# Adversarial Generation of Real-time Feedback with Neural Networks for Simulation-based Training

**Xingjun Ma, Sudanthi Wijewickrema, Shuo Zhou, Yun Zhou,**
**Zakaria Mhammedi, Stephen O'Leary, James Bailey**
The University of Melbourne, Australia
{xingjunm@student, swijewickrem@, zhous@student, yun.zhou@,
zmhammedi@student., sjoleary@, baileyj@}unimelb.edu.au

## Abstract

Simulation-based training (SBT) is gaining popularity as a low-cost and convenient training technique in a vast range of applications. However, for a SBT platform to be fully utilized as an effective training tool, it is essential that feedback on performance is provided automatically in real-time during training. It is the aim of this paper to develop an efficient and effective feedback generation method for the provision of real-time feedback in SBT. Existing methods either have low effectiveness in improving novice skills or suffer from low efficiency, resulting in their inability to be used in real-time. In this paper, we propose a neural network based method to generate feedback using the adversarial technique. The proposed method utilizes a bounded adversarial update to minimize a $L1$ regularized loss via back-propagation. We empirically show that the proposed method can be used to generate simple, yet effective feedback. Also, it was observed to have high effectiveness and efficiency when compared to existing methods, thus making it a promising option for real-time feedback generation in SBT.

## 1 Introduction

Supporting the learning process through interactive feedback is important [Billings, 2012]. Appropriate and timely feedback intervention increases learning motivation, facilitates skill acquisition/retention, and reduces the uncertainty of how a student is performing [Davis, 2005]. With the development of virtual reality techniques, simulation-based training (SBT) has become an effective training platform in a range of applications including surgery [Wijewickrema *et al.*, 2017; Ma *et al.*, 2017], military training [Cosma and Stanic, 2011], and driver/pilot training [de Groot *et al.*, 2011]. However, it still requires the presence of human experts so that real-time feedback can be provided during training to ensure that relevant skills are learned. This has been one of the obstacles to the spread of SBT systems [Lateef and others, 2010]. As such, it is important to automate the generation of real-time feedback in SBT.

Feedback generation is a classical problem in artificial intelligence (AI) systems. Intelligent tutoring systems are one such class of AI systems that aims to provide immediate instruction or feedback to learners [Billings, 2012]. Another example is autonomous driving systems that take the surrounding environment as input and output feedback to the car to adjust the steering wheel [Chen *et al.*, 2015]. In reinforcement learning systems such as mobile robot navigation, the hardware or software agent learns its behaviour based on reward feedback from the environment [Sutton and Barto, 1998].

When compared to the above mentioned applications, SBT focuses more on educational gains such as the acquisition of proper skills [Steadman *et al.*, 2006; Lateef and others, 2010]. As such, SBT requires a higher degree of "hands-on" experiential interaction. Figure 1 shows an example of such a SBT system: The University of Melbourne Virtual Reality Temporal Bone Surgical Simulator [Wijewickrema *et al.*, 2016]. Rule-based feedback tutoring methods that work in domains such as algebra and physics are not flexible enough for SBT that requires a high level of user interaction and complex user behaviour. Autonomous driving systems and reinforcement learning systems mainly focus on the outcomes and mostly deal with cognitive tasks. Therefore, feedback generation methods in these systems are not directly transferable to SBT, especially for non-cognitive SBT scenarios. The aim of this paper is to develop an automated feedback generation method that can be used in SBT via supervised learning.

Feedback generation in SBT has three challenges. First, feedback should be generated in a timely manner as delayed feedback can lead to confusion or even cause fatal consequences in reality. An acceptable time-limit is 1 second after inappropriate action is detected. This is because feedback should be provided before the learner makes the next move [Rojas *et al.*, 2014]. Second, feedback should be actionable instructions that can be followed by the trainee to improve skills or correct mistakes. This is because SBT tasks often consist of a series of delicate operations that require precise instructions. Third, feedback should be simple, referring to only a few aspects of the skill, as practically people cannot focus on many things at a time. Also, this reduces distractions to the trainee and decreases cognitive load, thus increasing the usefulness of the feedback [Sweller, 1988].

In this paper, we make the following contributions:

- We demonstrate how the adversarial technique can be

Figure 1: *The University of Melbourne Virtual Reality Temporal Bone Surgery Simulator*: it consists of a computer that runs a 3D model of a human temporal (ear) bone and a haptic device that provides tactile resistance to simulate drilling.

used to generate actionable knowledge or feedback with neural networks.

- We propose a novel neural network based feedback generation method that works with a $L1$ regularized loss to control the simplicity of feedback and a bounded update to ensure the generated feedback has practical meaning.

- We show that the proposed method has high effectiveness as well as high efficiency when compared to existing methods, making it possible to be used for real-time feedback generation in SBT.

The structure of the paper is as follows. Section 2 introduces related work in this field. Section 3 illustrates the real-time feedback process and the formal definition of the problem. The proposed feedback generation method is described in Section 4 and evaluated along with existing methods in Section 5. Section 6 concludes the paper.

## 2 Related Work

The simplest way to provide feedback in SBT is the rule-based approach. The "follow-me" approach (ghost drill) [Rhienmora *et al.*, 2011] and the "step-by-step" approach [Wijewickrema *et al.*, 2016] in surgical simulation are examples of this approach. However, it may be hard for a novice who has limited experience to follow a ghost drill at his own pace, and step-by-step feedback will not respond if the trainee does not follow the suggested paths.

Other works utilize artificial intelligence techniques to generate feedback that can change adaptively in response to the novice's abilities and skills. One example is the use of Dynamic Time Warping (DTW) to classify a time series of surgical data and support feedback provision in lumbar disk herniation surgery [Forestier *et al.*, 2012]. However, this approach is less accurate at the beginning of a procedure when not much data is available. A supervised pattern mining algorithm was used in temporal bone surgery to identify significant behavioural patterns, classified as novice or expert, based on existing examples [Zhou *et al.*, 2013b]. Here, when

a novice pattern is detected during drilling, the closest expert pattern was delivered as feedback. However, it is very difficult to identify significant expert/novice patterns as novices and experts often share a large proportion of similar patterns.

A similar attempt used a prediction model to discriminate the expertise levels using random forests, and then generate feedback directly from the prediction model itself [Zhou *et al.*, 2013a]. Here, the generated feedback was the optimal change that would change a novice level to an expert, based on votes of the random forest (Split Voting (SV)). Decision trees and random forests were used in other research areas as well to provide feedback. For example, a decision tree based method was used in customer relationship management to change disloyal customers to loyal ones [Yang *et al.*, 2003; 2007]. Generating feedback from additive tree models such as random forest and gradient boosted trees is NP-hard, but the exact solution can be found by solving a transformed integer linear programming (ILP) problem [Cui *et al.*, 2015].

In this paper, we propose a neural network based method to generate feedback using the adversarial technique. One intriguing property of neural networks is that the input can be changed by maximizing the prediction error so that it moves into a different class with high confidence [Szegedy *et al.*, 2013]. This property has been used to generate *adversarial examples* from deep neural nets in image classification [Goodfellow *et al.*, 2014]. An adversarial example is formed by applying small perturbations (imperceptible to the human eye) to the original image, such that the neural network misclassifies it with high confidence. Although the adversarial example has similarities to the feedback problem in that they both change the input to a different class, they are not synonymous. First, the adversarial example is formed by adding intentionally-designed noise that may result in states that do not exist or have practical meaning in a real-world dataset such as that of the feedback problem. Second, only a few changes to inputs are recommended for feedback, to make it useful to follow. These considerations lead to the formal problem definition below.

## 3 Problem Definition

In this section, we discuss the real-time feedback process, and show how skill/behaviour level is defined in SBT applications. We then formally define the feedback generation problem as applied to SBT.

### 3.1 Feedback Process Overview

Figure 2 illustrates the real-time feedback process in SBT. It operates in two steps: 1) offline training and 2) real-time feedback provision. In the offline stage, a feedback generation method is trained via supervised learning on labelled (novice/expert) skill samples. In real-time, when a trainee is practising on the simulator, novice skill will be captured and input into the feedback generation method to obtain feedback about where improvement is required. Technically, feedback is the suggested action that can improve novice skill to expert skill. Finally, the feedback will be delivered immediately to the trainee to improve behaviour. The focus of this paper is the feedback generation method as highlighted in grey.
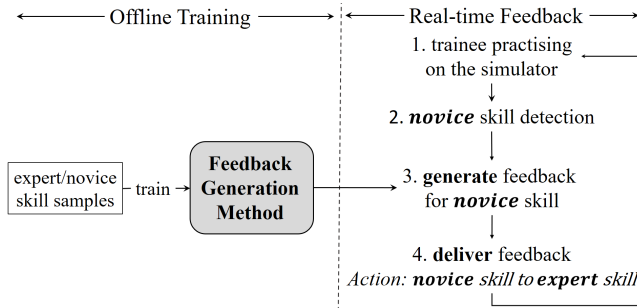
Figure 2: The real-time feedback process in SBT.

## 3.2 Definition of User Skill

SBT often works with multivariate time series data. This is because a SBT task often consists of a series of steps over a period of time. The skill level is usually defined over a period of time, based on the values of certain skill metrics.

In general, skill metrics are: 1) motion-based, 2) time-based, 3) position-based, or 4) system settings. Motion-based metrics are often signals captured from haptic devices or sensors, for example, the speed and engine rpm in a driving simulator. Time-based metrics measure quantities such as reaction time in military training. Position-based metrics relate to the location of the current procedure and include quantities such as position coordinates and distance to landmarks. System settings refer to measures that affect the environment such as the magnification level in surgical simulation.

**Definition 1.** *In simulation-based training, user skill is a feature vector summarizing user behaviour over an arbitrary period of time and annotated with class labels.*

For example, consider a SBT environment for surgery. Here, the level of skill can be defined by the quality of a stroke, a continuous motion of the drill with no abrupt changes in direction. The period of time here, over which the user behaviour is summarised, is the time interval of the complete stroke. Metrics that define the quality of a stroke within this time interval include measures such as stroke length, speed, acceleration, duration, straightness, and force [Zhou et al., 2015]. We denote such a skill metric as a *feature*. A vector of feature values that defines user skill is an *instance* and is associated with a class label that denotes the skill level (expert or novice).

## 3.3 Feedback Generation Problem

Here, we define the feedback generation problem in an expert-novice perspective. We acknowledge that there may be more than 2 levels of expertise in some SBT applications. However, this can be easily addressed using the one-vs-rest approach.

In SBT, the feedback generation problem is to find the optimal action that can be taken to change a novice instance to an expert instance. Suppose the dataset $\mathcal{D}$ consists of $m$ features, $n$ instances defined by the feature vector $\mathbf{x} = (x_1, ..., x_m)$, and associated with a class label $y \in \{0, 1\}$ (1: expert, 0: novice). $H(\mathbf{x})$ is a prediction model learnt over $\mathcal{D}$. The feedback generation problem can then be defined as follows.

**Problem 1.** *Given a prediction model $H(\mathbf{x})$ and a novice instance $\mathbf{x}$, the problem is to find the optimal action $A : \mathbf{x} \to \mathbf{x}_f$ that changes $\mathbf{x}$ to an instance $\mathbf{x}_f$ under limited cost $C$ such that $\mathbf{x}_f$ has the highest probability of being in the expert class:*

$$\underset{A: \, \mathbf{x} \to \mathbf{x}_f}{argmax} \quad H(\mathbf{x}_f)$$

$$subject \ to \quad \mathscr{C}(\mathbf{x}, \mathbf{x}_f) \leq C,$$

where, feedback $A : \mathbf{x} \to \mathbf{x}_f$ involves one or multiple feature changes (increase/decrease). For example, $A : (force = \mathbf{0.2}, duration = 0.3) \to (force = \mathbf{0.5}, duration = 0.3)$ is "increase $force$ to 0.5". $\mathscr{C}(\mathbf{x}, \mathbf{x}_f)$ is the cost function measuring the potential cost of feedback $A$. In SBT, $\mathscr{C}(\mathbf{x}, \mathbf{x}_f) = \|\mathbf{x} - \mathbf{x}_f\|_0$, i.e., the number of changed features. The cost limit $C$ is often a small integer such as 1 or 2 in SBT so as to meet the requirements discussed in Section 1.

## 4 Proposed Method

To tackle the feedback generation problem, we propose the use of a neural network as the prediction model $H(\mathbf{x})$ and introduce a method that directly generates feedback from the neural network. Let $H_{\boldsymbol{\theta}}(\mathbf{x})$, with parameters/weights $\boldsymbol{\theta}$, be the neural network learnt with respect to the loss function $J(\boldsymbol{\theta}, \mathbf{x}, y)$, where $\mathbf{x}$ is the input or feature vector, $y$ the class value associated with $\mathbf{x}$, and $y^*$ the target class we want $\mathbf{x}$ to be in.

Recall that during the training process, the weights $\boldsymbol{\theta}$ are updated so that the loss $J(\boldsymbol{\theta}, \mathbf{x}, y)$ is minimized. Therefore, if we keep $\boldsymbol{\theta}$ fixed while the input $\mathbf{x}$ is updated so that $J(\boldsymbol{\theta}, \mathbf{x}, y)$ is maximized, we can get a new instance that has high confidence of being in the opposite class to its original class $y$ [Szegedy et al., 2013]. To maximize $J(\boldsymbol{\theta}, \mathbf{x}, y)$, the input can be updated in the positive direction of the gradient following Equation (1), where $\epsilon$ is the learning rate.

$$\mathbf{x} = \mathbf{x} + \epsilon \nabla_{\mathbf{x}} J(\boldsymbol{\theta}, \mathbf{x}, y) \tag{1}$$

This is the property that has been used to generate adversarial examples in image classification. Since adversarial examples require small perturbations in the input image, [Goodfellow et al., 2014] applied a sign function to linearize the loss function around the current value of $\boldsymbol{\theta}$, as shown in Equation (2). This method updates all the pixels of the input image once to get small perturbations.

$$\mathbf{x} = \mathbf{x} + \epsilon \, sign(\nabla_{\mathbf{x}} J(\boldsymbol{\theta}, \mathbf{x}, y)) \tag{2}$$

Equation (1) works well for two-class tasks. However, for multi-class tasks, there are more than one opposite classes to $y$. This means using Equation (1) cannot guarantee the new instance has high confidence in the target class $y^*$. The alternative is minimizing the loss $J(\boldsymbol{\theta}, \mathbf{x}, y^*)$ with respect to the specific target class $y^*$ as defined in Equation (3).

$$\mathbf{x} = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} J(\boldsymbol{\theta}, \mathbf{x}, y^*) \tag{3}$$

Although Equation (3) works for both two-class and multi-class tasks, it still has two potential problems that limit its use for feedback generation in SBT. First, it may change all input

features, thus violating the constraint (e.g., $\|\mathbf{x} - \mathbf{x}_f\|_0 \leq C$) of Problem 1. Second, the update may explode the values of inputs to extremely small or large values, similar to the *exploding gradient problem* [Pascanu *et al.*, 2012]. However, in practice, some features may have a certain value range outside of which the feature is meaningless.

To solve the first problem, we introduce a $L1$ regularization term to $J(\boldsymbol{\theta}, \mathbf{x}, y^*)$ to control the sparsity of the change so as to generate simple feedback. The new loss function is defined in Equation (4), where $\lambda$ is the regularization parameter and $\mathbf{x}_0$ is the original input that needs to be changed.

$$J^{'}(\boldsymbol{\theta}, \mathbf{x}, y^*) = J(\boldsymbol{\theta}, \mathbf{x}, y^*) + \lambda||\mathbf{x} - \mathbf{x}_0||_1 \qquad (4)$$

To solve the second problem, we propose a bounded update approach (see Equations (5) and(6)) as an alternative to Equation (3). It incorporates the value range (defined by lower and upper bounds) of a feature into the update to ensure the updated feature value is still within range.

$$\mathbf{x} = \mathbf{x} - \epsilon\mathcal{S}_{\mathbf{x}}\Big(\mathbf{x}\mathcal{S}_{\mathbf{x}} - \frac{\mathbf{a}}{2}\big(1 + \mathcal{S}_{\mathbf{x}}\big) + \frac{\mathbf{b}}{2}\big(1 - \mathcal{S}_{\mathbf{x}}\big)\Big) \qquad (5)$$

$$\mathcal{S}_{\mathbf{x}} = sign(\nabla_{\mathbf{x}}J^{'}(\boldsymbol{\theta}, \mathbf{x}, y^*)) \qquad (6)$$

$\mathcal{S}_{\mathbf{x}}$ is the sign of the partial derivative of $J^{'}(\boldsymbol{\theta}, \mathbf{x}, y^*)$ with respect to $\mathbf{x}$. The upper and lower bounds of $\mathbf{x}$ are $\mathbf{a}$ and $\mathbf{b}$ respectively, i.e. $x_i \in [a_i, b_i]$.

According to Equation (5), if the gradient $\nabla_{x_i}J^{'}(\boldsymbol{\theta}, \mathbf{x}, y^*)$ is positive (i.e., $\mathcal{S}_{x_i} = 1$), the update will become $x_i = x_i - \epsilon(x_i - a_i)$ which means $x_i$ moves a small step towards its lower bound $a_i$. Similarly, a negative gradient gives $x_i = x_i + \epsilon(b_i - x_i)$, a move towards its upper bound $b$. No update will be applied if the gradient is zero, as in this case, $\mathcal{S}_{x_i} = 0$. This bounded update not only guarantees the correct update direction to minimize the loss, but also ensures that $x_i \in [a_i, b_i]$ always holds true (see Lemma 1 and proof).

**Lemma 1.** *If* $a_i \leq x_i \leq b_i$, $0 < \epsilon \ll 1$, *and* $x_i^{'} = x_i - \epsilon\mathcal{S}_{x_i}\big(x_i\mathcal{S}_{x_i} - \frac{a_i}{2}(1 + \mathcal{S}_{x_i}) + \frac{b_i}{2}(1 - \mathcal{S}_{x_i})\big)$, *then* $a_i \leq x_i^{'} \leq b_i$.

*Proof.* The sign function $\mathcal{S}_{x_i}$ only has 3 outputs: 1,0 or -1.

**Case 1.** *If* $\mathcal{S}_{x_i} = 0$, *then* $x_i^{'} = x_i$.

In this case, $a_i \leq x_i^{'} = x_i \leq b_i$ holds true.

**Case 2.** *If* $\mathcal{S}_{x_i} = 1$, *then* $x_i^{'} = x_i - \epsilon(x_i - a_i)$.

In this case, $x_i^{'} - a_i = (1 - \epsilon)(x_i - a_i)$ and $x_i^{'} - b_i = \epsilon(a_i - x_i) + (x_i - b_i)$. Then, $a_i \leq x_i \leq b_i$ and $0 < \epsilon \ll 1$ gives $x_i - a_i \geq 0$, $x_i - b_i \leq 0$ and $1 - \epsilon > 0$. Therefore, $x_i^{'} - a_i \geq 0$ and $x_i^{'} - b_i \leq 0$, that is, $a_i \leq x_i^{'} \leq b_i$.

**Case 3.** *If* $\mathcal{S}_{x_i} = -1$, *then* $x_i^{'} = x_i + \epsilon(b_i - x_i)$.

And in this case, $x_i^{'} - a_i = \epsilon(b_i - x_i) + (x_i - a_i)$ and $x_i^{'} - b_i = (1 - \epsilon)(x_i - b_i)$. Similarly, $b_i - x_i \geq 0$, $x_i - a_i \geq 0$ and $1 - \epsilon > 0$ gives $x_i^{'} - a_i \geq 0$ and $x_i^{'} - b_i \leq 0$, that is, $a_i \leq x_i^{'} \leq b_i$.

To conclude, in all cases, $a_i \leq x_i^{'} \leq b_i$ holds true. $\qquad\square$

Equations (4), (5) and (6) give the definition of the proposed "neural network-based feedback (NNFB) method". NNFB takes a novice instance $\mathbf{x}$ as input, iteratively updates $\mathbf{x}$ (different from the one-time-update in generating adversarial examples) until it converges or meets the terminating criteria. Let the generated new instance be $\mathbf{x}_f$, the feedback is then the action $A : \mathbf{x} \rightarrow \mathbf{x}_f$ (see the example in Problem 1).

When feedback is delivered, we need to ensure that it contains only $C$ features. Although, the $L1$ regularization reduces the number of feature changes in general, in the absence of valid feedback with a low number of feature changes, it may still result in ones with higher numbers of feature changes. To overcome this issue, we suggest a post-selection process that iteratively tests all feature changes and select the ones with $C$ or less changes that result in the best improvements.

The proposed method (NNFB) is easily generalizable to different SBT applications. First, the regularization term in $J^{'}(\boldsymbol{\theta}, \mathbf{x}, y^*)$ can be adjusted accordingly for different applications (for example, $L2$ norm for applications that prefer small changes). Furthermore, NNFB offers flexible control over feature changes as the lower and upper bounds are adjustable for different features and even for different input instances. For example, we can set $a_i = b_i = x_i$ for a categorical feature that cannot be changed, such as prior simulation experience. This flexibility also benefits those applications that have discrete cost functions as some explicit cost limits can be easily incorporated into the bounds.

## 5 Experimental Validation

In this section, we first describe the two real-world datasets that were used in the experiments. Then, we briefly introduce the existing methods that the proposed method was compared against, followed by the experimental setup. Finally, we discuss the experiment results.

### 5.1 Datasets

We tested our method on two real-world SBT datasets. These datasets were collected from a temporal bone surgical simulator designed to train surgeons in ear-related surgeries. 7 expert and 12 novice surgeons performed two different surgeries that require very different surgical skills: cortical mastoidectomy - dataset 1 ($\mathcal{D}1$) and posterior tympanotomy - dataset 2 ($\mathcal{D}2$). Surgical skill is defined by 6 numeric skill metrics: stroke length, drill speed, acceleration, time elapsed, the straightness of the trajectory and drill force (see example in Section 3.2 for more details). The skill metrics were recorded by the simulator at a rate of approximately 15 Hz. Overall, $\mathcal{D}1$ includes 60K skill instances (28K expert and 32K novice) while $\mathcal{D}2$ includes 14K skill instances (9K expert and 5K novice). Both datasets were normalized to the range of $[0, 1]$ using feature scaling as follows.

$$\mathbf{x}^{'} = \frac{\mathbf{x} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}} \qquad (7)$$

where, $\mathbf{x}$ and $\mathbf{x}'$ are the original and scaled feature vectors respectively and $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ are the minimum and maximum feature values of $\mathbf{x}$ respectively.

## 5.2 Compared Methods

Existing feedback generation methods compared with NNFB are as follows.

- Split Voting (SV): This is the random forest based state-of-the-art generation method for providing real-time feedback [Zhou *et al.*, 2013a] as discussed in Section 2.

- Integer Linear Programming (ILP): This method solves the random forest feedback generation problem by transforming it to an integer linear programming problem [Cui *et al.*, 2015] as discussed in Section 2.

- Random Iterative (RI): This method randomly selects a feature and iteratively selects the best value among the feature's value partitions in the random forest [Cui *et al.*, 2015].

- Random Random (RR): This method randomly picks a feature from a novice instance and selects a random change to that feature as the suggested feedback.

## 5.3 Experimental Setup

For testing, we randomly chose one novice participant, then took all instances performed by this novice as the test set. The remaining instances were used for training. This simulates the real-world scenario of an unknown novice using the simulator. Parameter tuning was performed on the training data based on a 11-fold leave-one-novice-out cross-validation. In each fold, we took all instances from one randomly chosen novice as the validation set.

All methods were restricted to generate feedback with only one feature change, which is a typical requirement in SBT. This is a binary task as there are only 2 skill levels (*expert* and *novice*). All methods were then evaluated using 2 measures: 1) efficiency and 2) effectiveness. Overall, a good feedback generation method should have high effectiveness and high efficiency (low time-cost).

Efficiency was measured using the time-cost (in seconds) spent on average to generate feedback for one novice instance. The novice instance $\mathbf{x}$ will be changed to the target instance $\mathbf{x}_f$ by the feedback $A : \mathbf{x} \rightarrow \mathbf{x}_f$ (see Section 3.3). Thus, we use the quality of the target instances (i.e., $\{\mathbf{x}_f\}$) to measure the effectiveness of the feedback. As defined in Equation (8), effectiveness $\mathscr{E}$ is the percentage of expert instances in $\{\mathbf{x}_f\}$.

$$\mathscr{E} = \frac{|\{\,\mathbf{x}_f\,|\,\mathbf{x}_f\ is\ an\ expert\ instance\}|}{|\{\mathbf{x}_f\}|} \quad (8)$$

However, how instances are classified is dependent on the classifier used. To obtain more convincing results, we used 6 classifiers of different types for evaluation. The evaluation classifiers are: neural network (NN), random forest (RF), logistic regression (LR), SVM (RBF kernal), naive Bayes (NB) and KNN ($K = 10$). A generation method that scores consistently high levels of $\mathscr{E}$ across classifiers is deemed effective.
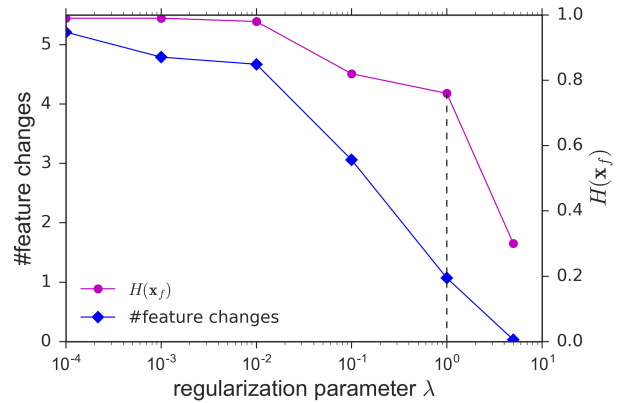


Figure 3: With the increases of $\lambda$, the number of feature changes in the feedback decreases but the confidence of $\mathbf{x}_f$ being in the expert class ($H(\mathbf{x}_f)$) remains high.

Experiments were carried out on a typical PC with 2.40GHz CPU. The ILP solver used for the ILP method was CPLEX[1] as suggested by the authors, and the neural network/random forest implementations we used were from scikit-learn. Default settings in scikit-learn were used for parameters not specifically mentioned here.

## 5.4 Parameter Tuning

Parameter tuning was performed on the training data with a 11-fold leave-one-novice-out cross-validation as mentioned above in Section 5.3. A two-layer neural network architecture was used for NNFB. For $\mathcal{D}1$, a neural network with 250 hidden neurons was selected for NNFB while a random forest with 120 trees was selected for SV, RI and ILP. For $\mathcal{D}2$, NNFB used a neural network with 120 hidden neurons while SV, RI and ILP used a random forest with 100 trees. These parameters were selected based on the turning point of the number of hidden neurons or the number of trees with respect to the mean squared error (MSE) of the neural network and random forest respectively.

In terms of the regularization parameter $\lambda$ in NNFB, Figure 3 indicates that larger $\lambda$ results in simple feedback with a fewer number of feature changes. When $\lambda = 1$, a feedback on average consists of only one feature change, but remains highly confident ($H(\mathbf{x}_f) > 0.7$) to change a novice instance to an expert instance. Therefore, we chose $\lambda = 1$ for NNFB. Since datasets have been normalised, the upper bounds for all features are 1 and the lower bounds are 0. Other settings for NNFB include Rectified Linear Unit (ReLU) activation function [Glorot *et al.*, 2011], cross entropy loss and learning rate $\epsilon = 1 \times 10^{-4}$.

## 5.5 Results

We first demonstrate the overall performance considering both effectiveness and efficiency. Figure 4 illustrates the effectiveness of each method as evaluated using 6 different evaluation classifiers with respect to the time-cost (inverses

---

[1]https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer

Table 1: The effectiveness (mean±std) tested by 6 evaluation classifiers. The best results are highlighted in bold.

| | | NN | RF | LR | SVM | NB | KNN |
|---|---|---|---|---|---|---|---|
| $\mathcal{D}1$ | RR | 0.19±0.06 | 0.23±0.10 | 0.35±0.07 | 0.27±0.06 | 0.32±0.12 | 0.30±0.05 |
| | RI | 0.44±0.07 | 0.39±0.04 | 0.50±0.08 | 0.46±0.06 | 0.42±0.12 | 0.40±0.08 |
| | SV | 0.63±0.07 | 0.59±0.06 | 0.60±0.07 | 0.62±0.06 | 0.50±0.11 | 0.53±0.07 |
| | ILP | 0.72±0.04 | **0.87±0.00** | 0.71±0.05 | 0.76±0.04 | **0.70±0.11** | **0.76±0.04** |
| | NNFB | **0.86±0.01** | 0.82±0.08 | **0.78±0.05** | **0.82±0.04** | 0.68±0.14 | 0.73±0.08 |
| $\mathcal{D}2$ | RR | 0.21±0.04 | 0.22±0.07 | 0.29±0.04 | 0.37±0.02 | 0.32±0.11 | 0.32±0.06 |
| | RI | 0.48±0.04 | 0.49±0.04 | 0.47±0.09 | 0.52±0.05 | 0.47±0.12 | 0.43±0.10 |
| | SV | 0.61±0.08 | 0.69±0.04 | 0.62±0.05 | 0.61±0.07 | 0.56±0.11 | 0.59±0.04 |
| | ILP | 0.88±0.04 | **0.90±0.02** | 0.79±0.07 | **0.77±0.03** | 0.78±0.12 | **0.84±0.09** |
| | NNFB | **0.92±0.02** | 0.82±0.06 | **0.81±0.07** | 0.72±0.05 | **0.79±0.11** | 0.81±0.07 |
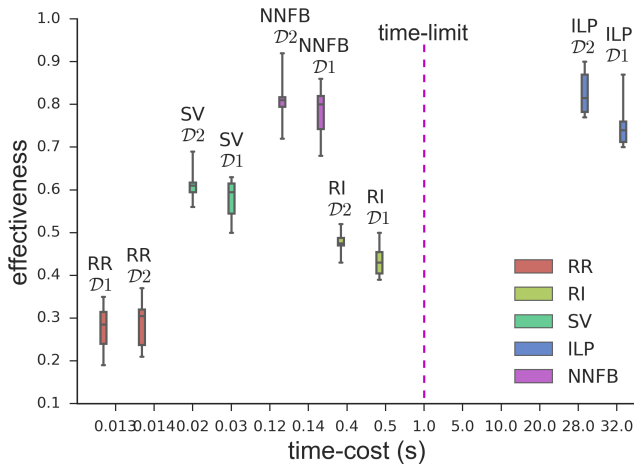


Figure 4: Box plot representing the performance of the 6 evaluation classifiers with respect to effectiveness and time-cost. Each method has two boxes that represent the 2 datasets $\mathcal{D}1$ and $\mathcal{D}2$. Colored view is recommended.

Table 2: The time-cost (mean±std in seconds) on average for generating one feedback, tested on datasets $\mathcal{D}1$ and $\mathcal{D}2$.

| | $\mathcal{D}1$ | $\mathcal{D}2$ |
|---|---|---|
| RR | 0.013±0.004 | 0.014±0.001 |
| RI | 0.504±0.098 | 0.401±0.020 |
| SV | 0.023±0.003 | 0.017±0.003 |
| ILP | 31.738±2.439 | 27.760±3.107 |
| NNFB | 0.142±0.029 | 0.121±0.016 |

NNFB. Thus, it can be concluded that in terms of both effectiveness and efficiency, the proposed method is the best suited for providing real-time feedback in SBT applications.

# 6 Conclusion and Discussion

In this paper, we introduced a technique for the adversarial generation of real-time feedback with neural networks for SBT. The proposed method (NNFB) applies a bounded adversarial update on the novice skill vector to generate an optimal expert skill vector in order to be used in the provision of feedback. To ensure that the suggested action is simple enough to practically undertake, we adopted $L1$ regularization to obtain feedback with a fewer number of feature changes. We explored theoretically the validity of NNFB, and showed empirically that it outperforms existing methods in providing effective real-time feedback.

Improving human performance in practice is a very challenging task. It involves many aspects of the learning process such as the learning environment, the task complexity, the knowledge level of the leaner, and the feedback intervention. In the future, we will deploy the proposed method to SBT environments and conduct user studies with human experts to further validate the method and investigate its effectiveness in teaching skills in practical applications.

to efficiency) for each dataset. As seen in the figure, the proposed method shows the desired performance of highest effectiveness at an acceptably low time-cost (within the real-time time-limit) when compared to the other methods. This proves that the adversarial technique can be used to generate effective and timely feedback for SBT. Note that the slightly higher variance of the NNFB method indicates the varying resistance of test classifiers to adversarial generation.

Detailed results for effectiveness of the feedback generation methods across the 6 evaluation classifiers is shown in Table 1 for the datatsets $\mathcal{D}1$ and $\mathcal{D}2$. On both datasets, NNFB achieved comparable performance to ILP and outperformed all others methods across all classifiers. However, as shown in Table 2, ILP violates the real-time time-limit as discussed in Section 2, and as such, will not be suitable for most SBT applications. Although both RR and SV are more efficient than the proposed method in feedback generation, they show significantly lower levels of effectiveness when compared to

# References

[Billings, 2012] DR Billings. Efficacy of adaptive feedback strategies in simulation-based training. *Military Psychology*, 24(2):114, 2012.

[Chen *et al.*, 2015] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *ICCV*, pages 2722–2730, 2015.

[Cosma and Stanic, 2011] Daniela Cosma and Mircea-Petru Stanic. Implementing a software modeling-simulation in military training. *Land Forces Academy Review*, 16(2):204, 2011.

[Cui *et al.*, 2015] Zhicheng Cui, Wenlin Chen, Yujie He, and Yixin Chen. Optimal action extraction for random forests and boosted trees. In *KDD*, pages 179–188, 2015.

[Davis, 2005] Walter D Davis. The interactive effects of goal orientation and feedback specificity on task performance. *Human Performance*, 18(4):409–426, 2005.

[de Groot *et al.*, 2011] Stefan de Groot, Joost CF de Winter, José Manuel López García, Max Mulder, and Peter A Wieringa. The effect of concurrent bandwidth feedback on learning the lane-keeping task in a driving simulator. *Human Factors*, 53(1):50–62, 2011.

[Forestier *et al.*, 2012] Germain Forestier, Florent Lalys, Laurent Riffaud, Brivael Trelhu, and Pierre Jannin. Classification of surgical processes using dynamic time warping. *Journal of Biomedical Informatics*, 45(2):255–264, 2012.

[Glorot *et al.*, 2011] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275, 2011.

[Goodfellow *et al.*, 2014] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv:1412.6572*, 2014.

[Lateef and others, 2010] Fatimah Lateef et al. Simulation-based learning: Just like the real thing. *Journal of emergencies, trauma, and shock*, 3(4):348, 2010.

[Ma *et al.*, 2017] Xingjun Ma, Sudanthi Wijewickrema, Yun Zhou, Bridget Copson, James Bailey, Gregor Kennedy, and Stephen OLeary. Simulation for training cochlear implant electrode insertion. 2017.

[Pascanu *et al.*, 2012] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *Computing Research Repository (CoRR) abs/1211.5063*, 2012.

[Rhienmora *et al.*, 2011] Phattanapon Rhienmora, Peter Haddawy, Siriwan Suebnukarn, and Matthew N Dailey. Intelligent dental training simulator with objective skill assessment and feedback. *Artificial intelligence in medicine*, 52(2):115–121, 2011.

[Rojas *et al.*, 2014] David Rojas, Faizal Haji, Rob Shewaga, Bill Kapralos, and Adam Dubrowski. The impact of secondary-task type on the sensitivity of reaction-time based measurement of cognitive load for novices learning surgical skills using simulation. *Stud Health Technol Inform*, 196:353–359, 2014.

[Steadman *et al.*, 2006] Randolph H Steadman, Wendy C Coates, Yue Ming Huang, Rima Matevosian, Baxter R Larmon, Lynne McCullough, and Danit Ariel. Simulation-based training is superior to problem-based learning for the acquisition of critical assessment and management skills. *Critical care medicine*, 34(1):151–157, 2006.

[Sutton and Barto, 1998] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[Sweller, 1988] John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive science*, 12(2):257–285, 1988.

[Szegedy *et al.*, 2013] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.

[Wijewickrema *et al.*, 2016] Sudanthi Wijewickrema, Yun Zhou, James Bailey, Gregor Kennedy, and Stephen O'Leary. Provision of automated step-by-step procedural guidance in virtual reality surgery simulation. In *VRST*, pages 69–72. ACM, 2016.

[Wijewickrema *et al.*, 2017] Sudanthi Wijewickrema, Bridget Copson, Yun Zhou, Xingjun Ma, Robert Briggs, James Bailey, Gregor Kennedy, and Stephen OLeary. Design and evaluation of a virtual reality simulation module for training advanced temporal bone surgery. 2017.

[Yang *et al.*, 2003] Qiang Yang, Jie Yin, Charles X Ling, and Tielin Chen. Postprocessing decision trees to extract actionable knowledge. In *ICDM*, pages 685–688, 2003.

[Yang *et al.*, 2007] Qiang Yang, Jie Yin, Charles Ling, and Rong Pan. Extracting actionable knowledge from decision trees. *TKDE*, 19(1):43–56, 2007.

[Zhou *et al.*, 2013a] Yun Zhou, James Bailey, Ioanna Ioannou, Sudanthi Wijewickrema, Gregor Kennedy, and Stephen OLeary. Constructive real time feedback for a temporal bone simulator. In *MICCAI*, pages 315–322. 2013.

[Zhou *et al.*, 2013b] Yun Zhou, James Bailey, Ioanna Ioannou, Sudanthi Wijewickrema, Stephen O'Leary, and Gregor Kennedy. Pattern-based real-time feedback for a temporal bone simulator. In *VRST*, pages 7–16, 2013.

[Zhou *et al.*, 2015] Yun Zhou, Ioanna Ioannou, Sudanthi Wijewickrema, James Bailey, Gregor Kennedy, and Stephen OLeary. Automated segmentation of surgical motion for performance analysis and feedback. In *MICCAI*, pages 379–386. 2015.