Handwritten Digit Recognition in Noisy Scenarios

Data Mining Final Project

Jinmei Zheng
University of Pittsburgh
jiz155@pitt.edu

Chunyi Wang
University of Pittsburgh
chw165@pitt.edu

ABSTRACT

This course project focus on the problem of recognizing handwritten digit in a noise scenario. To solve the problem, we sequentially apply image processing techniques, feature engineering, classifier selection, parameter tuning, as well as training neural network. Theoretical analysis and practical experiment results are contained, which enhance what we learned in Machine Learning course.

KEYWORDS

Digit recognition, neural network, feature engineering, image processing

ACM Reference Format:
Jinmei Zheng and Chunyi Wang. 2018. Handwritten Digit Recognition in Noisy Scenarios: Data Mining Final Project. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Digit recognition has been considered as a classical computer vision problem in the literature. For better understanding what we learned in Machine Learning class, we hereby explore this classical problem in this course project. To mimic a more practical scenario, we manually add noise into MNIST dataset and test our proposed classifiers by these noisy images. We therefore refer this application as noisy digit recognition task. In this project, we sequentially exploit image processing techniques to remove noise, feature engineering to extract features, model selections and parameter tuning to train classifiers. Moreover, we also explore the utility of modern neural network at the end of our project, where we implement well-known LeNet and from it create our novel architecture. Experiment demonstrate that our designed neural network achieves the highest accuracy, 95%, against all others in this noisy digit recognition task.

2 RELATED WORK

The early algorithms on classifiers methods, *K Nearest Neighbor, Support vector machine, and Back-Propagation Network* [10], sought to solve Handwritten Digit Recognition. However, such approaches eventually lead to different accuracy, time and space complexity.

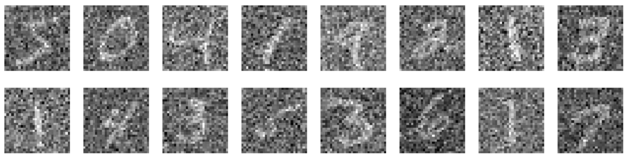

Figure 1: Our project is focusing on recognizing digits from noisy images.

2.1 K Nearest Neighbor

The measure of similarity, which is also referred to distance, play an important role in KNN algorithm. The direct implementation is to choose *Euclidean* distance [10] which is simple but works poorly on several cases, for example, the misalignment between the same images. Some of the other improvements like Tangent Distance are introduced by [11]: *Tangent Distance* can be immune from the shift change of images by making the image dimension to be identical to the number of pixels, and then project the original figure in the pixel space. Since KNN always have the worst performance among the algorithms, it is often regarded as a baseline algorithm for the comparison of the other improved algorithms on digit recognition.

2.2 Support vector machine

The key of the SVM algorithm is its kernel function. One common trick on dealing with kernel function is called *kernel-generated jitter* [7]. To use different kernel functions to generate converted images and choose the most favourable kernel function for the group of converted images. The other way is called *virtual SVM* [9]. It works well by leveraging the result of a initial SVM and gain transformed support vectors called virtual support vectors to do the next trainings. In the paper,for each support vector the author choose a total of 13 virtual vectors, that is, neighborhood vectors, horizontal and vertical vectors.

2.3 Back-Propagation Network

One of the most famous algorithm on digit recognition under a neural network framework is Fully Connected Multilayer [8] Comparing to the baseline linear classifier which is only dependent on input and output, the larger fully connected multilayer classifier applies hidden layers with much more neurons between input and output layer. The number of neurons of different layers can be tuned case by case. The other popular approach is called Convolutional Neural Networks(LeNet) [8], which developed by Dr. Yann LeCun. There are several versions of LeNet: LeNet1, LeNet4, Boosted LeNet4 and LeNet 5. LeNet uses nonlinear convolutional layer to map the input image to the features and then uses subsampling/pooling to extract

Conference'17, July 2017, Washington, DC, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

more relevant features. In the subsampling layer, the most useful method is max-pooling [5]. Max-pooling separates the previous input layer into a set of disjoint boxes, and output the maximum value of each sub box. LeNet4 has 4 times more feature mappings and 1 more fully connected layer than LeNet1.

2.4 Bilateral Filter[4]

Bilateral filter is a non-linear filter, used for smoothing images and keeping edges sharp simultaneously. The filter is basically a multiplication between two gaussian filter. Formally, the filtering proces s is represented by

$$I^{\text{filtered}}(x) = \frac{1}{W_p} \sum_{x_i \in \Omega} I(x_i) f_r(||I(x_i) - I(x)||) g_s(||x_i - x||) \tag{1}$$

where the normalization term is

$$W_p = \sum_{x_i \in \Omega} I(x_i) f_r(||I(x_i) - I(x)||) g_s(||x_i - x||) \tag{2}$$

The reason why this filter is capable to keep edges and smooth images is its consideration on both color domain and space domain.

Although there are many popular methods and models proved to be good solvers of Handwritten Digit Recognition on MNIST dataset, most researchers donâĂŹt pay attention to image processing and noise removal on real world dataset when the digit images have random spot or stain. Thus, our goal is to find a efficient digit classier for a noisy MNIST dataset to mimic a more practical scenario by manually adding noise perturbation and expect it to a certain extend fill the feature gap between MNIST and real world images.

3 NOISY MNIST DATASET

To apply machine learning techniques to solving the handwritten digit recognition problem, we utilize a commonly used dataset MNIST [10]. We understand that the digit images in MNIST are pre-processed beforehand including denoising, alignment, rotation, centering, and more. To mimic a more practical scenario, we manually add noise perturbation and expect it to a certain extend fill the feature gap between MNIST and real world images. Specifically, we add Gaussian noise with magnitude 0.5 to the MNIST images. Figure 2(top) visualizes the original MNIST images and Figure 2(bottom) shows the perturbed images. All experiments in our project are based on these noisy images. Our goal is to classify these noisy images into ten classes from zero to nine.

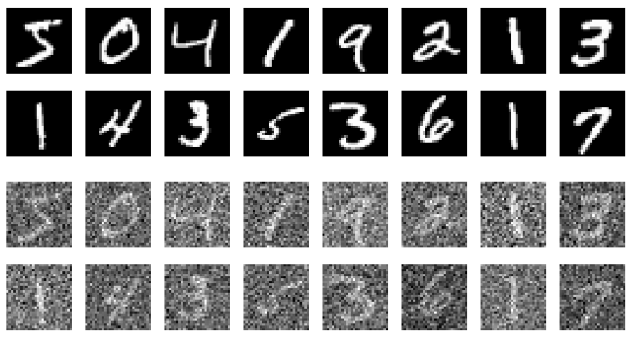

Figure 2: (Top) Original MNIST images. (Bottom) Noisy MNIST images.

4 BASELINES

For future comparison of our implemented algorithms, we utilize some naive implementation as our baselines. They are multi-class logistic regression and k nearest neighbor classifier. As for the feature extraction, we directly feed raw pixels into models, specifically by reshaping them into a one-dimensional vector, normalizing to [0, 1], and directly feeding these pixel intensity into the proposed baseline models. We summarize the accuracy into Table 1.

Logistic regression	KNN
83.53%	86.37%

Table 1: The accuracy of applying baseline methods—logistic regression, KNN—to noisy images.

5 IMAGE PROCESSING

Before extracting features from images, we believe some image processing techniques are inevitable in our case, especially removing noise. We first remove the noise and sharpen the images simultaneous, then transform the gray scale images into binary ones (black and white) and finally apply filters to make digits thinner. To evaluate if the each step contributes to our problem of digit classification, we apply the logistic regression and KNN to images after each step, and summarize the accuracy in Table 2.

	Noisy image	Noise removal	Binary	Erosion
Logistic regression	83.53%	85.47%	85.91%	82.68%
KNN	86.37%	91.50%	89.68%	85.05%

Table 2: The accuracy of applying logistic regression, KNN to images after each image processing steps.

5.1 Noise removal and sharpen

We explore several filtering methods including gaussian blurring [1], first derivative of gaussian filtering [2], Laplacian filtering [2], and etc. to remove noise, smooth background and simultaneous keep boundaries sharp. In our experiments we finally found that applying bilateral filters [4] to images achieves the best performance which removes noise and at the same time keep the boundaries as sharp as possible. Technically, after parameter searching, we select to use a bilateral filter with kernel size 12 pixels, variance 2.5 in color space and variance 1.5 in pixel space. Figure 2 visualizes the images before and after applying the bilateral filters. One can see that images after filtering are smooth at background and the major information (digits) are well kept at foreground. We also re-run the baseline models on these processed images. The accuracy are summarized in Table 2. One can see that the accuracy increases significantly which demonstrates the effectiveness of this image processing step.

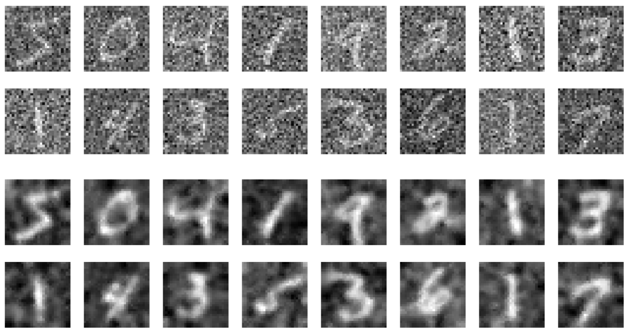

Figure 3: (Top) Noisy MNIST images. (Bottom) MNIST images after noise removal.

5.2 Transforming to binary images

We manually select a threshold such that any pixels with intensity less than the threshold will be removed to zero while pixels with intensity greater and equal to the threshold will be lifted to one. We expect this step could remove noise from images and remain the useful information. After manually searching between [0, 1], we finally select 0.32 as our threshold which roughly satisfies our goals. Figure 3 compares the gray scale images and binary images. One can see the majority of noise at background are removed by this step. We re-run the baseline models on these binary images to demonstrate the effectiveness. The accuracy are summarized in Table 2 where one can see that the accuracy of logistic regression increases after transforming to binary images while the accuracy of KNN decreases. This might be caused by the information loss after transforming to binary images.

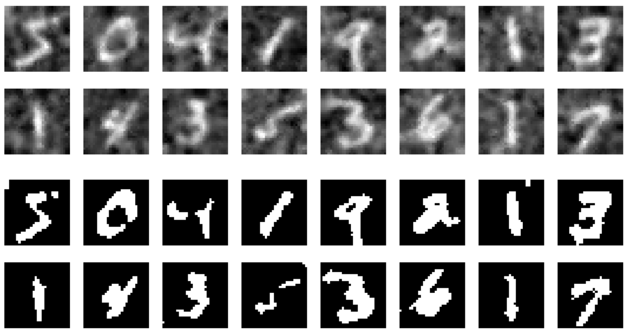

Figure 4: (Top) MNIST images after noise removal. (Bottom) Bianry MNIST images.

5.3 Thinning digits

We suppose that the width of digits are irrelevant to what the digit is. Based on this insight, we use morphological erosion [12] to make digits thinner and expect the thinner digits will build up a deeper feature gaps between different digits. In our experiments, we utilize a 2 x 2 pixel filter filled with one to apply erosion transformation. Figure 4 shows the digit images before and after erosion. One observes that eroded digits are clearly thinner, which proves that our implemented erosion transformation works well. We re-run the baseline models on the eroded images. Results are summarized in Table 2. Unfortunately, the accuracy decreases because thinning digits by erosion might break some thin parts and cause unnecessary broken digit.

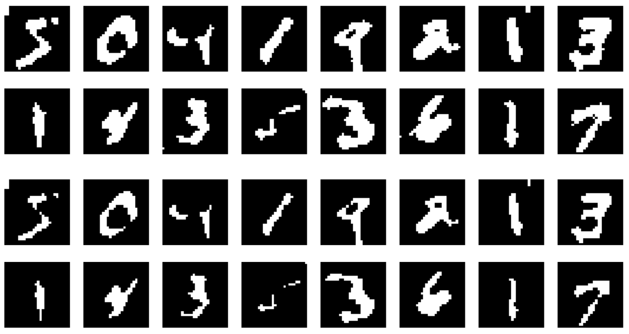

Figure 5: (Top) MNIST images after noise removal. (Bottom) Bianry MNIST images.

6 FEATURE EXTRACTION

After image processing, we extract features from images. From what we learned on the class, we first explore to use Principal Component Analysis (PCA). For better performance, we explore more image-specific features: histogram of oriented gradients and Harris corner features.

6.1 Principal component analysis

Our first feature method is PCA. We search the number of components, linearly from 2 to 300. The results in our experiments are summarized into Table 3. For time efficiency, we use logistic regression model in this experiments. One can see that the highest accuracy is achieved at 101 when feeding into original images, 101 when feeding into cleaned images, 134 when feeding into binary images, and 167 when feeding into eroded images. To compare the effect of above proposed image processing steps, we also compare among images, and found that logistic regression gets the best performance when feeding in PCA features with 134 components extracted from binary images. This demonstrates again the effectiveness of our image processing part. Moreover, from the perspective of using PCA, we also realize that PCA helps to improve the performance in our case.

# of components	Noisy	Clean	Binary	Erosion
2	42.19%	32.95%	35.73%	33.27%
35	84.33%	83.38%	85.22%	82.02%
68	85.15%	84.98%	85.98%	82.99%
101	85.16%	85.41%	86.33%	83.13%
134	85.16%	85.25%	86.44%	83.11%
167	85.09%	85.35%	86.37%	83.24%
200	84.87%	85.31%	86.35%	83.08%
233	85.09%	85.34%	86.26%	83.02%
266	84.78%	85.37%	86.24%	82.99%
300	84.94%	85.19%	86.08%	82.97%

Table 3: The accuracy of applying logistic regression, KNN to images after each image processing steps.

6.2 Histogram of oriented gradients

Histogram of Oriented Gradients (HOG) [3] is a feature descriptor commonly used in computer vision on images especially for object detection task. It is computed by densely counting the occurrences of oriented gradients in localized portion of images. For a global description of images, these positions are set in a dense grid uniformly. We use this descriptor to extract features in our noisy digit recognition task. As for the parameterization, due to its time complexity, we simple select some values from experience without tuning parameters. However, due to its effectiveness, we observe that HOG works best in almost all classifiers.

6.3 Harris corner features

Harris corner feature [6] is a descriptor used to detect corners. By applying Harris corner transformation to an image, we get a two-dimensional features with the same size as the image, where the high values in features indicates high possibility of corners at that position. We apply this to every single images, reshape the features to a vector, and then feed into various models. However, we found that this feature works worst compared to PCA or HOG. A potential reason for this is that solely using corner information is not sufficient in our noisy digit recognition task. Concatenating Harris corner features with other features might help improve the performance, which we shall remain as future work.

7 CLASSIFIERS

After feature extraction, we gain four kind of feature extraction set ,they are, raw binary image, after-HOG image,PCA features and HCF features. In order to decide our final model , we try several classifier models include SVM, D-tree, LDA, QDA and ensemble methods such as random forest and XGBoost.We experiment those four feature extraction set on our model to decide our final feature set and final selected model.

Support vector machine: As the paper [9] said, support vector machine model is a practical way to do digit recognition.We experiment those four feature extraction set on support vector model and after tuning parameters, the best performance model has the following parameters set : kernel='rbf', gamma=0.1, C=10. For the four feature descriptors, HOG set has the best accuracy of 0.9256 while HCF has the lowest accuracy of 0.6342. It shows that corner detector is not a good way to classify digits, as its performance is even worse than our baseline model. Though the average accuracy is enough high, it takes hours for SVM to train model and predict classes of the testset since our dataset has a huge amount of instances and the large number of dimension is not suitable for SVM.
Decision tree: We also use k fold cross validation to train several decision tree models on 4 feature descriptors.We tuned the decision tree parameters MaxDepth from 10 to 20 and to gain the best model with MaxDepth = 15 and the accuracy = 0.7761 when we applied on after- PCA dataset. In decision tree models, Binary dataset have the highest accuracy while PCA has a accuracy of 0.7761 as it loss less important information.

Linear discriminant analysis: LDA is a linear decision boundary classifier which has assumptions that the density function of input features in each class is normal and the variance of features is equal in all classes. In our case, LDA with after -HOG(Histogram of oriented gradients) dataset outperforms than LDA models with other three feature descriptors at the accuracy of 0.8818.

Quadratic discriminant analysis: QDA has a quadratic decision boundary and it assumes that the density function should also be normal distribution for every class and the variance can varies from class to class. In our experiments, QDA model with HOG dataset has the highest accuracy .

Random Forest Classifer: To improve our models, we also implement two ensemble methods, random forest classifier and XGboost classifier. random forest classifier is a ensemble method to reduce the variance of a single decision tree model by randomly select subset of features in trees and average the result of several trees. Unlike other model, In random Forest model , the best feature descriptor is binary raw image, instead of HOG, with a result of 0.9217 accuracy.

XGBoost Classifier: XGBoost is an improved gradient boosting algorithm that provides a parallel tree boosting to solve many problems in a fast and accurate way. XGboost can avoid overfitting as it used a regularized model and we can set the regularization parameters as we train models. After cross validation, we set parameters learning rate=0.5, n-estimator=100 and alpha = 10 to gain our XGboost models. It turns out that HOG dataset has the best accuracy of about 0.9256. Since XGBoost applies parallel tree boosting and have made use of most memory, it takes less time than random forest classifier and SVM.

Experiments: We conduct extensive experiments using all classifiers we mentioned in this section. To compare the performance of different features, we apply these classifiers with all extracted features respectively. For better performance, we tune parameters for classifiers decision tree, random forest and xgboost. For others (e.g. SVM), we select parameters from experience, due to their high time complexity. We summarized the accuracy using best parameters in the table 4 that contains SVM, Decision Tree, LDA, QDA, Random Forest and XGB. One can see that the model SVM using HOG features gets the highest accuracy—0.9256—beating all others.

	SVM	Decision Tree	LDA	QDA	Random Forest	XGB
RAW	77.07%	80.84%	82.57%	64.58%	92.17%	91.98%
PCA	91.75%	77.61%	82.80%	88.68%	90.47%	90.15%
HOG	92.56%	79.70%	88.18%	88.70%	90.55%	92.31%
CORNER	63.43%	59.80%	59.97%	10.32%	82.97%	86.76%

Table 4: The results summary of all classifiers.

8 NEURAL NETWORK

After feature engineering and model fitting, we want explore the performance of modern deep neural network on our noisy digit recognition task. Specificially, we use TensorFlow—a widely used deep learning packages—to implement, train, and test our models. Moreover, to facilitate fetching data and maintain the consistence of noise perturbation, we create our own dataset in TFRecords format instead of using existed MNIST TensorFlow dataset.

8.1 LeNet-5

We first implement a well-known neural network called LeNet-5 [10]. Its architecture is summarized in the Figure 6. One can see it starts with two 5×5 convolution layers, each followed by a 2×2 max pooling layers and ends with three fully connected layers. All layers are using ReLU activation function. This architecture gives a quite nice results summarized in Table 5. One can see that it outperforms almost all classical classifiers using manually designed features. However, due to the difficulty introduced by noise, the results is still lower than 95%.

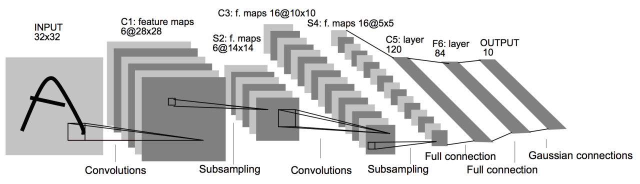

Figure 6: Architecture from LeNet-5.

8.2 Modified LeNet

To improve the performance, we increase the number of channels in each convolution and fully connected layers. This directly increases the model capacity and therefore dramatically improves the classification accuracy. Specifically, our modified neural network starts with a convolution layer using 5×5 kernels with 32 channels in output followed by bias and ReLU. Then it has a 2×2 max pooling layer. After pooling, we design another 5×5 convolution layer with 64 output channels followed by bias and ReLU. It is followed by another 2×2 max pooling layer. After this, we add two fully connected layer with 1024 and 64 units respectively, followed by ReLU. Finally, after a fully connected layer, we got the logits. The architecture is summarized in Figure 7. This neural network follows the same loss function—softmax cross entropy—as the original LeNet and usting the same optimizer, gradient descent. The loss during training is visualized in Figure 8. The accuracy is summarized in Table 5. One can see that the modified LeNet achieves a lower training error and clearly a higher test accuracy against the original LeNet.

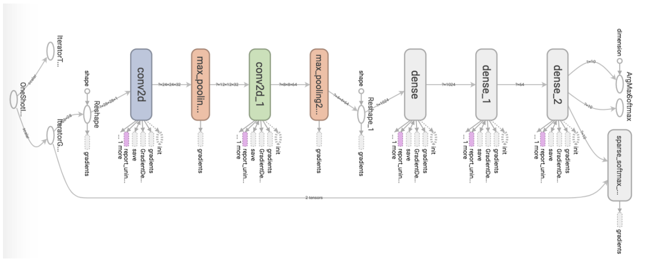

Figure 7: Architecture of modified LeNet.

LeNet-5	Modified LeNet
93.76%	95.03%

Table 5: The accuracy of neural network. Modified LeNet achieves the highest accuracy in our entire experiments.

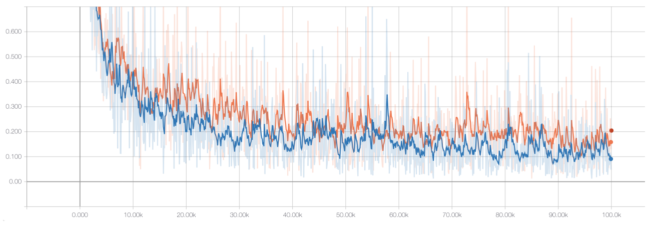

Figure 8: Loss during training. Blue line is smoothed loss for modified LeNet, while red line is for original LeNet.

9 CONCLUSION

In this project, we focused on the problem of recognizing noisy handwritten digits from gray-scale images. We used bilateral filters to remove noises and transformed images to binary ones before extracting features. We used several feature descriptors including PCA, HOG, and Harris corner features. As for classifiers, we exploited SVM, LDA, QDA, decision tree, random forest , and xgboost. Further, we proposed a novel neural network architecture by extending the model capacity of LeNet-5. Experiments demonstrated that the proposed neural network achieves the best performance—95% accuracy—on noisy MNIST dataset.

10 RESPONSIBILITIES

- Image pre-processing: Jinmei Zheng, Chunyi Wang
- Classifiers: Chunyi Wang
- Neural Network: Jinmei Zheng

REFERENCES

[1] Herman Blinchikoff and Helen Krause. 2001. *Filtering in the time and frequency domains*. The Institution of Engineering and Technology.

[2] Peter J Burt and Edward H Adelson. 1987. The Laplacian pyramid as a compact image code. In *Readings in Computer Vision*. Elsevier, 671–679.

[3] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1. IEEE, 886–893.

[4] Michael Elad. 2002. On the origin of the bilateral filter and ways to improve it. *IEEE Transactions on image processing* 11, 10 (2002), 1141–1151.

[5] Huang Fu Jie, Y-Lan Boureau, and Yann LeCun. 2007. Unsupervised learning of invariant feature hierarchies with applications to object recognition. *Computer Vision and Pattern Recognition* (2007).

[6] Chris Harris and Mike Stephens. 1988. A combined corner and edge detector.. In *Alvey vision conference*, Vol. 15. Citeseer, 10–5244.

[7] Daniel Keysers. 2007. Keysers, Daniel. "Comparison and combination of state-of-the-art techniques for handwritten character recognition: topping the mnist benchmark. (2007), arXiv:0710.2231.

[8] Yann LeCun. 1990. Handwritten zip code recognition with multilayer networks. *Proceedings. 10th International Conference on Pattern Recognition* 2 (1990), 35–40.

[9] Yann LeCun. 1995. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective* 261 (1995), 276.

[10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[11] Simard Patrice, Yann LeCun, and John Denker. 1993. Efficient pattern recognition using a new transformation distance. *Advances in neural information processing systems* (1993), 50–58.

[12] Luc Vincent. 1993. Morphological grayscale reconstruction in image analysis: applications and efficient algorithms. *IEEE transactions on image processing* 2, 2 (1993), 176–201.