. Zheng, G. Li, Y. Sun, J. Kong, G. Jiang, H. Xiong, B. Tao, S. Xu, H. Yu *et al.*, "Gesture recognition based on binocular vision," *Cluster Computing*, vol. 22, no. 6, pp. 13 261–13 271, 2019.

[19] D. He, K. Xu, and P. Zhou, "Defect detection of hot rolled steels with a new object detection framework called classification priority network," *Computers & Industrial Engineering*, vol. 128, pp. 290–297, 2019.

[20] Z. Wen, D. Liu, X. Liu, L. Zhong, Y. Lv, and Y. Jia, "Deep learning based smart radar vision system for object recognition," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 3, pp. 829–839, 2019.

[21] A. D O'Riordan, D. Toal, T. Newe, and G. Dooly, "Object recognition within smart manufacturing," *Procedia Manufacturing*, vol. 38, pp. 408–414, 2019.

[22] H. Liu, Y. Cong, C. Yang, and Y. Tang, "Efficient 3d object recognition via geometric information preservation," *Pattern Recognition*, vol. 92, pp. 135–145, 2019.

[23] E. Kara, M. Traquair, M. Simsek, B. Kantarci, and S. Khan, "Holistic design for deep learning-based discovery of tabular structures in datasheet images," *Engineering Applications of Artificial Intelligence*, vol. 90, p. 103551, 2020.

[24] S. Tuli, N. Basumatary, S. S. Gill, M. Kahani, R. C. Arya, G. S. Wander, and R. Buyya, "Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments," *Future Generation Computer Systems*, vol. 104, pp. 187–200, 2020.

[25] F. Liang, W. Yu, X. Liu, D. Griffith, and N. Golmie, "Towards edge-based deep learning in industrial internet of things," *IEEE Internet of Things Journal*, 2020.

[26] Y. Djenouri, A. Belhadi, J. C.-W. Lin, D. Djenouri, and A. Cano, "A survey on urban traffic anomalies detection algorithms," *IEEE Access*, vol. 7, pp. 12 192–12 205, 2019.

[27] Y. Djenouri, Z. Habbas, and D. Djenouri, "Data mining-based decomposition for solving the maxsat problem: toward a new approach," *IEEE Intelligent Systems*, vol. 32, no. 4, pp. 48–58, 2017.

[28] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[29] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1316–1322.

[30] D. Djenouri, R. Laidi, Y. Djenouri, and I. Balasingham, "Machine learning for smart building applications: Review and taxonomy," *ACM Computing Surveys*, vol. 52, no. 2, pp. 1–36, 2019.

Youcef Djenouri obtained the PhD in Computer Engineering from the University of Science and Technology USTHB Algiers, Algeria, in 2014. He is working as research scientist at SINTEF Digital, Oslo, Norway. He is working on topics related to artificial intelligence and data mining, with focus on deep learning, parallel computing, swarm and evolutionary algorithms. He published more than 70 papers in refereed journals, and conferences.

Gautam Srivastava was awarded his B.Sc. degree from Briar Cliff University in the U.S.A. in the year 2004, followed by his M.Sc. and Ph.D. degrees from the University of Victoria in Victoria, British Columbia, Canada in the years 2006 and 2012, respectively. Dr. G, as he is popularly known, is active in research in the field of Cryptography, Data Mining, Security and Privacy, and Blockchain Technology. In his 5 years as a research academic, he has published a total of 90 papers in high-impact conferences in many countries and in high-status journals (SCI, SCIE). He is an IEEE Senior Member.

Jerry Chun-Wei Lin received his Ph.D. from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan in 2010. He is a full Professor at the Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, Bergen, Norway. He has published more than 300 research articles in refereed journals (IEEE TKDE, IEEE TCYB, IEEE TII, IEEE SysJ, IEEE SensJ, ACM TKDD, ACM TDS, ACM TMIS) and international conferences (IEEE ICDE, IEEE ICDM, DASSFA, PKDD, PAKDD). His research interests include data mining, soft computing, artificial intelligence and machine learning, and privacy preserving and security technologies. He is the Editor-in-Chief of the International Journal of Data Science and Pattern Recognition, Associate/Guest Editor of IEEE Access, JIT, PlosOne, International Journal of Interactive Multimedia and Artificial Intelligence, IEEE TFS, IEEE TII, Applied Sciences, and Sensors. He is a Fellow of IET (FIET), a senior member for both IEEE and ACM.