DeepStealth: Leveraging Deep Learning Models for Stealth Assessment in Game-based Learning Environments

Wookhee Min, Megan H. Frankosky, Bradford W. Mott,
Jonathan P. Rowe, Eric Wiebe, Kristy Elizabeth Boyer, and James C. Lester

Center for Educational Informatics, North Carolina State University, Raleigh, NC 27695
{wmin, rmhardy, bwmott, jprowe, wiebe, keboyer, lester}@ncsu.edu

Abstract. A distinctive feature of intelligent game-based learning environments is their capacity for enabling stealth assessment. Stealth assessments gather information about student competencies in a manner that is invisible, and enable drawing valid inferences about student knowledge. We present a framework for stealth assessment that leverages *deep learning*, a family of machine learning methods that utilize deep artificial neural networks, to infer student competencies in a game-based learning environment for middle grade computational thinking, ENGAGE. Students' interaction data, collected during a classroom study with ENGAGE, as well as prior knowledge scores, are utilized to train deep networks for predicting students' post-test performance. Results indicate deep networks that are pre-trained using stacked denoising autoencoders achieve high predictive accuracy, significantly outperforming standard classification techniques such as support vector machines and naïve Bayes. The findings suggest that deep learning shows considerable promise for automatically inducing stealth assessment models for intelligent game-based learning environments.

Keywords: Game-Based Learning Environments, Stealth Assessment, Deep Learning, Computational Thinking, Educational Games.

1 Introduction

Recent years have witnessed growing interest in intelligent game-based learning environments, which simultaneously provide adaptive pedagogical functionalities delivered through intelligent tutoring systems and engaging learning experiences afforded by digital games [1–3]. A key benefit of game-based learning environments is their ability to embed problem-solving challenges within interactive virtual environments, which can enhance students' motivation [4] and facilitate learning through customized narratives and feedback [5, 6].

Stealth assessment is a pedagogical process, enabled by digital games, that involves real-time, invisible measurement of students' learning processes and outcomes. In game-based learning environments, stealth assessments have the potential to draw valid inferences about student competencies in an invisible and non-disruptive manner [6]. Stealth assessment is methodologically grounded in evidence-

centered design, and models typically consist of three components: a competency model, an evidence model, and a task model [6, 7]. A *competency model* represents students' knowledge and skills, which are modeled probabilistically. An *evidence model* identifies how observations of students' learning behaviors reveal student competencies on different skills and knowledge. A *task model* characterizes the challenges with which students interact, thereby producing evidence to infer the student's competency levels. Stealth assessments offer the potential to dynamically identify gaps in student knowledge, enabling personalized learning while simultaneously preventing students from "gaming the system" [5, 8]. With stealth assessments in place, it is possible for game-based learning environments to effectively diagnose student competencies and thus adaptively scaffold skill development, remediate misconceptions, and discourage behaviors that are not conducive to learning.

A key challenge posed by current stealth assessment techniques is manually devising models that enable valid inferences about student knowledge and skills. To address this challenge, we propose DeepStealth, a stealth assessment framework that leverages *deep learning* to automatically induce predictive models of student competency based on student interaction data. Deep learning is a family of machine learning methods that utilize deep artificial neural networks to model hierarchical representations of data for prediction tasks [9]. An empirical evaluation conducted with the ENGAGE game-based learning environment for middle grade computational thinking demonstrates that DeepStealth significantly outperforms support vector machines and naïve Bayes models at predicting students' competency levels (post-test performance on knowledge) under 10-fold cross validation. The results suggest that the DeepStealth approach holds significant promise for competency modeling in stealth assessment.

2 Related Work

Intelligent game-based learning environments seek to increase learners' motivation through rich settings, engaging characters, and compelling plots, and they foster learning through tailored scaffolding and context-sensitive feedback. Narrative-centered learning environments have been found to deliver experiences in which learning and engagement are synergistic [5]. Recent work in game-based learning has been undertaken for a broad range of subject matters ranging from high school mathematics [10] to language learning [3].

Intelligent game-based learning environments can support many forms of knowledge assessment. Shute proposed Bayesian network-based competency models, utilizing them in the context of stealth assessment [6]. Quellmalz and colleagues proposed an approach using simulation-based assessment, which was found to effectively assess science learning and inquiry practices [11]. Factor analysis techniques, such as performance factor analysis and matrix/tensor factorization, have been investigated for student performance prediction for knowledge assessment and problem-solving assessment [12, 13]. Bayesian knowledge tracing has been widely explored to assess latent knowledge and skills in the context of cognitive modeling

[14]. The approach presented here is the first to utilize deep learning techniques [9] to assess students' competency and performance levels within a technology-rich learning environment.

3 ENGAGE Game-Based Learning Environment

ENGAGE is an immersive game-based learning environment for middle school computer science education, built with the Unity game engine and FLARE user interface toolkit [15]. The curriculum underlying ENGAGE is based on the AP Computer Science Principles course [16] with learning objectives that are developmentally appropriate for U.S. middle school students (ages 11-13). The ENGAGE learning environment was designed to expose students to problems that encourage the development of computational thinking. Computational thinking is a problem-solving process that involves abstraction and algorithmic thinking, and leverages computational tools for data analysis, modeling, or simulations [17]. Additionally, the problem-solving activities within ENGAGE are designed to increase interest in computer science and provide a foundation for more advanced computer science work in high school.

In ENGAGE, students play the role of the protagonist who has been sent to a research facility to determine why all communication with the station has been lost. As students explore the research facility, they progress through each level, which consists of a series of interconnected rooms. Each room presents students with a set of computational challenges they solve by programming devices located in the room. Devices are programmed using a visual programming interface in which "blocks" that represent program elements are dragged and stacked together to create programs. Students interact with a cast of non-player characters who offer clues and relevant details via dialogue. The narrative is advanced through cinematics and learning is scaffolded by dialogue hints and animated vignettes.

The work presented in this paper focuses on students' problem-solving activities within ENGAGE's Digital World unit, which focuses on investigating how binary sequences are used to represent digital data. In one set of problem-solving activities, students must find the binary representation of a base-ten number to activate a lift device (Figure 1, left), which requires students to review an existing program for the lift device (Figure 1, right) to determine what base-ten number activates the lift, as

Figure 1. (Left) A lift device with an existing program, and (Right) the programming interface displaying the lift's program.

well as to understand the concept of bits in binary numbers and the weight assigned to each bit. Students read the program using the visual programming interface, flip binary tiles on the lift device (the white squares at the top of the lift device in Figure 1, left) to change the binary sequence until it matches the base-ten number specified in the program (Figure 1, right), and then stand on the lift device and execute its program, which allows them to jump to a previously inaccessible area of the room. Students are provided immediate feedback on the base-ten number interpretation of the binary sequence as they flip tiles on the lift device through a display above the binary sequence (e.g., 31 in Figure 1, left). In later parts of the level, the function that converts the binary sequence to a base-ten number is corrupted and students must correct the weights associated with each bit before executing the program.

During gameplay, students complete 16 problem-solving activities. Eleven activities involve matching binary sequences with base-ten numbers, and five activities involve correcting weights of bits. Four key features are logged from these problem-solving interactions. The features include the number of binary tile flips, the number of binary tile double flips (a binary tile flipped and then immediately flipped again), the number of times the device programs are executed, and the amount of time students spent in the programming interface. Since the tasks can be solved in a brute-force manner without understanding the binary representation of numbers or the programs that control the devices, it is important to be able to dynamically assess students' competency levels on binary representation.

In this work, we analyze the interaction data from 49 students (28 male, 21 female) from a teacher-led deployment of ENGAGE in two public middle school classrooms. Students interacted with ENGAGE over several weeks in their science class. Prior to beginning the unit, and immediately following the unit, students completed online pre- and post-test assessments measuring cognitive skills, computer science attitudes, and subject knowledge (e.g., binary representation). We investigated whether students achieved improvements in content knowledge from the Digital World, and a paired t-test comparing pre-test (M=0.42, SD=0.23) to post-test (M=0.63, SD=0.25) indicated that students' learning gains were statistically significant with a large effect size, $t(48) = 6.22$, $p < .001$, $d = .89$.

4 Feature Engineering: Problem-Solving Strategy Identification

In this section, we present a clustering analysis, which was conducted as a precursor to training deep learning models for stealth assessment. We anticipate that clustering on students' interaction data has the potential to identify distinct problem-solving strategies that students employed during ENGAGE's binary number learning activities. We hypothesize that clustering can serve as a form of automated feature engineering, producing strategy cluster features that enhance the predictive performance of the deep learning-based competency model. To perform the clustering analysis, we utilized expectation maximization (EM) [18] on the four features from the game interaction logs described in Section 3.

EM clustering was conducted on 49 students' interaction data with standardized features, where the number of clusters was automatically chosen to maximize the log-

likelihood of the data based on 10-fold cross validation. We found three student clusters that reflected distinct differences in students' problem-solving strategies. We also conducted an after-clustering analysis to examine normalized learning gains of each individual cluster [19]. All significance tests were performed using the Wilcoxon signed-rank test.

Cluster 1 containing 31 students (63%) included the most efficient problem solvers. These students comprised the highest performing group, with averaged normalized learning gains of 0.40. This group had significantly lower number of flips and double flips, and low program reading time compared to the other two clusters (all with $p<.01$). These results indicate that this group easily acquired concept knowledge, understood program meaning, and as a result were able to solve the binary challenges in the most principled way among the three groups.

Students in Cluster 2, 12% (6 students), and Cluster 3, 24% (12 students), differed along one principal dimension. Cluster 2 had a significantly higher number of double flips than Cluster 3 ($p=0.01$) while conducting a similar number of single flips ($p=0.84$). This difference could indicate that Cluster 2 may have reevaluated the problem by leveraging a guess-and-check strategy. Cluster 3 did not use a double flip strategy indicating that they may have been randomly flipping tiles to solve the problem. Overall, the number of program executions did not differ between clusters, indicating that each group only differed in the way that they prepared a solution to the problem, and not in the number of times they submitted an answer to the problem. With respect to averaged normalized learning gains, students in Cluster 2 and Cluster 3 achieved 0.26 and 0.20, respectively.

This clustering analysis suggests that each group exhibited distinctive patterns of interaction while solving the computational challenges. Because problem-solving strategies are related to students' learning processes, as evidenced by observed differences in normalized learning gains across groups, it appears that it may be possible to utilize students' problem-solving strategy clusters as inputs to deep learning-based competency models to improve their predictive performance. Next, we describe an automated stealth assessment framework that uses the clustering results as input, and we evaluate their benefit.

5 Stealth Assessment Leveraging Deep Learning

In this section, we first describe how our work is framed in evidence-centered design to support stealth assessment from the three model perspectives.

- *Competency Model:* We assess students' competency levels on knowledge about binary representation. We run cross validation tests to evaluate the model's performance based on actual labels from students' post-test performance.
- *Evidence Model:* Students' knowledge about binary representation can be revealed through their interactions during gameplay. These interactions are characterized by the four features described in Section 3. Also, the three problem-solving strategies discussed in Section 4 as well as the students' pre-test scores on the knowledge assessment, cognitive skill, and computer science attitude are considered as evidence.

- *Task Model:* We use 16 problem-solving tasks that have an objective of either "finding binary representations" or "correcting weights" in ENGAGE. Students' interactions on these tasks produce evidence that informs competency models to assess students' knowledge about binary representation.

Prior to training a predictive model for inferring students' competency, each student's data is labeled with their post-test performance, based on a tertile split ('A', 'B', or 'C') with respect to post-test scores in knowledge assessments, and thus the prediction is cast as a three-class classification that predicts one's performance.

Three input feature sets are considered to evaluate the impact of the engineered feature, strategy, as evidence: 1) the raw feature set (RF) based on the four features logged from interactions in the game, 2) the strategy feature set (SF) based on the strategy captured from the clustering, and 3) the combined feature set (CF) leveraging both RF and SF, and are independently analyzed to measure impacts of the features. These feature sets also include external learning measures, such as pre-test scores on knowledge, cognitive skills, and computer science attitudes, which are accessible and promising as predictors when reasoning competencies during the game-based learning activities.

The current work on competency modeling employs deep learning (DL), which learns hierarchical representations through multi-layer abstraction of data, often in the context of artificial neural networks [9]. One specific type of DL leverages *pre-training* that initializes weights in deep neural architectures [9, 20] with the objective of minimizing the reconstruction error of the original input. The pre-training process has been shown to help find a region of parameter space that can reach a better local optimum in a non-convex optimization graph, without which training deep neural networks often becomes hard to optimize due to underfitting and vanishing/exploding gradient problems [9]. After pre-training, models can be fine-tuned with standard optimization methods such as stochastic gradient descents to perform classification or regression tasks. DL based techniques have proven very successful in achieving state-of-the-art performance on a wide range of machine learning tasks [9].

We utilize DL with a pre-training technique, stacked denoising autoencoders (SDAEs) [20], for competency modeling. We describe the SDAE technique, as well as autoencoders (AEs), a basic form of SDAEs. An autoencoder (AE) is a pre-training technique that features (1) encoding (f) that deterministically maps (W_1) an input vector (x) into a hidden representation $f(x)$ using a non-linear transformation characterized by an activation function, s (Equation 1) and (2) decoding (g) that maps (W_2) the hidden representation $f(x)$ back to $g(f(x))$, a reconstructed vector of the input vector (x), using s (Equation 2). The objective in AEs is on learning representations (W_1 and W_2) along with two bias terms (b_1 and b_2) by minimizing the reconstruction error between x and $g(f(x))$ through backpropagation methods (e.g., stochastic gradient descent) using an unsupervised criterion.

$$f(x) = s(W_1 x + b_1). \quad (1)$$

$$g(f(x)) = s(W_2 f(x) + b_2). \quad (2)$$

Stacked denoising autoencoders (SDAEs) extend AEs from two perspectives. First, deep neural networks can be pre-trained by stacking layers of AEs (stacked autoencoders). Second, SDAEs hypothesize that effective representations should be

able to robustly recover corrupted inputs into the uncorrupted original inputs (denoising) [20]. The first objective can be achieved using layer-wise pre-training, which is a technique to pre-train the entire network using an iterative process, from the first to last layer. For example, once the first layer is pre-trained based on the reconstruction criterion, the output from the first layer is fed as the input to the next layer, and then the second layer can be pre-trained in the same manner. On the other hand, to achieve the latter objective, SDAE performs a corruption process by injecting noise (we set random neurons to 0) into the original input vector (x), as illustrated in Figure 2. In this method, the input vector x is partially corrupted into x' based on the corruption level that defines the probability of corrupting input neurons. Then, x' is deterministically mapped to $f(x')$ via an encoding process, and f(x') is recovered to $g(f(x'))$ via a decoding process following standard AEs. A key difference in SDAE is that the objective function is to minimize the reconstruction error (L) between the uncorrupted input x and the decoded output based on the corrupted input, $g(f(x'))$, interpreted as denoising corrupted inputs. As a result, SDAEs provide benefits over AEs by effectively dealing with noisy input data utilizing denoising techniques and preventing weights from reaching a trivial solution (identity matrix) that could cause overfitting in prediction tasks [20]. As noted above, once pre-training of the entire network is completed, all the initialized weights from pre-training can be fine-tuned using a supervised criterion.

As in other machine learning techniques, selecting hyperparameters for DL and a corruption level (fraction of corrupted input neurons) for SDAE often must be empirically determined. In this work, on the one hand, we explore the model space using a grid search of some parameters such as the corruption level (0.02, 0.1, or 0.5), the number of hidden layers (3–5), and the input feature set (Raw, Strategy, or Combined). On the other hand, we have fixed the following parameters: the number of neurons per hidden layer (50), the gradient descent optimization method (stochastic gradient descent), the learning rate (0.1), the activation function (sigmoid), the loss criterion (mean square error), and the number of epochs for gradient descent learning (100). Input to DL is encoded with 70 neurons (64 raw features, 6 pre-test scores), 9 neurons (3 strategy features, 6 pre-test scores), or 73 neurons (64 raw features, 3 strategy features, 6 pre-test scores) based on the feature set that the model utilizes, RF, SF, and CF, respectively, while every model has 3 output neurons ('A', 'B', and 'C') for the predicted post-test performance.

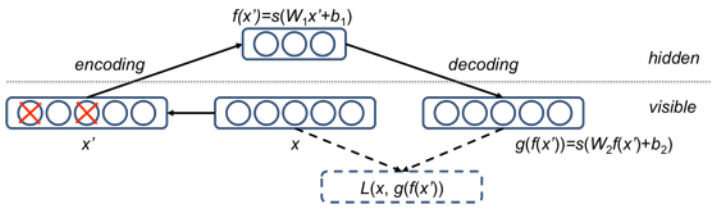

Figure 2. Illustration of stacked denoising autoencoders; red crosses denote corruption [20].

6 Empirical Evaluation

We evaluate how accurate our predictive competency models are using 10-fold student-level cross validation. Similar to the method applied to SDAEs, we run a grid search for choosing support vector machines' (SVMs) hyperparameters, and the model that achieves the highest accuracy rate is selected according to cross validation results. This work examines two hyperparameters that are popularly explored for optimization: the penalty parameter (C) and gamma (γ) for SVMs with a radial basis function [21]. C is chosen from {1, 10, 100}, and γ is chosen from {0.005, 0.01, 0.1, 0.5, 1.0}. For naïve Bayes (NB) models, we use non-parametric kernel density estimation, since our training data does not necessarily follow a normal distribution.

Table 1. Averaged accuracy rates of the highest performing NB, SVM, and DL.[1]

	NB	SVM	DL
Raw Feature Set (RF)	53.5%	56.5%	65.5%
Strategy Feature Set (SF)	51.0%	55.0%	55.0%
Combined Feature Set (CF)	53.5%	56.5%	**71.5%**

Model evaluation is performed along two dimensions. In the first evaluation, we train models based on the three machine learning techniques (NB, SVM, DL) along with all adjustable parameters and evaluate models' predictive performance using 10-fold cross validation. From the results, we choose the model that achieves the highest average accuracy rate per pair of machine learning approach and feature set (Table 1). In cross validations, all models use the same split of the training and validation set for pairwise comparisons. Overall, the highest performing DL model (the number of hidden layers: 3, corruption level: .02, combined feature set) achieves 71.5% accuracy rate, which significantly outperforms both the highest performing models from NB and SVM (C=1, γ=0.005) as well as the majority class baseline (36.7%).

For additional analyses, we run the Friedman test with a post hoc analysis with Wilcoxon signed-rank tests to compare high performing models. The Friedman test indicates there is a statistically significant difference in accuracy rates depending on the models, χ^2 (2)=6.93, p=.03. The Wilcoxon signed-rank post hoc tests indicate the DL model elicits statistically significant improvements in accuracy rates over the SVM model (Z=-2.06, p=.04) and the NB model (Z=-2.25, p=.02), but SVM vs. NB does not constitute a statistically significant difference.

In the second evaluation, by aggregating fold-based validation accuracies and conducting Wilcoxon tests, we measure the impact of each parameter used in DL, such as input feature set types, corruption levels, and the number of layers (Table 2). DL models that leverage CF utilizing both RF and SF obtain the highest average accuracy rate. The Wilcoxon signed-rank test indicates that there is not a statistically significant difference between CF-based models and RF-based models (Z=-1.69,

[1] With respect to DL's runtime performance, a prediction takes 0.3 milliseconds on average for 3 hidden layer models on the test machine with a 2.7 GHz Intel Core i7 CPU and 8 GB RAM.

p=.09). However, CF shows promise as a strong predictor set by achieving both the highest performance across models (Table 1) and the highest average performance across parameter settings (Table 2). This result demonstrates that the performance of DL can be further improved by taking advantage of human-engineered features, while SVMs and NBs seem to not easily benefit from the additional information in this evaluation.

Table 2. Parameter-wise SDAE model evaluation (left: feature set as the independent variable (IV), middle: corruption level as IV, right: number of layers as IV).

Feature Set	Accuracy Rate	Corruption Level	Accuracy Rate	Num. of Layers	Accuracy Rate
Raw (RF)	60.7%	0.02	57.8%	3	57.9%
Combined (CF)	**62.8%**	**0.1**	**58.2%**	4	56.4%
Strategy (SF)	49.5%	0.5	57.2%	**5**	**58.2%**

7 Conclusions and Future Work

This paper has introduced DeepStealth, a novel stealth assessment framework based on deep learning, which shows considerable promise for accurately assessing learners' competency levels. Using data from classroom studies with a game-based learning environment for middle grade computational thinking, we conducted an empirical evaluation that found that DeepStealth, which uses deep learning models with stacked denoising autoencoders, significantly outperforms baseline approaches, including naïve Bayes models and support vector machines, as well as the majority class baseline. Moreover, results suggest that the performance of deep learning in DeepStealth can be further improved when utilizing engineered features through deep learning's pre-training and fine-tuning process. In the future, it will be important to investigate how much the external learning measures (i.e., pre-test scores) contribute to the competency model's performance beyond game interaction logs, explore other deep learning techniques that can effectively deal with evidence with variant lengths, and examine parameter and hyperparameter optimization techniques for improved performance. Together, these techniques may be able to further improve deep learning-based approaches to stealth assessment.

Acknowledgments. This research was supported by the National Science Foundation under Grant CNS-1138497. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Lester, J., Ha, E., Lee, S., Mott, B., Rowe, J., Sabourin, J.: Serious games get smart: Intelligent game-based learning environments. *AI Magazine*, 34(4), 31–45 (2013)
2. Jackson, T., McNamara, D.: Motivation and Performance in a Game-based Intelligent

Tutoring System. *Journal of Educational Psychology*, 105(4), 1036–1049 (2013)
3. Johnson, L.: Serious use of a serious game for language learning. *International Journal of Artificial Intelligence in Education*, 20(2), 175–195 (2010)
4. Garris, R., Ahlers, R., Driskell, E.: Games, motivation, and learning: a research and practice model. *Simulation & Gaming*, 33(4), 441–467 (2002)
5. Rowe, J., Shores, L., Mott, B., Lester, J.: Integrating learning, problem solving, and engagement in narrative-centered learning environments. *International Journal of Artificial Intelligence in Education*, 21(2), 115–133 (2011)
6. Shute, V.: Stealth assessment in computer-based games to support learning. *Computer games and instruction*, 55(2), 503–524 (2011)
7. Mislevy, R., Steinberg, L., Almond, R.: On the structure of educational assessment. *Measurement: Interdisciplinary research and perspective*, 1(1), 3–62 (2003)
8. Baker, R., Corbett, A., Koedinger, K., Wagner, A.: Off-task behavior in the cognitive tutor classroom: when students game the system. In: *SIGCHI conference on Human factors in computing systems*, pp. 383-390 (2004)
9. Bengio, Y.: Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 1–127 (2009)
10. Kebritchi, M., Hirumi, A., Bai, H.: The effects of modern mathematics computer games on mathematics achievement and class motivation. *Computers & Education*, 55(2), 427–443 (2010)
11. Quellmalz, E., Timms, M., Silberglitt, M., Buckley, B.: Science assessments for all: Integrating science simulations into balanced state science assessment systems. *Journal of Research in Science Teaching*, 49(3), 363–393 (2012)
12. Sahebi, S., Huang, Y., Brusilovsky, P.: Predicting Student Performance in Solving Parameterized Exercises. In: *the 12th International Conference on Intelligent Tutoring Systems*, pp. 496–503 (2014)
13. Min, W., Rowe, J., Mott, B., Lester, J.: Personalizing Embedded Assessment Sequences in Narrative-Centered Learning Environments: A Collaborative Filtering Approach. In: *the 16th International Conference on Artificial Intelligence in Education*, pp. 369–378 (2013)
14. Corbett, A., Anderson, J.: Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4), 253–278 (1994)
15. Mott, B., Rowe, J., Min, W., Taylor, R., Lester, J.: FLARE: An Open Source Toolkit for Creating Expressive User Interfaces for Serious Games. In: *the 9th International Conference on the Foundations of Digital Games*, (2014)
16. AP® Computer Science Principles Draft Curriculum Framework: 2014. http://www.csprinciples.org/. Accessed: 2014-09-05
17. Barr, V., Stephenson, C.: Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM Inroads,* 2(1), 48–54 (2011)
18. Fraley, C., Raftery, A.: How many clusters? Which clustering method? Answers via model-based cluster analysis. *The computer journal*, 41(8), 578–588 (1998)
19. Marx, J., Cummings, K.: Normalized change. *American Journal of Physics*, 75(1), 87–91 (2007)
20. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11, 3371–3408 (2010)
21. Keerthi, S., Lin, C.: Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural computation*, 15(7), 1667–1689 (2003)