UNSUPERVISED DEEP EMBEDDING FOR CLUSTERING AND ANOMALY DETECTION FOR FOOD INDUSTRY AUTOMATION

Oskar Hint (s161559)
Danmarks Tekniske Universitet

Valentin Skokov (s161070)
Danmarks Tekniske Universitet

ABSTRACT

Currently in food industry most quality assurance tasks are performed manually by workers. Since most of these tasks involve visual assessment they are good candidates to be automated through machine learning. This project investigates an unsupervised deep learning approach for clustering and anomaly detection on images of pork chops. Clustering based on Unsupervised Deep Embedding for Clustering Analysis was applied, and modifications to the base algorithm for anomaly detection were implemented. Clustering approach did not achieve significant performance, 2-class anomaly detector achieved 10% accuracy increase over the naive baseline. Results suggest that for this problem an unsupervised approach is unlikely to achieve comparable performance with supervised methods.

1. INTRODUCTION

Sensomind [1] is a Danish startup 'Powering up Manufacturing Facilities with Artificial Intelligence'. One such manufacturing facility is a food production line where currently removing items which do not meet certain quality standards is typically a task performed by humans. This is a tedious process that requires many manhours and the speed of the line can be also be limited by the speed with which workers can reliably and continuously inspect it. The solution Sensomind is aiming to provide is an automated system consisting of a hardware module with a camera, placed right above production line, and software that is able to detect abnormalities fast and reliably.

1.1. Dataset

One example of a food production line, and also the one we are working within this project, is a meat conveyor line. Our dataset consists of 385 separate images of pork chops, originally 400 $\times$ 400 in size. Furthermore, the images are divided into 3 classes: 'normal' - 210 images present, 'glove' - 65 images, and 'cut' - 110 images. As evident from the names, the first class represents pieces of meat that comply with quality standards while second and third class do not. For the second class, the pork chops have some foreign material (plastic) on them, and for the third one, the chops are cut too much meaning they do not contain the amount of fat tissue required. Examples of all 3 classes can be seen in Figure 1.

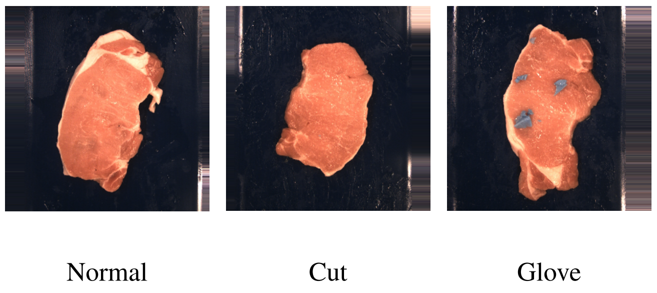

Fig. 1. Exemplary samples of normal, cut, and glove class pork chops

2. THEORETICAL APPROACH AND ALGORITHM OF CHOICE

Since collecting large representative labelled datasets for food industry quality assurance is very time-consuming and complicated, it would be ideal if the quality assurance problem can be reasonably addressed through unsupervised means.

The unsupervised approach we elected is based on 2015 paper by Junyuan Xie, Ross Girshick , and Ali Farhadi [2], more specifically we followed the steps described there and tried to fit the approach to the task at hand.

Therefore, we would like to give acknowledgements to them and their work, that enabled us to look at the task from an interesting perspective. When the following text mentions 'original paper', it should be referred to their work. Furthermore, this section in particular and the whole project in general is heavily based on their paper, with addition of remarks of our own experience with it.

3. DEEP EMBEDDED CLUSTERING (DEC)

Consider the classical task of clustering n points $\{x_i \in X\}_{i=1}^{n}$ into k clusters. There are multiple algorithms and solutions for this task, however, when dealing with high-dimensional data like images, dimensionality reduction must be applied, as it is excessively unwise to operate in space *data space X*. Classical methods like PCA and ICA are well known and widely used, but are far from perfect, and therefore we would like to get a better non-linear data transformation $f_\theta : X \rightarrow Z$, where θ is learnable parameter(s). Neural networks are a natural choice, due to having being shown to be able to efficiently approximate any function and due to their capability of feature learning.

Original algorithm consists of 2 phases: (1) parameter initialization by finding an efficient transformation f_θ and feature space Z and (2) clustering and parameter optimization in the new feature space, where we use an auxiliary target distribution and minimize the KullbackLeibler (KL) divergence to it. First, we will give a small explanation of phase (1) as it proved to be much more important for achieving accurate clustering.

3.1. Parameter initialization with SAE

We search for transformation f_θ using a stacked autoencoder (SAE), as this method has been recently shown to be able to achieve efficient and meaningfully separated representations of the real-world datasets [3].

The idea behind SAE is to first pretrain the weights of each layer by training them separately as denoising autoencoders, trained to reconstruct the previous layers output after random corruption. A denoising autoencoder is a 2-layer NN given by:

$$\tilde{x} \sim \text{Dropout}(x) \tag{1}$$

$$h = g_1(W_1 \cdot \tilde{x} + b_1) \tag{2}$$

$$\tilde{h} \sim \text{Dropout}(h) \tag{3}$$

$$y = g_2(W_2 \cdot \tilde{h} + b_2) \tag{4}$$

where *Dropout(x)* is a function that randomly sets a portion of x's input dimensions to 0 and g_1 and g_2 are respective activation functions. We use ReLU activation everywhere, except for the first layer, that uses the original data as it's input. training is performed by minimizing $\left\|x - y\right\|_2^2$.

After performing such layer-wise pretraining we concatenate all encoder layers followed by all decoder layers, in reverse layer-wise training order, to form a deep autoencoder and then finetune it to minimize reconstruction loss. The final result is a multilayer deep autoencoder with a bottleneck coding layer in the middle. After that we use the encoding half of the resulting SAE as our data initial transformation f_θ. Finally, to initialise cluster centers for the next phase, we feed the data into encoder to get it's feature space representation and run *kmeans*, obtaining *k* centroids μ_j in feature space Z.

3.2. Clustering with KL-divergence

With initial estimate of the mapping f_θ and centroids $\left\{\mu_j\right\}_{j=1}^k$ we then proceed to the next phase, that consists of 2 steps: first, we compute soft assignment between centroids and embeded points, and second, we update function f_θ and refine centroids by learning from high confidence assignments using an auxiliary target distribution. Process is repeated until convergence (or after exhausting resources given for the task, mainly time).

3.2.1. *Soft assignment*

Using *Student's t-distribution* as a kernel following measure of dissimilarity between embedded point $zi = f_\theta(x_i | x_i \in X) \in Z$ and cluster centroid μ_j:

$$q_{ij} = \frac{(1 + \left\|z_i - \mu_j\right\|^2)^{-\frac{1}{2}}}{\sum_{j'}(1 + \left\|z_i - \mu_{j'}\right\|^2)^{-\frac{1}{2}}} \tag{5}$$

This can be treated as a probability that point i belongs to a cluster j.

3.2.2. *KL divergence and auxiliary distribution*

DEC's next step is to refine clusters by learning from high confidence assignements. Auxillary distribution P used for the task id given by:

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j'} q_{ij'}^2 / f_j'} \tag{6}$$

Where $f_j' = \sum_i q_{ij'}$. With this distribution calculated, we can find KullbackLeibler divergence that is also used as loss function:

$$L = \text{KL}(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{7}$$

In-depth discussion on selection of auxiliary distribution and reasons for use of KL divergence are to be found in the original paper.

3.2.3. *Optimization*

Using KL divergence as a loss function, we then update both centroids μ_j and encoder with Stochastic Gradient Descent. Gradients of our L are given by:

$$\frac{\partial L}{\partial \mu_j} = -\frac{1}{2} \sum_i (1 + \left\|z_i - \mu_j\right\|^2)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j) \tag{8}$$

$$\frac{\partial L}{\partial_i} = \frac{1}{2} \sum_i (1 + \left\|z_i - \mu_j\right\|^2)^{-1} \times (p_{ij} - q_{ij})(z_i - \mu_j) \tag{9}$$

Gradients $\frac{\partial L}{\partial_i}$ are then passed down to encoder that is optimized using standard backpropagation. The procedure is then repeated until either one of the following is met: 1) reached set limit of iterations, 2) less than a specified % of points change clusters between 2 successive iterations.

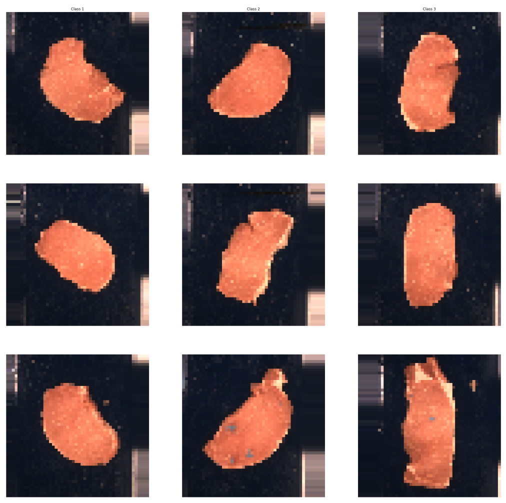

Fig. 2. Examples of classes (columnwise) as clustered initially

3.3. Encoder structure

As always the case with Deep Nets, number of layers and nodes is a parameter to choose values for. As it stands now, we did not change the structure from the one used in the original paper [2], that being $d - 2000 - 500 - 500 - 10$ where d is the number of input dimensions. For future work however, encoder structure could possible be changed for better effect.

4. EXPERIMENTS AND RESULTS

4.1. Initial implementation of unmodified algorithm for 3-class problem

We started by evaluating the aforementioned method of obtaining a new feature space Z and clustering using KL-divergence on our dataset of 3 classes. To obtain a metric of evaluation the ground truth labels were compared to the cluster assignments, this yields the unsupervised clustering accuracy. Without any modifications to the original method the obtained clustering accuracy was not any better than naively predicting everything to be normal which yields 54.5% accuracy.

Inspecting the clusters it was apparent that the clustering is based mostly on the orientation of the pork chops - whether they were positioned straight or angled towards left/right, and the size of the pork chop, as can be seen in Figure 2, where each column represents items in the same cluster according to the model. We tried to address the issue via data augmentation to generate a dataset of images where the orientation is

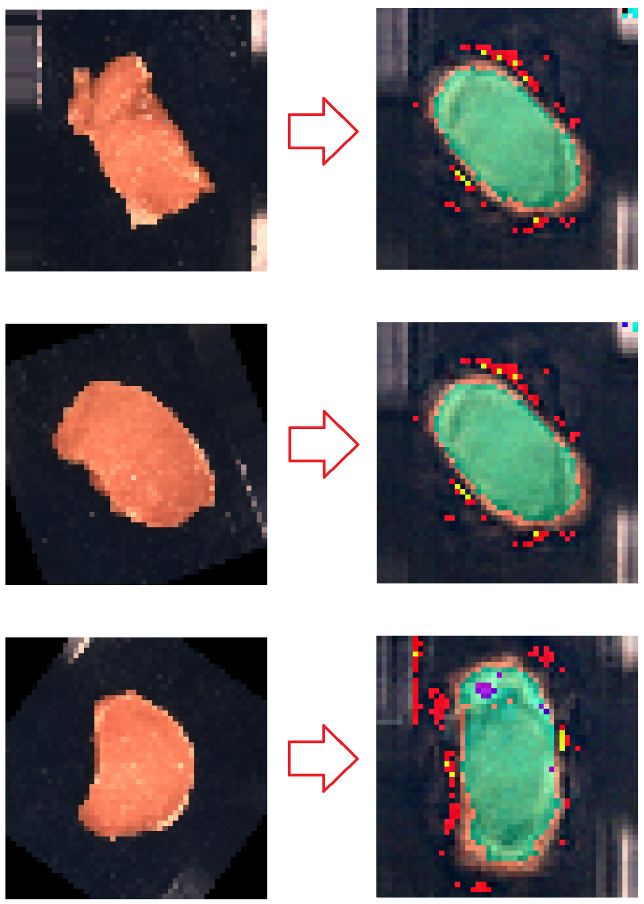

Fig. 3. Examples of encoder transforming image into a cluster prototype

sampled uniformly from 360°. However, this effectively just changed the soft cluster assignments on our dataset but did not improve the clustering accuracy.

Further inspecting the clusters we noticed that the glove class images were essentially equally likely in all clusters, i.e. likely the smaller features containing information on the presence of plastic were overpowered by more significant features such as size and orientation. Since intuitively size features should help discriminate between normal and cut classes (since cut pieces tend to be significantly smaller) but not between the first 2 classes and glove class, we decided to shift our focus towards the easier 2-class problem of differentiating normal images from cut.

4.2. Modified algorithm for 2-class problem

For the 2-class problem the naive baseline (everything is normal) accuracy is 65.6%. Applying the original method yielded a slightly better accuracy of around 68%. However, we noticed that the accuracy typically got slightly worse as the KL-divergence-based optimization processed. This was because the features which better differentiate the clusters in an unsupervised setting do not correspond well to the features which are important in terms of our clustering accuracy. Moreover, the iterative KL-divergence optimization changes the SAE in such a way that the reconstructions (and therefore likely also the hidden/bottleneck representations) look more like cluster prototypes, see figure 3. This means the negative effect caused by the optimization on our clustering accuracy is amplified.

We thus decided to discard the iterative clustering process from the original paper and just do *kmeans* and use eq. 5 to obtain soft assignments, although this last step is strictly not needed if we are just interested in the clustering accuracy. Furthermore, we observed that fitting a single *kmeans* cluster and interpreting everything that is some distance away from it as belonging to the other class yielded better performance than fitting 2 clusters. This anomaly detection scheme achieves around 74-75% accuracy depending on the choice of threshold distance/probability.

Inspecting the assignments from our anomaly detection model we observed that prototypical normal pork chops (see figure 1) were generally well identified. The somewhat low accuracy stemmed from the fact that in the normal class of pork chops there are many samples which actually look a lot like cut pork chops. One of the more extreme cases of this can be seen in figure 4, even though the food items seem quite similar they belong to different classes. So for our anomaly detector the number of non-typical normal pork chops is too high to achieve better performance.

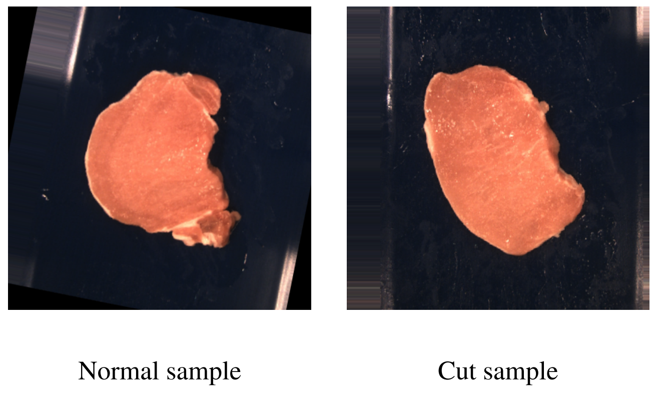

Fig. 4. A normal and cut pork chop which both have high probability of belonging to the anomalous (i.e. cut) class according to the 2-class detector

5. DISCUSSION AND FUTURE WORK

The results for purely unsupervised approach show that the task at hand is more difficult than initially expected. Consider again figure 4, a sufficiently good classifier or anomaly detector would need to deem the normal sample closer to the prototypical normal sample in figure 1 than to the very similar looking cut sample in figure 4. To do this, the model would need to give more importance to specific features (in this case features related to fat content) but in a purely unsupervised setting there is no information available to make this decision. Therefore, a good idea for future works is to add supervised elements, like using pre-defined (supervised) auxiliary distribution for KL-divergence optimization part.

Other means of data augmentations besides the tried out rotation can be implemented, as well as ignoring features in space Z that have large variance or deemed unnecessary by any other means, since those represent typically size/orientation, and distort the outcome of clustering. Also, even when devoting significant time and effort on an unsupervised approach, the accuracy will likely be nowhere near as good as for (semi)supervised, even when accounting for the small size of the labelled dataset.

6. CONCLUSION

Overall results of our findings show that a fully unsupervised approach might not be able to produce meaningful results, while some elements of it, like dimensionality reduction via SAE should be employed further. With 3-class problem we have observed multiple problems with applying the original algorithm, some of which are inherent to the unsupervised nature of DEC, and overall low accuracy. We achieved about 10% increase in unsupervised clustering accuracy over the naive baseline for the 2-class problem of differentiating cut and normal pork chops. For further improvements we suggest employing a supervised or semi-supervised approach.

All of the relevant code can be found at dedicated GitHub repository [4].

References

[1] *Sensomind ApS*. URL: https://www.sensomind.com.

[2] Ali Farhadi Junyuan Xie Ross Girshick. *Unsupervised Deep Embedding for Clustering Analysis*. URL: http://proceedings.mlr.press/v48/xieb16.pdf.

[3] G. E. Hinton and R. R. Salakhutdinov. *Reducing the Dimensionality of Data with Neural Networks*. URL: https://www.cs.toronto.edu/~hinton/science.pdf.

[4] Oskar Hint & Valentin Skokov. *Project's github repository*. URL: https://github.com/oh54/02456-Deep-Learning-project.