Comparing Deep Learning And Support Vector Machines for Autonomous Waste Sorting

George E. Sakr
Electrical and
Computer Engineering Department
St. Joseph University
Beirut, Lebanon.
Email: georges.sakr@usj.edu.lb

Maria Mokbel
Electrical and
Computer Engineering Department
St. Joseph University
Beirut, Lebanon.
Email: maria.mokbel@net.usj.edu.lb

Ahmad Darwich
Electrical and
Computer Engineering Department
St. Joseph University
Beirut, Lebanon.
Email: ahmad.darwich@net.usj.edu.lb

Mia Nasr Khneisser
Electrical and
Computer Engineering Department
St. Joseph University
Beirut, Lebanon.
Email: mia.nasrkhneisser@net.usj.edu.lb

Ali Hadi
Electrical and
Computer Engineering Department
St. Joseph University
Beirut, Lebanon.
Email: ali.hadi1@net.usj.edu.lb

Abstract*—Waste sorting is the process of separating waste into different types. The current trend is to efficiently separate the waste in order to appropriately deal with it. The separation must be done as early as possible in order to reduce the contamination of waste by other materials. The need to automate this process is a significant facilitator for waste companies. This research aims to automate waste sorting by applying machine learning techniques to recognize the type of waste from their images only. Two popular learning algorithms were used: deep learning with convolution neural networks (CNN) and support vector machines (SVM). Each algorithm creates a different classifier that separates waste into 3 main categories: plastic, paper and metal using only 256 x 256 colored png image of the waste. The accuracies of the two classifiers are compared in order to choose the best one and implement it on a raspberry pi 3. The pi controls a mechanical system that guides the waste from its initial position into the corresponding container. However, in this paper we only compare the two machines learning techniques and implement the best model on the pi in order to measure its speed of classification. SVM achieved high classification accuracy 94.8% while CNN achieved only 83%. SVM also showed an exceptional adaptation to different types of wastes. NVIDIA DIGITS was used for training the CNN while Matlab 2016a was used to train the SVM. The SVM model was finally implemented on a Raspberry pi 3 where it produced quick classification, taking on average 0.1s per image.

I. Introduction

Waste sorting is becoming particularly important at house levels [1]. The current trend is to efficiently separate the waste in order to appropriately deal with it. Separating the different elements found in waste streams enables the recovery of useful materials hence, minimizes the large quantity of material sent to landfill and allows recyclable materials to reach its destination. Companies also sort and recycle materials in order to extract value [1]. Hence the need of smart waste sorting is growing. From another perspective, the computational technology is growing and everyday used gadgets are becoming smarter. With their computational power, they can be integrated into an environmentally efficient system of waste sorting.

The Generation of waste has increased dramatically in recent time [2]. If these are not disposed properly, they can have a detrimental affect on the environment. The sorting of waste should be done at the earliest stage possible, in order to maximize the amount of recyclable items and reduce the possibility of being contaminated by other items. Making the trash bin smarter helps in this matter by automatically sorting waste on both home and large scales. In this research we aim at creating such a device by comparing two very popular classifiers CNN and SVM. The classifier that shows the better accuracy is implemented for autonomous waste sorting. The result is a Raspberry Pi controlled trash bin. The smart bin takes a photo of the waste and automatically recognizes its type and redirects it to the corresponding container. Section II contains a review on similar techniques, section III introduces the data collection procedure and preprocessing, section IV introduces briefly CNN and describes the training phase in block diagrams and the network architecture used, section V gives a brief introduction on SVM and the techniques used for training, section VI compares the accuracies of both models, section VII shows the implementation of our model on a raspberry pi and describes the classification procedure, finally we concludes the work with a summary of results, drawbacks and future work.

II. Background

Convolution Neural Networks [3] have had a great impact on pattern recognition [4]. Before the CNN era, features were manually picked and designed and then followed by a classifier. The revolutionary part of CNN is that features are mostly learned automatically from the training data. The architecture of CNN makes it specifically powerful in image recognition. In particular the convolution operation captures the 2D nature of an image. Also the use of a sliding kernel helps in reducing the amount of parameters to be learned by weight sharing. CNN have been in commercial use for over twenty years [5], The adoption of CNN has exploded in the last few years because of two recent developments. First, large, labeled data sets such as the Large Scale Visual Recognition Challenge (ILSVRC) [6] have become available for training and validation. Second, CNN learning algorithms have been implemented on the massively parallel graphics processing units (GPUs) which accelerate learning and inference. A full description of CNN is found in section IV.

On the other hand SVM has been showing state of the art accuracies in many applications, from medical [7] to environmental [8] to many other applications. In short SVM is a minimum risk classifier. SVM always finds the planes that separates the data by leaving the biggest margin possible between both classes. Recently SVM has been combined with convolution neural network in order to make better predictions. SVM has also been recently bundled with the bag of features technique [9] that is used heavily in image recognition. A brief description on SVM and the bag of features technique is presented in section V.

Automatic waste sorting has been the project of many researches. A recent research uses OWL ontology in order to sort waste [10]. However the sorted elements needed to be equipped with RFID in order for the sorting to be accurate. Another approach used optical sensors [11] in order to differentiate textures and colors. However optical sensors are more expensive than a Pi camera and the features needed to be hand crafted in order to make the classification.

In this paper, we present a new usage for CNN AlexNet [12] and SVM bundled with the bag of features technique for automatic waste sorting. The presented methods are an end to end learning approaches of waste classification. There will be no need to craft any features manually, instead the CNN and SVM will be able to discover patterns automatically. We then implement the best model on a raspberry pi 3 which classifies the waste automatically through their image taken by a Pi camera. But before training the models, waste images had to be collected. The data collection procedure is described in the next section.

III. Data Collection

Training data was collected by taking picture of many items. A pi camera is connected to a raspberry pi 3 which implements a simple python script that captures images. They are stored as 256 x 256 lossless png colored images. The system also contains a small preliminary chamber cube shaped, in which the waste is thrown initially. This chamber is lit by 10 LEDs of 1 watt each. When the chamber door opens the LEDs turn on automatically. After the closure of the chamber, the LEDs will be on for 5 seconds during which a picture of the waste is capture automatically from the pi. For the data collection phase, a push button was used to trigger the capturing script. In order to achieve better learning we need to capture images of a variety of waste types. The captured items included plastic water bottles in all shapes and sizes, the bottles were also bent and twisted in order to create more images. Papers of all shapes were also used, fast food cups and boxes, soft drink cans and other type of beverages were also included. Once the image is captured it is placed manually in its corresponding folder (plastic, paper or metal). A total of 2000 pictures were collected: 667 plastic, 667 paper and 666 metal. These images were then split into 3 sets: training, validation and testing. The training set consists of 60% of the data and is used to get the optimal weights for the deep network as well as for SVM. The validation data consists of 20% of the images and are used to get the accuracy of the classifier at every training iteration. The testing set consists of the remaining 20% and are used on the model that achieved the highest validation accuracy in order to report the final accuracy of the model.

In order to reduce overfitting the training set was augmented artificially to 6000 images using label-preserving transformations [13], [14], [15]. The method used is described in [4] and consists of altering the intensities of the RGB channels in training images only. The method applies principle component analysis (PCA) on the RGB components of the training set. Then we add a multiple of the found principle components, with a proportionality factor equal to the corresponding eigenvalue multiplied by a random variable drawn from a Gaussian distribution with zero mean and standard deviation of 0.1. In the next section we describe CNN and how the described training data is used in the training phase.

IV. Convolution Neural Network

In this section we present a brief description of a convolution neural network and highlight its advantages over other techniques. In a regular neural network, every neuron in the input layer is connected to all pixels of the image. Hence an image of size 28 x 28 will result in 784 input weights per neuron. However it is strange to treat all pixels in an image on equal terms. For instance, the network treats pixels that are far away from each other and pixels that are close to each other on the same term. The reason is that the regular neural network does not take into account the spacial structure of an image. Convolution neural networks try to take advantage of spatial structure in an image. These network use a special architecture which is well adapted to image recognition. Convolution neural network are based on 3 ideas: local receptive field or kernel, weight sharing and pooling.

A. Local receptive field

In the regular neural networks the neurons are thought to be stacked in a vertical line. However, in a convolution neural network, the neuron are represented as a 2D matrix of neurons. And instead of every pixel being connected to all neurons of the network, only a small part will be connected. More precisely each neuron of the input layer will be connected for example to a 5 x 5 region in the image corresponding to 25 inputs. The size of the kernel is variable and is chosen based on the validation set accuracy. That region in the image is called the local receptive field. So every neuron is trying to learn weights for its receptive field. We then slide the local receptive field a certain amount of pixels called stride across the image. Every time a stride is taken, the receptive field changes its position and it creates another receptive field connected to a different neuron.

B. Shared weights and biases

We mentioned above that every neuron is connected to a different receptive field, but also all the neurons share the same weights. This leads to the network being able to learn the same thing but across the whole image. So all the neurons will learn how to detect a specific thing but in different parts of the image. This idea is very useful because if a neuron learns to detect a vertical edge in one part of the image, it is also useful to use the same network to detect this edge in another position in the image. This property makes CNN shift invariant. Hence if a paper is detected on the top left part of an image it would be also detected if it happens to be on the bottom right of the image. The problem now is that one convolution layer will learn how to recognize 1 thing in the image. Hence a convolution network is created by stacking many convolution layers on top of each other. So by stacking 3 layers on top of each other the network will be able to learn 3 different things across the whole image. Maybe it will learn how to detect the same edge but in different colors. In practice many more layers are used.

C. Pooling

In addition to convolution layers. the convolution network contains also pooling layers. They are usually used after the convolution layer in order to simplify its output. A popular pooling layer is the “max-pooling” layer. This layer of neuron is made up for example of 2 x 2 map. Hence every 4 neurons from the convolution layer are reduced to a single neuron corresponding to the one that has the maximum output. Thus we obtain a reduced number of outputs by a factor of 4. Max pooling is applied to all the convolution layers. So if the convolution network is 3 x 28 x 28 (3 layers of 28 x 28 neurons) it will be reduced to 3 x 14 x 14 (still 3 layers but 14 x 14 neurons.) Logically a max pooling layer is asking the receptive field if it has detected something and throwing away all other information about where it has been detected. This makes sense because if a feature is detected, its location is not important. This helps reduce the number of parameters needed in subsequent layers.
Multiple CNN layers are combined together to form a deep network. One popular deep network is the AlexNet [4] that is described next

D. AlexNet

AlexNet is a deep network consisting of 7 layers. In total it contains 650,000 neurons, 60 million parameters and 630 million connections. The activation function used is the rectified linear unit (ReLU). The first convolution layer receives a 256 x 256 x 3 image. A total of 96 layers of local receptive fields of size 5x5 are applied to the image with a stride of 1 and a padding of 2. Paddings are used in order to keep the size of the output equal to the size of the input. In other words, a 5x5 window sliding on a 256 x 256 image with a stride of 1 can take 252 x 252 different positions. By padding the image with 2 pixels from all sides will result in a 256 x 256 output hence, the same size as the input. Padding is important in order to go deep in the network, otherwise the number of features will be reduced to an insignificant number after only few layers. So the output is now 256 x 256 x 96. The local receptive field layer is followed by a maxpool layer with kernel size 3 and a stride of 2 with no paddings. The second part is identical to the first convolution network followed by a similar maxpool layer. The third part is also identical to the initial convolution network followed by an identical maxpool layer. The final part is a softmax layer used for data classification. This layer consists of a fully connected layer of 3 neurons. Every neuron is connected to every output from the previous layer. Every neuron of the output layer will have its own output. The network chooses the one having the highest output. Neuron 1 corresponds to plastic, neuron 2 corresponds to paper and neuron 3 to metal. Figure 1 shows the original AlexNet with all coefficients.

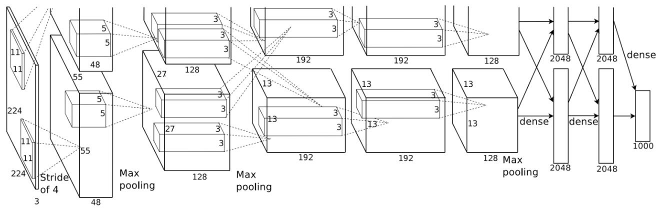

Fig. 1. AlexNet

The advantage of using a CNN resides in the fact that the kernel will act like a filter, sweeping the image from left to right and top to bottom making the classification process shift invariant. Hence if a bottle is captured in the middle of the image or anywhere else in the image it will be recognized by that network. So in summary the convolution neural networks would be learning new feature from pixels. Every part of the network will learn its own features from the previous layer. Finally those features are presented to the softmax layer in order to make the final decision.

E. End to End Training

Training images are fed into the classifier which then computes a proposed class. The proposed class is compared to the desired class for that image and the weights of the CNN are adjusted to bring the CNN output closer to the desired output. The weight adjustment is accomplished using back propagation as implemented in the DIGITS machine learning package. A block diagram of our training system is shown in figure 2. Once trained the network can generate classes from the images of the pi camera as shown in figure 3. Due to the limitation in memory size for the GPU, the images were squashed from 256 x 256 to 32 x 32. This reduction in size will affect the accuracy of CNN compared to SVM that does not have this limitation.

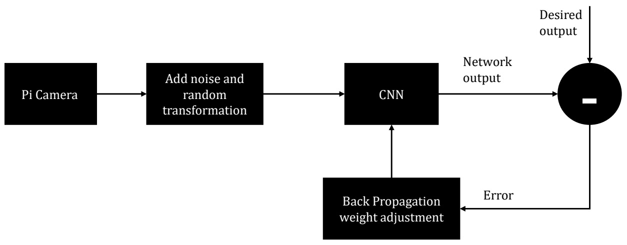
Pi Camera
Add noise and random transformation
CNN
Network output
Desired output
-
Back Propagation weight adjustment
Error

Fig. 2. Training the neural network

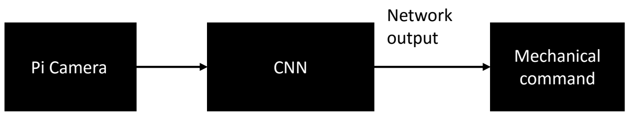

Fig. 3. Testing the neural network

V. SVM and Bag of Features

In this section we introduce SVM and the bag of features technique for image classification.

A. SVM

In the simplest form, SVM uses a linear hyperplane to create a classifier with a maximal margin [16]. In other cases, where the data is not linearly separable, the data is mapped into a higher dimension feature space. This task is achieved using various nonlinear mapping functions: polynomial, sigmoid and Radial Basis Functions (RBF) such as gaussian RBF. In the higher dimension feature space the SVM algorithm separates the data using a linear hyperplane. Not like other techniques, probability model and probability density functions do not need to be known apriori. This is very important for generalization purposes, as in practical situations, there is not enough information about the underlying probability laws and distributions between the inputs and the outputs. Since SVM has been recording state of the art accuracies in many fields, and since it has an excellent generalization ability, it is used also in this research.

What follows is an introduction to the theory of SVM and the general equation of the hyperplane that will separate the two classes. In the case of linearly separable data the approach is to find among all the separating hyperplanes the one that maximizes the margin. Clearly, any other hyperplane will have a greater expected risk than this hyperplane.
During the learning stage the machine uses the training data to find the parameters $\mathbf{w} = [w_1 w_2 ... w_n]^T$ and b of a decision function $d(\mathbf{x}, \mathbf{w}, b)$ given by:

$$d(\mathbf{x}, \mathbf{w}, b) = \mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^{n} w_i x_i + b \tag{1}$$

The separating hyperplane follows the equation $d(\mathbf{x}, \mathbf{w}, b) = 0$. In the testing phase, an unseen vector x, will produce an output y according to the following indicator function:

$$y = sign(d(\mathbf{x}, \mathbf{w}, b)) \tag{2}$$

In other words the decision rule is: if $d(\mathbf{x}, \mathbf{w}, b) > 0$ then x belongs to class 1 and if $d(\mathbf{x}, \mathbf{w}, b) < 0$ then x belongs to class 2.

The weight vector and the bias are obtained by minimizing the following equation:

$$L_d(\alpha) = 0.5 \alpha^T H \alpha - f^T \alpha \tag{3}$$

subject to the following constraints:

$$\begin{aligned} y^T \alpha &= 0, \\ \alpha &\geq 0. \end{aligned}$$

Where H denotes the Hessian matrix given by: $H = y_i y_j (x_i x_j)$ and f is the unity vector $f = [1, 1...1]^T$. Having the solutions α_{0i} of the dual optimization problem will be sufficient to determine the weight vector and the bias using the following equations:

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_{0i} y_i x_i \tag{4}$$

$$b = \frac{1}{N} \sum_{i=1}^{N} (\frac{1}{y_i} - x_i^T \mathbf{w}) \tag{5}$$

where N represents the number of support vectors.
The linear classifier presented above has limited capabilities since it is only used with linearly separable data while in most practical applications data is random and is not linearly separable. The nonlinear data has to be mapped to a new feature space of higher dimension using a suitable mapping function, $\Phi(x)$, which is of very high dimension potentially infinite. Fortunately, in all the equations, this function appears only in the form of a dot product.

From the theory of reproducing kernel Hilbert spaces [17], which is beyond the scope of this paper, a kernel function is defined to be:

$$K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j). \tag{6}$$

By replacing the dot product $x_i \cdot x_j$ by $K(x_i, x_j)$ in all the previous equations, the non-linear hyperplane is determined as:

$$d(\mathbf{x}) = \sum_{i=1}^{l} y_i \alpha_i K(x_i, \mathbf{x}) \tag{7}$$

This remarkable characteristic of the kernel transformation gives the ability for SVM to operate on multi-dimensional data without affecting the processing time. Indeed in the linear case, the processing time is roughly the time needed to invert the Hessian matrix which is of $\mathcal{O}(n^3)$ where n is the number of training points. Since the transformation from the linear to the non-linear case is performed by the simple kernel transformation, the dimension of the Hessian matrix is not changed and hence the processing time is the same, thus its applicability and high performance in multi-dimensional data.

B. *Bag of Features*

The bag of features technique is similar to the bag of words used in natural language processing. An 8 x 8 window scans the whole image from top to bottom in order to find features. This procedure is carried out on the whole training set. Once this is done a vocabulary of different features is generated. Hence every image is described by a vector that has the same length as the vocabulary size, and where every element of the vector shows the number of times that the corresponding feature appeared in that image. In order to reduce the number of features, only the top 80% of the vocabulary are kept in the vocabulary set. Then K-means [18] algorithm is used to cluster similar features together. At the end the features are reduced to the K centroids of the K clusters found by K-means. So every image now is reduced to a k-dimentional vector which is fed to the SVM for training. The value of K as well as the number of features found in our training set are shown in the next section.

VI. Simulation and Results

A. *Training Software*

The training of a huge network like AlexNet or even an SVM with large training set cannot take place on a raspberry pi. It has to be done offline and the resulting model will be implemented on the pi. For training we used an HP laptop equipped with an Intel i7 processor and an NVIDIA 740 M GPU with 2 GB of memory dedicated to the GPU. As for the training software for CNN we used NVIDIA DIGITS which stands for Interactive Deep Learning GPU Training System installed on an Ubuntu 14.04 system. The machine learning library “Caffe” [19] was used to create the network and use it in DIGITS. For SVM we used Matlab 2016a installed also on Ubuntu 14.04. For both models the mean image of all images in the training set was computed and subtracted from all images. This mean normalization step corresponds to a brightness normalization. Hence 2 images that are the same but differ in brightness will be considered as the same image. This will improve the accuracy of detection.

B. *Results and Discussion*

We now present the results for AlexNet and SVM.

1) AlexNet: In order to optimize the weights of the network, mini batch gradient descent backpropagation algorithm was used. Every weight update occurs after the whole batch is sent into the network and the error of the batch is propagated backwards into the weights in order to update them. The batch size used in this study is 100. Usually, a high batch size results in better accuracy and better convergence. However, the batch size is limited by the amount of memory available on the GPU. So 100 was the maximum number of images that we can use in the batch before running out of GPU memory. Every weight is updated by a small portion proportional to the error. The proportionality factor is called the learning rate α and is set to 0.01 in this experiment. A large learning rate will lead to the algorithm diverging, and smaller learning rate will make the convergence very slow. Figure 4 shows the result of the convergence on the training and validation sets. It shows the variation of the accuracy on the validation set, the mean square error loss on the training and validation sets with respect to the number of epochs.

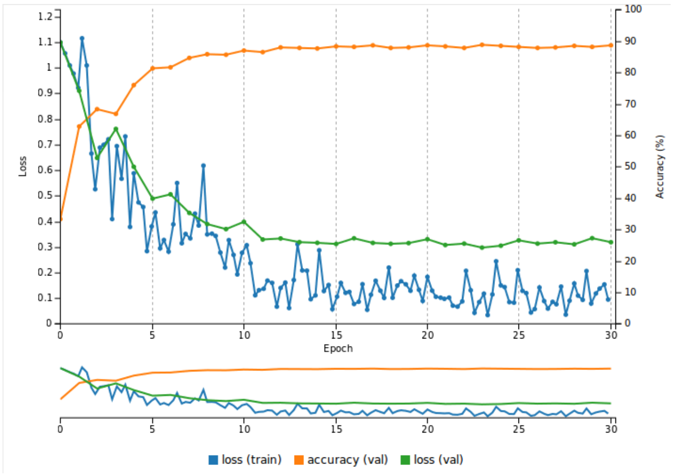

Fig. 4. Convergence

The results show that the highest accuracy achieved during training is 89% on the validation set. The graph also shows that the training error and the loss on the validation data are far from each other. The loss on the training data is 0.1 while the loss on the validation data is 0.3. This is a sign of overfitting which is solved by adding more images to the data set, or by adding more features which corresponds to keeping the image 256 x 256 instead of squashing it to 32 x 32. However due to hardware limitation this was not possible. This issue is a drawback in our research which will be addressed in future work. The team will capture more images and we will also acquire a larger NVIDIA Tesla GPU with 12GB of memory which will reduce overfitting and increase the prediction accuracy. Finally the best model was tested on the testing set and an accuracy of 83% was achieved with 89% detection of paper, 84% detection of plastic and 76% detection of metal.

2) SVM: In order to get the bag of features needed to train SVM, the “bagOfFeatures” function from Matlab was used. This method takes the training set as a parameter as well as the number of clusters for K-means. K usually has a value of 500 however we changed K between 400 and 600 with a step of 10 and calculated the accuracy on the validation set. The K that yielded that highest accuracy is 500 and was used on the test set. Once the bag of features is done the SVM training was performed using the "trainImageCategoryClassifier" function. This function returns an SVM model that can be applied to the test set by using the function "evaluate". The test set accuracy was 94.8% with a 99% detection of paper, 96% detection of plastic and 90% detection of metal.

Hence the two models achieved good accuracies with SVM achieving the best accuracy so far. However, more images and more GPU memory in the future will favor CNN because it will reduce the effect of overfitting that was observed. Since SVM model achieved a better accuracy we implemented this model on the raspberry pi 3 described in the next section.

VII. Hardware implementation

The above SVM model was implemented on a raspberry pi 3 connected to a high definition pi camera. To start, the hardware support package for raspberry pi in Matlab was acquired and the special pi firmware was downloaded and installed on the pi. During training the bag of words structure was saved along with the category classifier in a ".mat" file. The file size was 114 kb which can fit perfectly on the pie. A Matlab script was written to wait for a push button to be pressed and then capture an image. Once the button is pressed, the camera takes a snapshot of the waste and saves it in a 256 x 256 colored lossless png file. The image is sent to the preloaded category classifier for classification and the class is outputted in the form of a LED. Every class is represented by a LED that is turned on based on the decision. The image classification was performed quickly with an average classification time of 0.1 second with a standard deviation of 0.005s.

VIII. Conclusion and future work

In this research we presented a comparison of deep learning and SVM for waste sorting. AlexNet was able to achieve good classification accuracy of 83% however SVM bettered it with 94.8% on the test set. The SVM model was implemented on a raspberry pi and demonstrated the ability to sort waste by their images only. The drawback in our study was the small amount of images in the training set. In the future we will increase the amount of images and their variety. Another technical drawback was the low GPU memory available for this study. The 2GB of GPU memory obliged us to scale down the image from its original 256 x 256 size to 32 x 32. This reduction also contributed to more overfitting problems. This issue will be solved in the future by getting a dedicated machine learning server with 2 Tesla GPUs of 12GB memory each. These GPUs will enable us to use the resolution of 256 x 256 while keeping 100 images as a batch size. Finally the implemented model had very low average execution time (0.1s) on a raspberry pi 3.

Acknowledgment

The authors would like to thank the dean of the faculty of engineering at St Joseph university for providing us with the material needed for this study as well as ordering the new deep learning server with multiple Tesla GPUs.

References

[1] C. Capel, "Waste sorting - a look at the separation and sorting techniques in todays european market," *Waste Management World*, 2008.

[2] P. B.-T. C. K. Daniel Hoornweg, "Environment: Waste production must peak this century," *Nature*, 2013.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[5] L. D. Jackel, D. Sharman, C. E. Stenard, B. I. Strom, and D. Zuckert, "Optical character recognition for self-service banking," *AT&T technical journal*, vol. 74, no. 4, pp. 16–24, 1995.

[6] A. Berg, J. Deng, and L. Fei-Fei, "Large scale visual recognition challenge (ilsvrc), 2010," *URL http://www. image-net. org/challenges/LSVRC*, 2010.

[7] G. E. Sakr, I. H. Elhajj, and H. A.-S. Huijer, "Support vector machines to define and detect agitation transition," *Affective Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 98–108, 2010.

[8] G. E. Sakr, I. H. Elhajj, and G. Mitri, "Efficient forest fire occurrence prediction for developing countries using two weather parameters," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 5, pp. 888–894, 2011.

[9] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 524–531.

[10] A. Sinha and P. Couderc, "Using owl ontologies for selective waste sorting and recycling," in *OWLED-2012*, 2012.

[11] J. Huang, T. Pretz, and Z. Bian, "Intelligent solid waste processing using optical sensor based sorting technology," in *Image and Signal Processing (CISP), 2010 3rd International Congress on*, vol. 4. IEEE, 2010, pp. 1657–1661.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[13] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *null*. IEEE, 2003, p. 958.

[14] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[15] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "High-performance neural networks for visual object classification," *arXiv preprint arXiv:1102.0183*, 2011.

[16] V. Kecman, *Learning and soft computing: support vector machines, neural networks, and fuzzy logic models*. MIT press, 2001.

[17] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American mathematical society*, vol. 68, no. 3, pp. 337–404, 1950.

[18] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, no. 14. Oakland, CA, USA., 1967, pp. 281–297.

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.