Proceedings of the 9th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'23)
Brunel University, London, United Kingdom - August 03-05, 2023
Paper No. CIST 148
DOI: 10.11159/cist23.148

Adaptable and Efficient Digit Recognition System for Challenging Datasets: A Case Study on Pump Flowmeter Digits

Mahdis Salehpoor[1], Mohammad Elsayyed[1], Witold Kinsner[1], and Nariman Sepehri[2]
[1]Department of Electrical & Computer Engineering
[2] Department of Mechanical Engineering
University of Manitoba Winnipeg, MB, Canada
Salehpom@myumanitoba.ca; Elsayyem@myumanitoba.ca; Witold.Kinsner@umanitoba.ca;
Nariman.Sepehri@umanitoba.ca

Abstract – Machine digit recognition from various multi-digit displays is a complex task due to the sheer number of unique digit forms, each varying significantly in shape, size, and orientation. Traditional digit recognition libraries may not perform well for all cases, especially when dealing with digital screens that can be highly variable in terms of style, fonts, colour, contrast, intensity, pixel resolution, digit aspect ratio, and spacing. To address these challenges, we present a digit recognition algorithm that is designed to be fast, easy to use, and highly adaptable. Unlike a single fit-all solution, our system can be easily modified to fit different use cases and applications, incorporating additional layers of flexibility and adaptability. This is desirable since different types of displayed digits may have unique features or characteristics that traditional digit recognition libraries do not capture well. To further demonstrate the efficacy of the proposed system, we tested it on a unique pump-flowmeter digits format, which poses significant challenges for digit recognition algorithms due to the complicated shape and layout of the digits. This paper provides a detailed step-by-step account of our system's development and its performance on this challenging dataset. The presented system achieved an accuracy of 80% on test data, is simple and can be used by researchers, developers, and practitioners working in fields such as handwriting recognition, computer vision, machine learning, image processing, pattern recognition, and neural networks.

***Keywords*:** Digit recognition; adaptable system; pattern recognition; computer vision; machine learning; image processing

1. Introduction

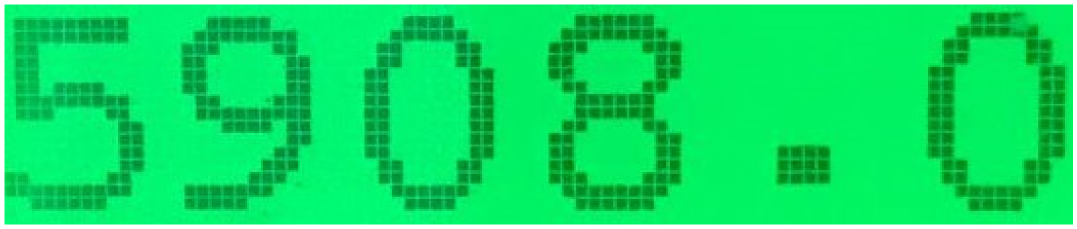
Fig. 1. Flowmeter digits.

The field of digit recognition has seen significant advancements in recent years with the development of various libraries and tools, including the widely used OCR library in Python. However, these traditional digit recognition libraries may not always be efficient or effective in recognizing specific types of digits. Fig. 1, shows an example of these unique digits which were tested in this project. To address this challenge, we present a system that combines transfer learning and image processing techniques to improve digit recognition accuracy. Unlike other complex algorithms and systems, our approach utilizes simple concepts and Python image processing tools to create a system that is easy to use and highly adaptable.

This paper provides a comprehensive account of the training and testing processes of our system, including an analysis of how pre-processing of images affects overall system performance. In addition, we explore various approaches to training the data, including training with a threshold filter and multiple filters such as blurring and eroding. We also examine the effects of testing with and without pre-processing and discuss how the system can be modified to recognize other types of digits and text with minor modifications to the image pre-processing stage. Our goal is to provide a system that is highly accurate, efficient, and accessible to researchers, developers, and practitioners in fields such as machine learning, computer vision, and pattern recognition. Through our detailed account of the development and testing of this new system, we aim to contribute to the ongoing efforts to improve and advance digit recognition technology.

In addition to our work on improving digit recognition accuracy through transfer learning and image processing techniques, we also ventured into real-time image classification using an external camera reading system. Specifically, we tested our system on images of a water pump flowmeter, where we aimed to detect any digits in the image. This application of our system could prove particularly useful in industrial settings where there is a need for efficient and accurate image classification. Currently, our algorithm takes approximately 30 seconds to classify an image, demonstrating the potential for further optimization and improvement. This paper presents the development and testing of this real-time image classification system, as well as its potential applications in the field of industrial digit recognition. It was intended for industrial applications.

2. Selection of Deep Learning Model

In this section, we delve into the intricate workings of transfer learning, where we leverage the power of a pre-trained model (VGG16) to train a customized model with the capability to accurately classify the distinct digits on the pump's flowmeter.

2.1. Motivation for Transfer Learning

Transfer learning is a machine learning technique that involves adapting a pre-trained model to a new task. Without transfer learning, training a new model for a new task from scratch can be a very time-consuming and computationally expensive process, especially if the dataset is small and specific because it lacks the required diversity, variability, and noise distortion. Therefore, using transfer learning can enhance the efficiency and effectiveness of a model for a new task significantly [1].

In the context of digit recognition, pre-trained models exist that have been trained on large datasets of digits, such as the Modified National Institute of Standards and Technology (MNIST) dataset with 60,000 training images and 10,000 testing images. However, these pre-trained models may not be as effective at recognizing specialized digits since they have not been trained specifically on these types of digits. This is where transfer learning becomes useful.

2.2. VGG16 Model

The VGG16 model is used in the designed system. The VGG16 model, developed by the Visual Geometry Group (VGG) at the Department of Engineering Science, University of Oxford, is a convolutional neural network (CNN) architecture composed of 16 layers, including 13 convolutional layers and three fully connected layers [2]. Trained on the ImageNet dataset, which comprises millions of images categorized into 1,000 different classes, VGG16 is a powerful model widely used for digit recognition and transfer learning tasks. Its extensive pre-training on a diverse image dataset makes it an excellent starting point for new tasks involving specialized digit recognition [2].

While the VGG16 is a good choice, it is important to consider alternative models for different scenarios. Two notable alternatives include the ResNet architecture and the Inception architecture (GoogLeNet). ResNet is known for its ability to train very deep networks using skip connections, and Inception introduces inception modules to capture features at different scales [3].

The VGG16 was chosen for our digit recognition and transfer learning because of its straightforward architecture and because it has already been trained on a large and diverse dataset like ImageNet. This pre-training makes it an advantageous starting point for digit recognition tasks.

It is worth noting that for other cases, such as limited computational resources or real-time applications, alternative models might be preferred. For instance, models like MobileNets [4] or SqueezeNet are designed for efficiency, making them more suitable for scenarios with limited computational resources. Similarly, MobileNet or EfficientNet are known for their faster inference times, making them better suited for real-time processing. Additionally, specialized tasks like object segmentation may benefit from models such as U-Net or Mask R-CNN.

3. Training VGG16 Model

In order to train our digit recognition system, we utilized a dataset consisting of 11 classes: digits 0 through 9, as well well as the decimal point represented by the period character. Each class was represented by 30 images, resulting in a total total of 330 images used for training. Fig. 3 provides a clear visual representation of the training data for each of the classes. classes. It is worth noting that the digits in our training set are unique and differ from other types of digital digits, such as as those found on LCD screens or 7-segment displays.

The training process was a critical step in the development of our system, as it involved training an existing model on new data to improve recognition accuracy. The training involved the use of various image processing techniques, such as threshold filters and morphological operations, to enhance the quality of the images and make them more suitable for training. This section will delve into the specifics of the training process and provide an analysis of how pre-processing the data affected the overall performance of the system.

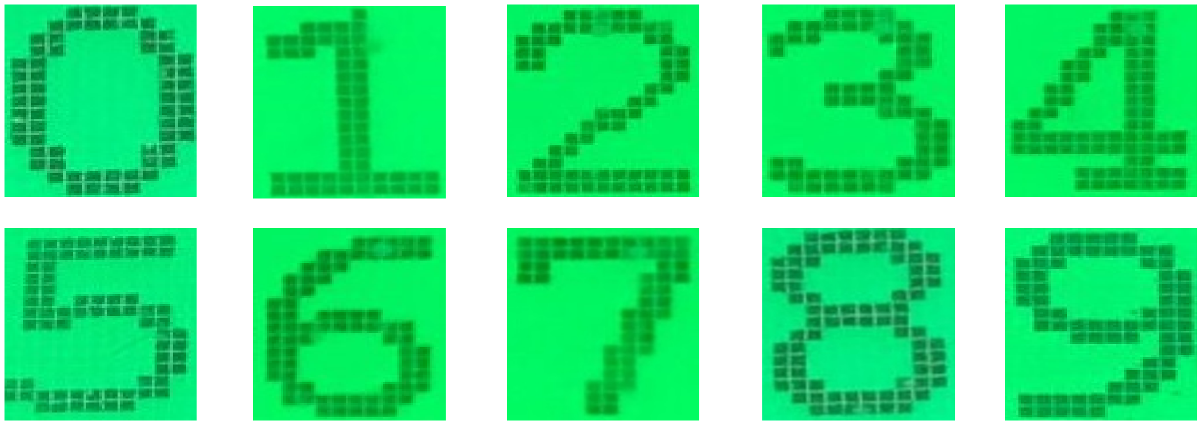

Fig. 3. Images used for training.

3.1. Adding New Training Layers

Initially, the system was trained on raw images with only minimal pre-processing steps, such as converting the image to grayscale. However, this approach resulted in a low-accuracy model, with only 5% accuracy on 50 test images. To address this issue, more pre-processing steps were added, including rescaling, rotation, shifting, and zooming to generate more variations of the images. After training the model for 10 epochs using the pre-processed data with an Adam optimizer, categorical cross-entropy loss function, and accuracy metric, the trained model was tested again. However, despite these improvements, as shown in Fig. 4 the model still struggled with the noisy and misleading information contained in the images. As a result, the accuracy of the model remained low.

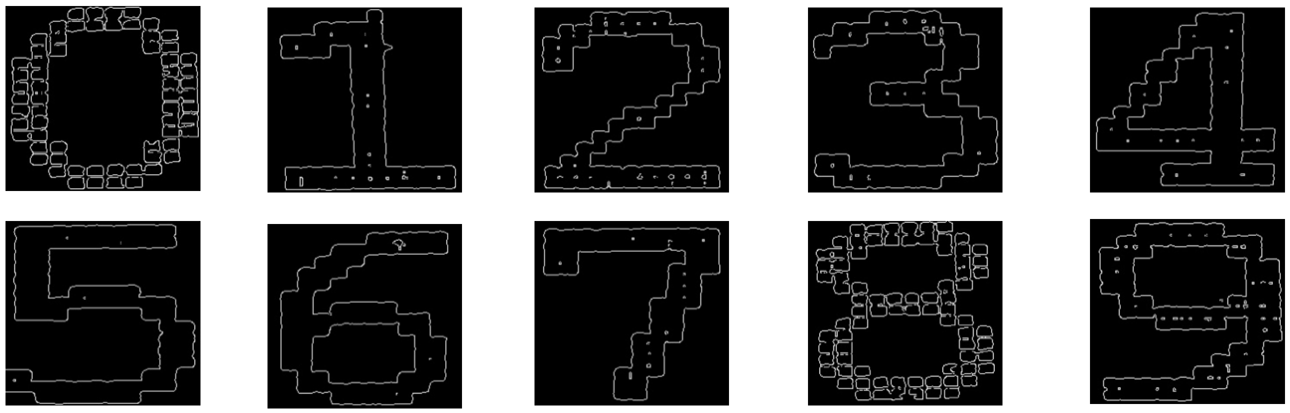
Fig. 4. Preprocessed training images without erosion and dilosion.

To further improve the system, a pre-processing function was implemented and went through several steps. the image was converted to grayscale to simplify the information in the image. Secondly, a threshold filter was applied the grayscale image. Thresholding is a process of converting grayscale images to binary images, where pixels above threshold are set to 1, and those below are set to 0.

Figure 5 shows the effect of each of the applied filters on the image. The pre-processing steps increased the accuracy of the model to 60% for 50 test images. However, we noticed that the thresholding filter did not work the same on all images, depending on the quality and the brightness of the images, the output of the thresholding was different.

To address this issue, we developed different pre-processing functions for different types of images. The threshold for blurry images was set differently than the threshold for better-quality images. This resulted in more fine-tuned training and better use of training data. In the end, we achieved a model with a higher accuracy of 75%.

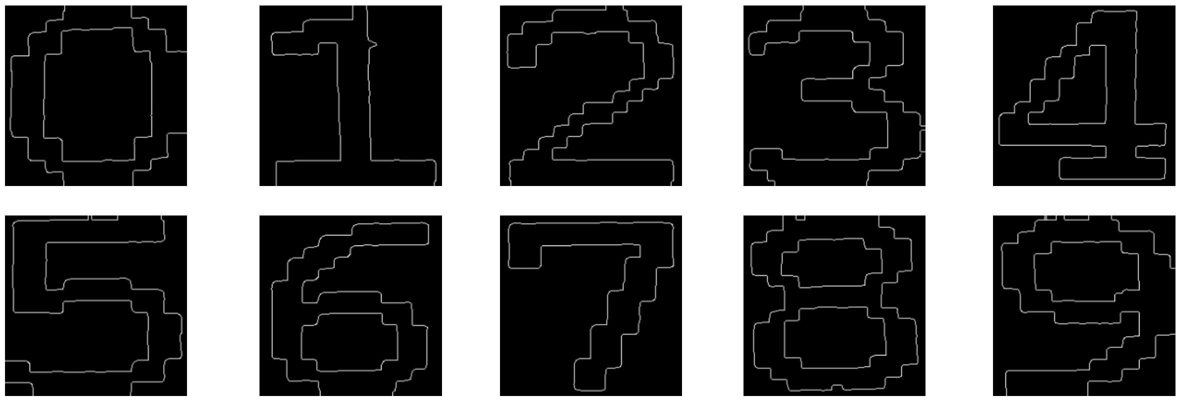
Fig. 5. Training images after eroding and dilating.

4. Testing the Model

After training the model on pre-processed data, the next step was to test its accuracy on unseen test data. The test dataset consisted of 50 images, each containing 3 to 5 digits. However, before testing the model on these images, it was necessary to pre-process them. This involved a series of steps to prepare the image for classification.

The first step in the pre-processing pipeline was to apply a threshold filter to the image. This was done to convert the image from a grayscale format to a binary image, where pixels above a certain threshold are set to 1 and those below are set to 0. This helped to isolate the digits from the background and improve the accuracy of the edge detection algorithm that would be applied later. Figure 6 shows an example of the threshold filter applied to an image.

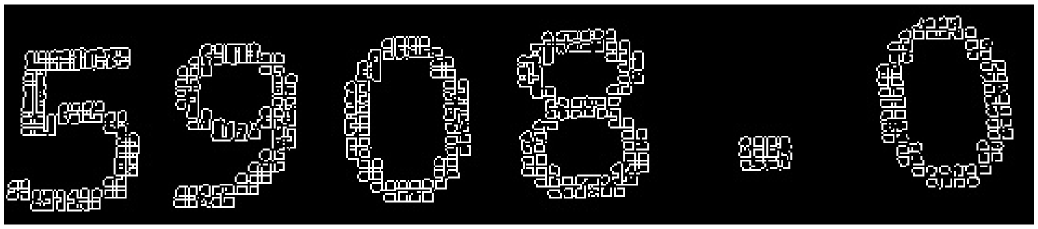

Fig. 6. Image after applying threshold.

Secondly, the image was dilated. As shown in Fig. 7, this step enlarges the white regions of the image and fills in small gaps between the objects.

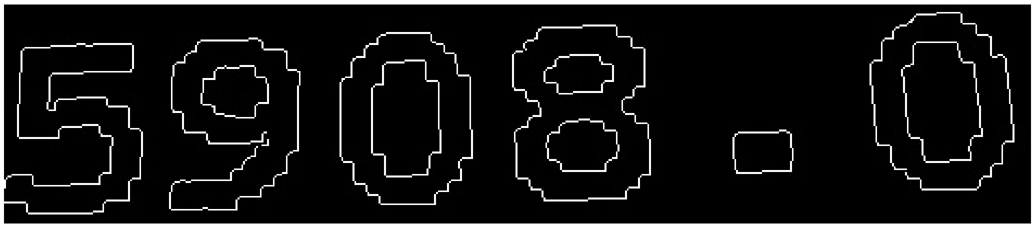

Fig. 7. Image after dilation.

Thirdly, the code applied an erode filter on the image. As shown in Fig. 8, this step shrinks the white regions of the image and removes small protrusions from the objects.

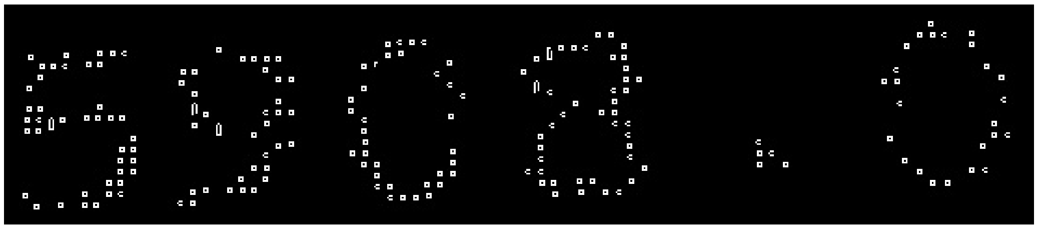

Fig. 8. Image after erosion.

The next step in the pre-processing pipeline was to perform both erosion and dilation operations on the binary image. These operations were useful in removing not only any noise in the image, but also in filling gaps between objects, making it easier to detect the edges of the digits. Figure 9, shows the resulting image.

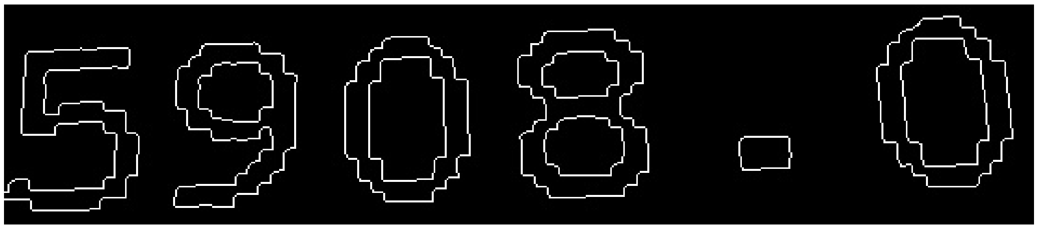

Fig. 9. Image after both dilation and erosion.

Once the image was pre-processed, the next step was to develop an edge detection algorithm to identify the location of the digits in the image. This algorithm was able to identify the edges of each digit and draw a bounding box around it. However, in some cases, there were multiple bounding boxes in the same area, which could lead to misclassification. To avoid this, as shown in Fig. 10, we only considered the biggest bounding box and the classified each digit and the dot character in that area.

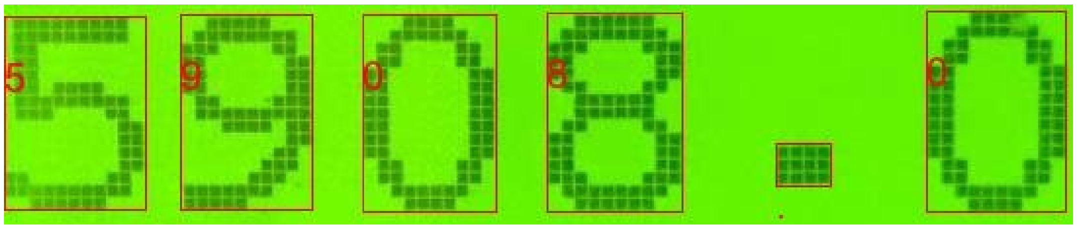

Fig. 10. Classified digits.

After identifying the location of each digit in the image, we passed each bounding box to the image classification algorithm to determine the closest digit. This process involved feeding the cropped image of each digit into the model and using the output to assign a label. Once all the digits in the image had been classified, we were able to determine the final label for the image.

The entire process of pre-processing and classifying each image was automated using Python scripts. This allowed us to test the model on the 50 test images quickly and efficiently. The results of the testing showed that the model was able to classify accurately the digits in the images with an accuracy of 80%. This performance is a significant improvement over the initial 5% accuracy achieved before the pre-processing steps were added and demonstrates the effectiveness of the pre-processing pipeline in improving the accuracy of the model.

5. Recommendations for Further Work

In order to improve the accuracy and generalization ability of the dynamic digit recognition model, there are several recommendations that can be considered in the future. First, collecting more training digits in various forms and styles can help prevent overfitting and improve the overall accuracy of the model. Additionally, data augmentation techniques such as random cropping, flipping, and rotating the images can be used to increase the amount of training data and improve the robustness of the model.

Another recommendation is to use a better pre-trained model to improve the accuracy of the mode. For example, extensions of the VGG16 model could also be considered. They include the Residual Neural Network (ResNet) [3], the Generative Pre-trained Transformer (GPT), ChatGPT, AlphaGo Zero, AlphaStar, and AlfaFold. Variants of the ResNet could also be employed, including MobileNets [4], ResNeXt [5], DenseNet [6], and WideDenseNet [7]. Fine-tuning the pre-trained model on the specific task of digit recognition is also important to improve its accuracy. Regularization techniques such as L1 and L2, dropout, and early stopping can help prevent overfitting and improve the generalization ability of the model.

Finally, exploring other image pre-processing techniques such as histogram equalization, adaptive thresholding, and denoising, can potentially improve the accuracy of the model. Overall, by implementing these recommendations, the dynamic digit recognition model can be further improved and serve as a strong basis for other image recognition algorithms, such as text recognition.

6. Conclusion

The system developed in this study utilized transfer learning and image pre-processing techniques to create a dynamic digit recognition model. The model was able to classify accurately digits in images and demonstrated the effectiveness of transfer learning in improving model accuracy. The use of pre-trained models and fine-tuning allowed for efficient training and saved computational resources. Additionally, the implementation of various image pre-processing techniques improved the model's robustness and accuracy. The simplicity and dynamic nature of the model make it a strong candidate for future image recognition algorithms, such as text recognition. However, to further improve the model's accuracy, future recommendations include collecting more training data, exploring additional image pre-processing techniques, and regularizing the model to prevent overfitting. Overall, the developed system showcases the potential of transfer learning and image pre-processing in creating effective and efficient image recognition models.

References

[1] Asmaul Hosna, Ethel Merry, Jigmey Gyalmo, Zulfikar Alom, Zeyar Aung, and Mohammad Abdul Azim, "Transfer learning: a friendly introduction," *Journal of Big Data*, vol. 9, ar. 102, Oct. 2022, 19 pages. [Online]. https://doi.org/10.1186/s40537-022-00652-w, https://journalofbigdata.springeropen.com/articles/10.1186/s40537-022-00652-w

[2] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Vision and Pattern Recognition*, arXiv:1409.1556 [cs.CV], Sep. 2014, last revised Apr. 2015, 14 pages. [Online]. https://arxiv.org/abs/1409.1556v6

[3] [HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition" in *2016 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR): pp. 770–778; Dec 10, 2015, 12 pages. https://arxiv.org/abs/1512.03385; https://arxiv.org/pdf/1512.03385.pdf, https://doi.org/10.48550/arXiv.1512.03385

[4] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," Apr 2017, 9 pages. https://arxiv.org/abs/1704.04861; https://arxiv.org/pdf/1704.04861.pdf

[5] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated Residual Transformations for Deep Neural Networks," arXiv, Apr 2017. https://arxiv.org/abs/1611.05431, https://arxiv.org/pdf/1611.05431.pdf

[6] Gao Huan, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger, "Densely connected convolutional networks," arXiv, Jan 2018. https://arxiv.org/pdf/1608.06993; https://arxiv.org/pdf/1608.06993.pdf

[7] Sergey Zagoruyko and Nikos Komodakis, "Wide residual networks," arXiv, Jun 2017. https://arxiv.org/abs/1605.07146 https://arxiv.org/pdf/1605.07146.pdf