# AUTONOMOUS CHOICE OF DEEP NEURAL NETWORK PARAMETERS BY A MODIFIED GENERATIVE ADVERSARIAL NETWORK

*Yantao Lu and Senem Velipasalar*
Department of Electrical Engineering and Computer Science
Syracuse University, Syracuse, NY, USA 13244-1240

## ABSTRACT

The choice of parameters, and the design of the network architecture are important factors affecting the performance of deep neural networks. However, this task still heavily depends on trial and error, and empirical results. Considering that there are many design and parameter choices, it is very hard to cover every configuration, and find the optimal structure. In this paper, we propose a novel method that autonomously and simultaneously optimizes multiple parameters of any given deep neural network by using a modified generative adversarial network (GAN). In our approach, two different models compete and improve each other progressively. Without loss of generality, the proposed method has been tested with three different neural network architectures, and three very different datasets and applications. The results show that the presented approach can simultaneously and successfully optimize multiple neural network parameters, and achieve increased accuracy in all three scenarios.

***Index Terms—*** Deep learning, neural networks, parameter choice, generative adversarial networks

## 1. INTRODUCTION

Training of deep learning methods requires large amounts of data, and they usually perform better when training data size is increased. Yet, for some applications, it is not always possible to obtain more data when the existing dataset is not large enough. Even though the raw data can be collected easily, data labeling or annotation is difficult, expensive and time consuming. Successors of [1] yielded better accuracies with less number of parameters on the same benchmark with some architectural modifications using the same building blocks. This shows that the choice of parameters, and the design of the architecture are important factors affecting the performance. Many researchers proposed different CNN architectures [2, 3, 4, 5, 6, 7, 8] to achieve higher accuracy.

However, building the neural network structure still heavily depends on trial and error, and empirical results. Considering that there are many design and parameter choices, it is not possible to cover every possibility, and it is very hard to find the optimal structure. Moreover, the hyper-parameters in the training phase also play an important role on how well the model will perform. Likewise, these parameters are also tuned manually in an empirical way most of the time.

Most existing work on optimizing network architectures is based on the genetic algorithms (GA) or evolutionary algorithms [9, 10, 11, 12, 13]. Many of these works focus only on a small subset of many design choices. More importantly, GA-based optimization uses a given set of blueprints and models, and performs a search over a limited set of candidates. GAs can search better solutions from limited possibilities, such as type of layers and activation functions. However, they cannot search for a solution that is not defined before. In addition, the complexity of GAs increases significantly when the number of choices increases to large scale. Rylander [14] has shown that the generations needed for convergence increases exponentially with the node size.

Generative Adversarial Networks (GANs) [15] are an important milestone in deep learning research, and have been adapted in many applications [16, 17, 18, 19]. In a GAN, a generator and discriminator are trained together until discriminator cannot distinguish the generated instances from the instances in the source domain. In this paper, we propose a novel and systematic way, which adopts a 'modified' GAN to find the optimal network structure and parameters for any given neural network model. In our approach, two different models compete and improve each other progressively with a GAN-based strategy. For this work, we have tested the performance of our approach on three different base neural network structures covering Long Short Term Memory (LSTM) networks and 3D CNNs, and different applications. Without loss of generality, we have chosen simpler network structures (not necessarily very deep ones) to optimize in order to show that the performance improvement is obtained not because of the increasing number of layers, but instead thanks to better refinement and optimization of the network parameters.

## 2. PROPOSED METHOD

The proposed architecture using a modified GAN-based network is shown in Fig. 1. In contrast to a traditional GAN, it is composed of two generators ($G_1$ and $G_2$), two evaluators ($E_1$ and $E_2$), and one discriminator ($D$).

The information, data, or work presented herein was funded in part by National Science Foundation under Grants 1739748 and 1816732, and by the Advanced Research Projects Agency-Energy, U.S. Department of Energy, under Award Number DE-AR0000940. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

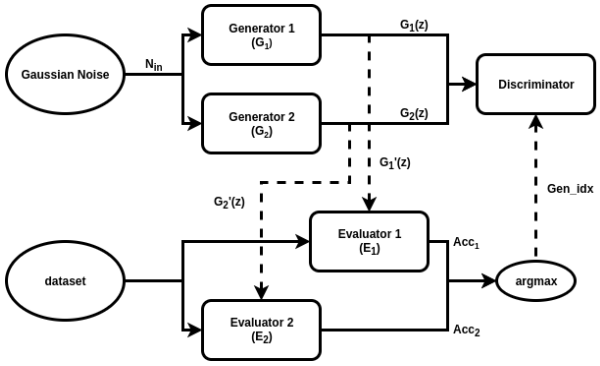


**Fig. 1**. Proposed modified GAN with 2 generators, and 2 evaluators.

### 2.1. Generative part

The two generators $G_1$ and $G_2$ have the same neural network structure shown in Fig. 2. Their input is a Gaussian noise vector $z \sim p_{noise}(z)$. As seen in Fig. 2, generators are composed of fully connected layers with leaky relu activations. At the output layer, $tanh$ is employed so that $G_i^j(z) \in (-1, 1)$, where $j \in \{1, 2, ..., length(G_i(z))\}$ and $i \in \{1, 2\}$. Then, the range of $G_i(z)$ is changed from $(-1, 1)$ to $(pm_{min}^j, pm_{max}^j)$ by using

$$G_i(z)' = [G_i(z) \times \frac{pm_{max} - pm_{min}}{2} + \frac{pm_{max} + pm_{min}}{2}]. \quad (1)$$

In (1), $pm_{max}$ and $pm_{min}$ are preset maxima and minima values, which are defined empirically based on values that a certain parameter can take, so that the value of the refined parameters can only change between $pm_{max}$ and $pm_{min}$. The re-scaled values $G_1(z)'$ and $G_2(z)'$ are then used as parameters of the evaluator networks. The length of $G_i(z)$ is determined by the number of network parameters that are refined, and is set at the generator network's last fully connected layer.

Generators are trained/improved by the discriminator, which is a binary classifier used to differentiate the results from generator outputs $G_1(z)$ and $G_2(z)$. Labels "H" and "L" (where $H \in \{1, 2\}, L =!H$) represent the generators with higher and lower accuracy results, respectively. The generator, which has the worse performance and is labeled by "L", is trained by the stochastic gradient descent (SGD) from the discriminator to minimize $log(1 - D(G_L(z))$ by using

$$\bigtriangledown_{G_L} \frac{1}{m} \sum_{j=1}^{m} log(1 - D(G_L(z^{(j)}))), \quad (2)$$

where $m$ is the number of epochs.

When $G'_H(z)$ becomes equal to $G'_L(z)$ for two consecutive iterations, the weights of $G_L$ will be re-initialized to default random values. The purpose of this step is to prevent the optimization stopping at a local maxima and also prevent the vanishing $tanh$ gradient problem.

### 2.2. Evaluation part

The evaluator networks have the same structure as the neural network whose parameters are being optimized or refined.


**Fig. 2**. Generator network

Thus, one of the strengths of the proposed approach is that it can be used to refine/optimize parameters of any deep neural network structure. As will be shown in Sec. 3, we have tested the proposed approach with three different network structures, and different sets of parameters.

Evaluator networks are built by using the parameters $G_1(z)'$ and $G_2(z)'$ provided by the generators. The training data $x \sim p_{data}(x)$ is used to evaluate these network models. We employ an early stopping criteria. More specifically, if no improvement is observed in $c$ epoches, the training is stopped. We then obtain the accuracies $acc_i = \mathbb{E}_{x \sim p_{data}(x)} E_i(x)$, $i = \{1, 2\}$. Let $H$ be the value of $i$ resulting in higher accuracy, and $L =!H$. Then "H" is used as the ground truth label for the discriminator, which marks the generator with better parameters, and trains the worse generator $G_L$.

### 2.3. Discriminator

We define the discriminator $D$ as a network (Fig. 3), whose output is a scalar softmax output, which is used for binary classification between better and worse generator. $G_1(z)$ and $G_2(z)$ are fed into the discriminator $D$, and the ground truth label indicating which is the better generator comes from the evaluators. Let $D(G(z))$ represent the probability that $G(z)$ came from the more accurate generator $G_H$ rather than $G_L$. We train $D$ to maximize the probability of assigning the correct label to the outputs $G_1(z)$ and $G_2(z)$ of both generators. Moreover, we simultaneously train the worse generator $G_L$ to minimize $log(1 - D(G_L(z))$. The discriminator $D$ provides the gradients to train the worse performing generator. The whole process can be expressed by:

$$min_{G_H} max_D \mathbb{E}_{z \sim p_z(z)} (log(D(G_H(z))) + log(1 - D(G_L(z)))),$$

where, $H = argmax_{i=\{1,2\}} (\mathbb{E}_{x \sim p_{data}(x)} E_i(x))$, $L =!H$.

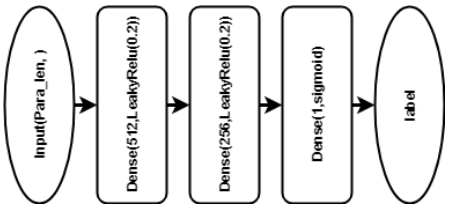


**Fig. 3**. Discriminator network

## 3. EXPERIMENTAL RESULTS

To show the promise of the proposed approach, we tested its performance, without loss of generality, with three different neural network structures, and different training data

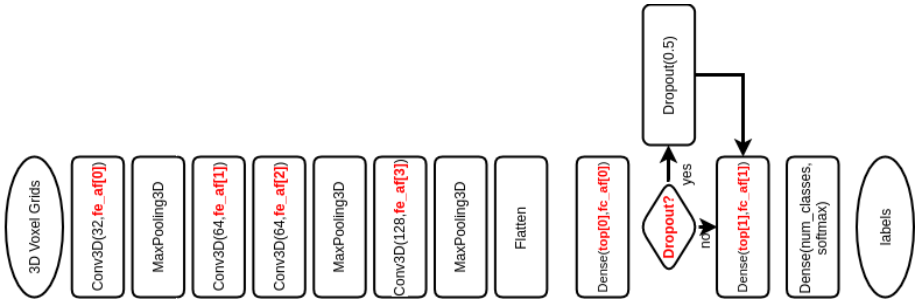


**Fig. 4**. Evaluator network for shape classification on ModelNet

types and applications. The network architectures, whose parameters are optimized, are shown in Figures 4, 5 and 6, wherein the parameters that are being refined/optimized are highlighted in red.

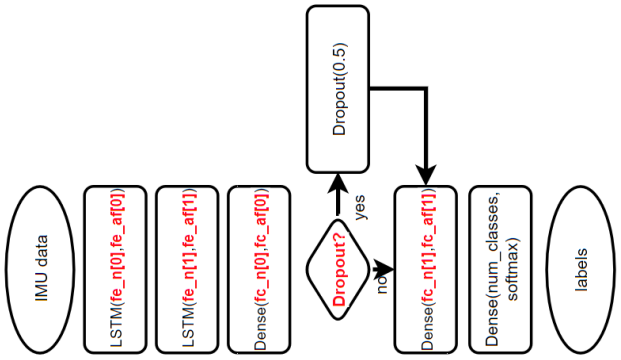


**Fig. 5**. Evaluator network for activity classification

### 3.1. Experiments with ModelNet

We applied the proposed approach on a 3D convolutional network by using the ModelNet40 dataset [20]. ModelNet is a dataset of 3D point clouds. The goal is to perform shape classification over 40 shape classes. Some example voxelized objects from the ModelNet40 dataset are shown in Fig. 7.

The 3D CNN model, shown in Fig. 4, is used for evaluators. Each generator's output is a 9-dimensional vector composed of different parameter settings. Two of the parameters are the number of neurons for two fully connected layers. Six of the parameters indicate the choice of activation function for fully connected and convolutional layers from ('Sigmoid', 'Relu', 'Linear', 'Tanh') functions. One of the nine parameters is a flag indicating whether to add a dropout layer between fully connected layers. In this case, $pm_{max}$ and $pm_{min}$ are set to be: [4000, 4000, 4, 4, 4, 4, 4, 4, 1] and [1, 1, 0, 0, 0, 0, 0, 0, 0], respectively. Selecting the number of neurons is a regression problem and choosing the activation function is a classification problem. In other words, for choosing the activation function, the $tanh$ output is put into bins, and the corresponding function is selected. It should be emphasized that the number of neurons is in a large range, which would be very difficult to handle with only a GA.

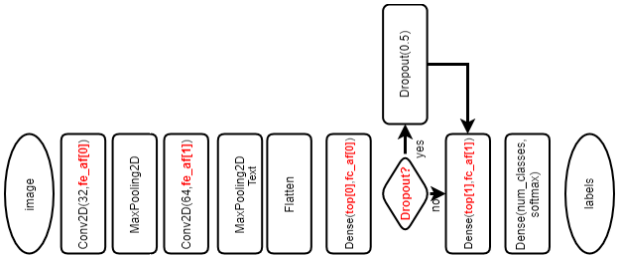


**Fig. 6**. Evaluator network for character recognition

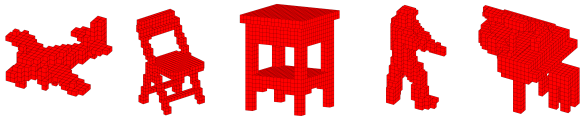

**Fig. 7**. Sample voxelized objects from ModelNet40 dataset.

The accuracy over number of epochs is shown in Fig. 8. The blue and red lines show the accuracies for Generator 1 and Generator 2, respectively. Green line is the saved model with the refined parameters providing the best accuracy. The accuracies of the original network (start accuracy) and the proposed approach (end accuracy) are presented in Table 1 for different early stopping criteria, more specifically, when $c$=1 and $c$=5. As can be seen, the proposed approach provides an increase in accuracy by autonomously and simultaneously refining nine parameters of the network in a systematic way.

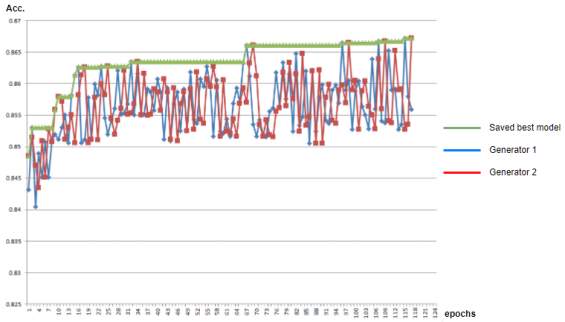


**Fig. 8**. Accuracy over number of epochs for the two generators on the ModelNet dataset.

| Number of epochs (c) for early stopping | 1 | 5 |
|---|---|---|
| Start accuracy | 83.43% | 84.31% |
| **End accuracy** | **85.71%** | **86.72%** |

**Table 1**. Accuracies of the original network (start accuracy) and the proposed approach (end accuracy) with different early stopping criteria (when c=1, and c=5).

### 3.2. Experiments with UCI HAR Dataset and an LSTM-based network

UCI HAR dataset [21] is composed of Inertial Measurement Unit (IMU) data captured during activities of standing, sitting, laying, walking, walking upstairs and walking downstairs. These activities were performed by 30 subjects, and the 3-axial linear acceleration and 3-axial angular velocity were collected at a constant rate of 50Hz.

In this case, the network model shown in Fig. 5 is used for evaluators. As can be seen, this network is an LSTM model. The output of each generator is a 9-dimensional vector which is composed of different parameter settings. More specifically, first four of the parameters are the number of neurons for two fully connected layers and two LSTM layers. Next four of the parameters indicate the choice of activation function for fully connected and two LSTM layers from ('Sigmoid', 'Relu', 'Linear', 'Tanh') functions. Last of the nine parameters is a flag indicating whether to add a dropout layer between the fully connected layers. In this case, $pm_{max}$ and $pm_{min}$ are set to be: $[4000, 4000, 2000, 2000, 1, 1, 1, 1, 1]$ and $[10, 10, 10, 10, 0, 0, 0, 0, 0]$, respectively.

The accuracy over number of epochs is shown in Fig. 9. The blue and red lines show the accuracies for Generator 1 and 2, respectively. Green line is the saved model with the refined parameters providing the best accuracy. The accuracies of the original network (baseline accuracy) and the proposed approach are presented in the second row of Table 2. The proposed approach provides an increase in accuracy for this LSTM network and this IMU dataset as well.

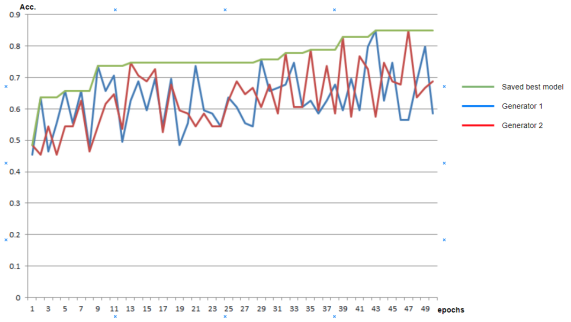


**Fig. 9**. Accuracy over number of epochs for the two generators on the UCI human activity recognition dataset.

| Dataset | Baseline Accuracy | Proposed Method |
|---|---|---|
| ModelNet | 84.17% | **86.72%** |
| UCI HAR | 81.17% | **84.85%** |
| Words built from Chars74k | 85.5% | **86.64%** |

**Table 2**. Accuracies of the original networks (baseline accuracy) and the proposed approach.

### 3.3. Experiments with Chars74k Dataset

We also tested our proposed approach with a word recognition method [22], which uses the characters from the Chars74k dataset [23] to build words. Chars74k dataset contains 64 classes (0-9, A-Z, a-z), 7705 characters obtained from natural images, 3410 hand-drawn characters using a tablet PC and 62992 synthesized characters from computer fonts giving a total of over 74K images. Some example words built from these characters are shown in Fig. 10.

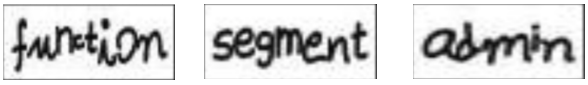


**Fig. 10**. Sample words built from the Chars74k dataset.

[22] uses the network model shown in Fig. 6 for character recognition, and then employs belief propagation for word recognition. We used the same network model in Fig. 6 for our evaluators, and performed the word recognition the same way to compare the word recognition accuracies. For the generators, the output is a 7-D vector composed of different parameter settings. More specifically, first two of the parameters are the number of neurons for two fully connected layers. Next four of the parameters indicate the choice of activation function for fully connected and convolutional layers from ('Sigmoid', 'Relu', 'Linear', 'Tanh') functions. The last of the seven parameters is a flag indicating whether to add a dropout layer between fully connected layers. In this case, $pm_{max}$ and $pm_{min}$ are set to be: $[4000, 4000, 4, 4, 4, 4, 1]$ and $[10, 10, 0, 0, 0, 0, 0]$, respectively.

The word recognition accuracies obtained by using the original network [22] (baseline accuracy) and the proposed approach are presented in the last row of Table 2. As can be seen, the proposed approach consistently provides an increase in accuracy for different types of networks and datasets.

In Table 3, we present the parameters used in the original networks, and the parameters that were refined and optimized by the proposed method for all three different scenarios.

| Dataset | Baseline Parameters | Parameters w/ Prop. Meth. |
|---|---|---|
| ModelNet | [1024,256,1,1,1,1,1,1,0] | [1242,1790,2,2,1,1,1,1,1] |
| UCI HAR | [512,2014,1024,256,1,1,1,1,0] | [771,939,2597,1403,1,1,1,1,0] |
| Words Chars74k | [1024,256,1,1,1,1,0] | [2804,2121,1,1,1,1,1] |

**Table 3**. Parameters used by the original networks and the parameters that were refined and chosen by the proposed approach.

## 4. CONCLUSION

We have presented a novel method to autonomously and simultaneously optimize multiple parameters of any given deep neural network. The set of parameters can include the number of neurons, the type of activation function, the choice of using drop out and so on. In our proposed approach, two different models compete and improve each other progressively. The use of the proposed modified GAN allows the choice of parameters from a very large range of values as opposed to using a small set. Without loss of generality, the proposed method has been tested with three different neural network architectures, and three different datasets. The results show that the presented approach can simultaneously and successfully optimize multiple neural network parameters, and achieve increased accuracy in all three scenarios.

## 5. REFERENCES


[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] Matthew D Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[4] Min Lin, Qiang Chen, and Shuicheng Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[5] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, et al., "Going deeper with convolutions," Cvpr, 2015.

[6] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[9] PG Benardos and G-C Vosniakos, "Optimizing feedforward artificial neural network architecture," *Engineering Applications of Artificial Intelligence*, vol. 20, no. 3, pp. 365–382, 2007.

[10] Badar Ul Islam, Zuhairi Baharudin, Muhammad Qamar Raza, and Perumal Nallagownden, "Optimization of neural network architecture using genetic algorithm for load forecasting," in *Intelligent and Advanced Systems (ICIAS), 2014 5th International Conference on*. IEEE, 2014, pp. 1–6.

[11] Kenneth O Stanley and Risto Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[12] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat, "Evolving deep neural networks," *arXiv preprint arXiv:1703.00548*, 2017.

[13] Marylyn D Ritchie, Bill C White, Joel S Parker, Lance W Hahn, and Jason H Moore, "Optimizationof neural network architecture using genetic programming improvesdetection and modeling of gene-gene interactions in studies of humandiseases," *BMC bioinformatics*, vol. 4, no. 1, pp. 28, 2003.

[14] Bart Ian Rylander, *Computational Complexity and the Genetic Algorithm*, Ph.D. thesis, Moscow, ID, USA, 2001, AAI3022336.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[16] Alec Radford, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *ICLR*, 2016.

[17] Justin Johnson, Alexandre Alahi, and Li Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*, 2016.

[18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint*, 2017.

[19] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.

[20] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[21] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones.," in *ESANN*, 2013.

[22] Yilan Li, Zhe Li, and Qinru Qiu, "Assisting fuzzy offline handwriting recognition using recurrent belief propagation," in *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*. IEEE, 2016, pp. 1–8.

[23] T. E. de Campos, B. R. Babu, and M. Varma, "Character recognition in natural images," in *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009.