RankNEAT: Outperforming Stochastic Gradient Search in Preference Learning Tasks

Kosmas Pinitas
University of Malta
Msida, Malta
kosmas.pinitas@um.edu.mt

Konstantinos Makantasis
University of Malta
Msida, Malta
konstantinos.makantasis@um.edu.mt

Antonios Liapis
University of Malta
Msida, Malta
antonios.liapis@um.edu.mt

Georgios N. Yannakakis
University of Malta
Msida, Malta
georgios.yannakakis@um.edu.mt

ABSTRACT

Stochastic gradient descent (SGD) is a premium optimization method for training neural networks, especially for learning objectively defined labels such as image objects and events. When a neural network is instead faced with subjectively defined labels—such as human demonstrations or annotations—SGD may struggle to explore the deceptive and noisy loss landscapes caused by the inherent bias and subjectivity of humans. While neural networks are often trained via preference learning algorithms in an effort to eliminate such data noise, the *de facto* training methods rely on gradient descent. Motivated by the lack of empirical studies on the impact of evolutionary search to the training of preference learners, we introduce the RankNEAT algorithm which learns to rank through neuroevolution of augmenting topologies. We test the hypothesis that RankNEAT outperforms traditional gradient-based preference learning within the affective computing domain, in particular predicting annotated player arousal from the game footage of three dissimilar games. RankNEAT yields superior performances compared to the gradient-based preference learner (RankNet) in the majority of experiments since its architecture optimization capacity acts as an efficient feature selection mechanism, thereby, eliminating overfitting. Results suggest that RankNEAT is a viable and highly efficient evolutionary alternative to preference learning.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Genetic algorithms**; • **Human-centered computing** → *Human computer interaction (HCI)*; • **Applied computing** → *Computer games*.

KEYWORDS

Preference learning, neuroevolution, NEAT, RankNet, vision transformers, stochastic gradient descent, affect modeling, computer games

GECCO '22, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9237-2/22/07…$15.00
https://doi.org/10.1145/3512290.3528744

ACM Reference Format:
Kosmas Pinitas, Konstantinos Makantasis, Antonios Liapis, and Georgios N. Yannakakis. 2022. RankNEAT: Outperforming Stochastic Gradient Search in Preference Learning Tasks. In *Genetic and Evolutionary Computation Conference (GECCO '22), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3512290.3528744

1 INTRODUCTION

Forms of gradient descent are the natural choice of optimization method for training deep neural networks to predict objectively defined labels in tasks such as image and speech recognition, fraud detection, and event prediction. Over the last few years, we have witnessed a rapidly growing interest in the use of neural networks that are able to classify subjectively defined labels. This family of learning-to-rank or preference learning algorithms [9] that train neural networks—such as RankNet [2], DeepRank [29] and LambdaMART [3]—yield good performance by relying primarily on gradient descent methods. Subjectively defined labels, however, including human demonstrations (e.g. creative tasks, navigation traces and paths) or human annotations (e.g. of emotion or aesthetics) yield highly complex, deceptive and noisy loss landscapes for a neural network to learn. Assuming that the plasticity of neuroevolutionary processes would be beneficial for such loss landscapes, in this paper we test the hypothesis that evolutionary search would be a better optimizer for neural network training in preference learning (PL) tasks compared to stochastic gradient descent (SGD).

To test our hypothesis, this paper explores the efficacy of neuroevolutionary search in PL tasks by building on the efficient and popular RankNet [2] architecture and enhancing its search capacity through neuroevolution. In particular, we introduce a novel algorithm named *RankNEAT* that relies on the Siamese neural network architecture of RankNet and learns to rank via NeuroEvolution of Augmenting Topologies (NEAT) [36]. Unlike traditional gradient-based PL methods, RankNEAT resembles the process of plasticity [7], which induces changes in both the coupling strength and the spatial organization of synapses in biological neural networks. RankNEAT learns to rank subjectively defined labels with high degrees of accuracy through its ability to optimize the synaptic parameters such as the network's weights and the edge architecture simultaneously. We test RankNEAT (neuroevolution) and compare it against the vanilla RankNet (stochastic gradient decent) in the task of player affect modeling across three games, using the AGAIN

[23] dataset of arousal-annotated gameplay videos. Player modeling [46] is an important subfield in game research since it promotes the development of reliable human computer interaction systems and consequently improves the users' experience. Our current approach feeds images of gameplay to a pretrained vision transformer, while the last fully-connected layer of the network is then trained to predict ordinal values of arousal, using RankNet or RankNEAT. Results indicate that RankNEAT is superior to SGD (RankNet) in training PL models of arousal in the majority of experiments performed. Our key findings suggest that RankNEAT is a viable PL paradigm which achieves comparable or significantly higher performances to RankNet. In this first experiment, RankNEAT optimizes the edge topology of the networks' last layer, resembling an evolutionary feature selection strategy that eliminates unnecessary features from the observed input space. Additional studies should explore how RankNEAT performs in other subjectively defined tasks and hyperparameter setups, such as increasing the topological complexity.

This paper is novel in many ways. First, to the best of our knowledge, this is the first time a NEAT-based preference learner is introduced, combining a traditional learning-to-rank neural network architecture with neuroevolution. Second, RankNEAT is tested broadly across three dissimilar games from the same genre showcasing the robustness of the method for affect modeling. Third, the proposed approach is compared thoroughly against SGD (RankNet) across different games and hyperparameters. Finally, RankNEAT is combined with vision transformers (pretrained on ImageNet) enabling us to offer general-purpose representations for solving tasks with subjectively defined labels.

2 RELATED WORK

This section surveys related work on the performance comparison of evolutionary algorithms and gradient descent for training neural networks (see Section 2.1) and on the intersection of evolutionary search and affect modeling (see Section 2.2).

2.1 Evolution versus Backpropagation for Neural Network Training

Although SGD is currently the most widely applied training algorithm for neural networks, there has been a rapidly growing interest in employing evolutionary algorithms for optimizing deep learning models over the last years [34, 36, 47, 49]. Evolution and gradient descent through backpropagation (BP) are, however, fundamentally different and thus their comparison is a challenging task that numerous studies have tried to tackle. Indicatively, Mandischer [18] pitted evolutionary strategies (ES) against BP for neural network training on several benchmark problems, evaluating them based on the computational effort required to reach a certain error limit and their ability to converge. Results showed that while ES were good for training neural networks with non-differentiable activation functions, they still cannot compete with BP in large-scale problems. Siddique et al. [35] proposed a genetic algorithm (GA) capable of outperforming BP in function approximation in terms of convergence. Sexton et al. [33] compared a GA and BP on in-sample, interpolation, and extrapolation data in terms of root-mean-square error, number of epochs, and execution time. Results showed that GAs can be employed to strike a balance between model over-parameterization and model robustness. Gupta et al. [12] compared GAs and BP in terms of effectiveness, ease-of-use, and efficiency for training neural networks, showing that the former can provide better results in a chaotic time series problem. Gudise et al. [11] conducted a comparative study which demonstrated that the weights of a feedforward neural network tend to converge faster with the particle swarm optimization than with BP when it comes to function approximation. Finally, Sexton et al. [32] compared evolution and BP across ten real-world classification problems, showing that BP reached a higher classification error on average. Yannakakis et al. [42] employed supervised and genetic approaches to study the emergence of cooperative behavior among agents in a complex simulated environment and demonstrated that a genetic approach based on rewarding and minimal communication resulted in more efficient computational models of multi-agent spatial organization than supervised learning mechanisms. Zhang et al. [48] conducted several MNIST-based experiments in order to shed light on the relationship between the OpenAI ES and SGD by measuring the correlation between the approximated gradients computed by the algorithms and developing an SGD-based proxy for ES. The results obtained by the ES proxy are identical with those obtained by ES, and consequently, it holds that SGD with noise is equivalent to ES. Morse & Stanley [25] compared an evolutionary algorithm that evaluates individuals on a small number of training samples per generation and SGD on several benchmarks and showed that the former could optimize large neural networks about as fast and effectively as the latter.

In this work, we extend the literature by comparing neuroevolution and SGD performance in preference learning problems for the first time. In particular, we introduce a NEAT-based preference learner capable of predicting player arousal from gameplay footage and compare its performance with that obtained via SGD. Our results show that combining the topological and global optimization properties of NEAT with the Siamese network architecture of the traditional RankNet can result in robust learning-to-rank models that outperform the BP models trained via SGD.

2.2 Modeling Affect via Evolutionary Search

Affective computing is the study of emotions, their manifestations and expressions, and the ways to capture (model) them computationally [30]. While research at the intersection of affective computing and evolutionary algorithms has been active over the last decade, studies in the literature are still relatively sparse. For instance, Martinez et al. [21] presented a genetic search-based feature selection method for improving the accuracy of the affective models, comparing it against sequential forward feature selection and random search in a game survey dataset. Tahir et al. [37] introduced a binary chaotic genetic algorithm for feature selection, which achieved scores two times higher than a baseline genetic algorithm in identifying seven emotional states. Finally, Alvarez et al. [1] employed artificial evolution to select speech feature subsets that optimize the success rate of emotion recognition.

When it comes to games, the domain we study in this paper, player modeling [46] refers to the study of models that accurately predict how a player behaves and feels while playing a game. Affect models based on gameplay can provide valuable insights into how

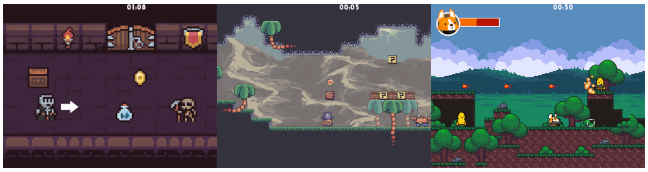


Figure 1: The three games used in this study. From left to right: Endless, Pirates!, Run'N'Gun.

players interact with games. As input of such models, most studies apply domain knowledge by manually authoring high-level hand-crafted gameplay features. For instance, Frommel et al. [8] employed the input parameters on a graphics tablet and in-game performance to detect the players' current emotional state. Similarly Melhart et al. [24] showed that hand crafting features that describe the player's input, the artificial agents' actions, and the gameplay context on a high level can yield general models of player arousal.

Although domain knowledge may lead to remarkable results, hand-crafted features do not necessarily reduce the data needs of the algorithms but do introduce a critical data preprocessing step. Automated feature extraction, on the other hand, may address such issues. Literature in this vein is fairly sparse. Ng et al. [28] used a deep Convolutional Neural Network (CNN) pretrained on the generic ImageNet dataset to perform emotion recognition on small datasets. Makantasis et al. [16] employed three CNN architectures to predict player arousal from gameplay footage, showcasing that a mapping between gameplay video streams and the player's arousal exists. The same authors also introduced a methodology for predicting arousal from audiovisual features and demonstrated that fusing high-level pixel and audio representations can yield highly accurate models of affect [17]. Finally, the study of Martinez et al. [20] seems to be the first to introduce a deep PL methodology for predicting emotional states from physiological signals. In particular, they showed that using auto-encoders and CNNs to find a mapping from raw signals to learnable features can outperform ad-hoc feature extraction and selection.

In contrast to all aforementioned studies, in this work we employ a pretrained Vision Transformer to extract high-level representations from gameplay footage and fine-tune our RankNEAT model to construct player arousal models for three platformer games. We also compare the behavior of evolutionary PL against gradient-based PL. Results verify that RankNEAT outperforms RankNet in most experiments performed due to the global optimization capabilities of the former. At the same time, its architecture search capacity corresponds to an effective mechanism of feature elimination.

3 CASE STUDY: PREDICTING PLAYER AFFECT

The neuroevolutionary learn-to-rank methodology proposed in this paper is tested on a challenging dataset of three games which includes many players' gameplay and emotion annotations. The games are created for the purposes of general affect modeling and are part of the AGAIN dataset [23]. In this paper we focus on the three games of the platformer genre featured in AGAIN as they offer sufficiently diverse gameplay properties without needing excessive computation for experimental validation. The three games

Figure 2: Arousal annotation of a player's gameplay footage using the time-continuous unbounded RankTrace protocol.

are shown in Fig. 1 and include **Endless**, an infinite runner where players must avoid obstacles while automatically moving ever rightward, **Pirates!**, a jumping platformer similar to *Super Mario Bros* (Nintendo, 1985), and **Run'N'Gun**, a more complex game which requires players to move while aiming and shooting at enemies. All games have arcade-style controls of varying complexity (with Run'N'Gun being the most complex), assign a score to the player for in-game actions, and finish after two minutes for the purposes of data collection.

The AGAIN dataset was collected through Mechanical Turk, with players first playing and then annotating each game. Annotation was done using a stimulated recall protocol, showing the player's own gameplay as a recorded video and requiring them to provide moment-to-moment annotation of arousal using the RankTrace annotation tool [15]. Figure 2 shows the arousal annotation trace: the player can keep increasing or decreasing their arousal annotation (unbounded) and can view their entire annotation so far.

The arousal annotations are preprocessed before being used for modelling tasks. First, the trace is normalized to $[0, 1]$ with min-max normalization. Due to the reaction time between a stimulus and a player's emotional response, we processed the data into time windows of 3 seconds. In particular, we calculate the mean arousal value of 3-second time window which we then use as the subjectively defined label for training (see Section 4.2). Gameplay videos are captured at 24Hz, resulting in 72 frames per 3-second window. Each gameplay frame is rescaled to a 224 by 224 pixel RGB image, and the 72 frames of the 3-second window are processed iteratively through a Vision Transformer (see Section 4.1).

When aligning arousal time windows and gameplay frames' time windows, we apply 1 second lag (shifting the annotations 1 second back compared to the video data) to simulate delays in the annotation process [19, 24]. After processing the data into 3-second windows and data cleanup (e.g. videos with missing frames due to errors during video recording), the dataset across all three games

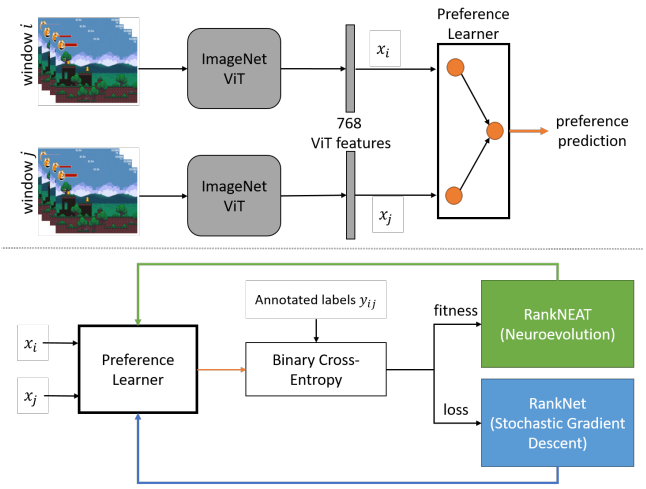


Figure 3: Illustration of the preference learning architecture (top image) and training methods (bottom image) used. ImageNet ViT (see Section 4.1) is not trainable; only the preference learner is trained. Gameplay footage feeds the Siamese PL architecture and the binary cross-entropy is calculated on the difference between the network's output and respective preference (annotated) label y_{ij} (see Section 4.2). The training parameters of the preference learner are optimized either: (a) via RankNEAT (Section 4.2) where the binary cross-entropy becomes the fitness, or (b) via RankNet [2] where the binary cross-entropy serves traditionally as the loss function.

consists of 262 gameplay videos corresponding to 111 different players. In particular, 103 gameplay videos remain for Endless (4, 120 time windows), 92 videos for Pirates! (3, 680 time windows), and 67 videos for Run'N'Gun (2, 680 time windows). Each video within the same game corresponds to a different player, which is important for cross-validation purposes (see Section 5).

4 METHODOLOGY

This section describes the main components of the algorithms examined in this paper including the Vision Transformers [6] used to extract high-level features from the video data and the two training methods used for our preference learning task: SGD and neuroevolution. An overview of our approach is presented in Fig. 3.

4.1 Vision Transformer

A Transformer is an architecture that utilizes an attention mechanism to discover dependencies between input and output. Although Transformers still employ an encoder and decoder, they eschew recurrence and thus require less training time while achieving better results than other sequence transduction models. The Vision Transformer (ViT) is a Transformer-based architecture for image classification tasks, using a single image as input and mapping it to a high-level vector representation, which, in turn, is fed to a multilinear perceptron responsible for conducting the classification task.

In this study, we use a ViT pre-trained on ImageNet 1K [5] as a backbone model to retrieve high-level vector representations of gameplay frame sequences. Since each time window consists of 72 frames, we replicated the weights of the first layer of ViT 72 times to account for input mismatch. Through this process, the $72 \times 224 \times 224 \times 3$ tensor of pixel values bound between $[0, 1]$ is transformed into a vector of 768 real values that represent higher representations of the gameplay video segment (see Fig. 3).

4.2 Preference Learner

Preference learning involves learning to distinguish data points in an ordinal manner [9], and thus can be applied to any supervised problem as long as the labels represent ordinal relationships. Since emotions are ordinal by nature [40, 41], in this study we develop a preference learner based on the RankNet architecture [2] to predict players' arousal using ViT representations of gameplay footage. Our arousal models are trained on pairs of gameplay windows.

Specifically, we formulate the arousal prediction task as a PL problem in the following way. Let us denote as $\mathcal{X}$ the space of ViT representations of gameplay footage windows and the data corresponding to the k-th gameplay video as $\mathcal{D}_k = \{(x_i, \lambda_i)\}_{i=1}^{N}$, where $x_i \in \mathcal{X}$, and $\lambda_i \in [0, 1]$ stands for the arousal annotation of the gameplay's i-th window. To employ a PL model, we transform $\mathcal{D}_k$ to $\tilde{\mathcal{D}}_k = \{(x_i, x_j, y_{ij})\}_{i,j=1}^{N}$, where y_{ij} equals 1 if $\lambda_i - \lambda_j > P_t$ and 0 if $\lambda_j - \lambda_i > P_t$. It should be noted that when $|\lambda_i - \lambda_j| < P_t$ the pair (x_i, x_j) is not included in the dataset $\tilde{\mathcal{D}}_k$. The preference threshold P_t controls whether or not the difference between the labels qualifies as a preference, while parameter k emphasizes the fact that pairs are produced with datapoints belonging to the same gameplay footage. Finally, the above data transformation procedure results in a perfectly balanced binary classification dataset.

As mentioned above, we adopt the RankNet model [2] for addressing the aforementioned PL problem. RankNet employs a neural network that receives as input pairs (x_i, x_j) and their respective labels y_{ij} and outputs $z_{ij} = f(x_i) - f(x_j)$, where f is a scalar function computed by the neural network. RankNet training aims to estimate the parameters of the neural network that minimize the binary cross-entropy loss of $\sigma(z_{ij})$ with respect to y_{ij}, where $\sigma(\cdot)$ is the sigmoid logistic function. In our experiments, we consider linear functions f, that is neural networks with no hidden layers, and we estimate the parameters of f using two fundamentally different optimization methods: SGD with backpropagation—as traditionally employed in RankNet training—and neuroevolution, as described below through RankNEAT.

RankNEAT. NeuroEvolution of Augmenting Topologies is an established algorithm [36] which goes beyond earlier approaches to neuroevolution which represented only the weights of the network as a vector in the genotype. While the typical NEAT algorithm starts from a minimal network (with only input and output nodes) and expands it with new nodes and edges, in this paper we use a simplified version of NEAT which does not add new nodes and thus does not expand the size of the network. It should be noted, however, that other features of NEAT which are crucial to its success, such as speciation and custom operators for adding and removing edges are maintained. In RankNEAT we use NEAT to train our RankNet model by optimizing the parameters of the linear function $f(\cdot)$ via weight mutations, crossover, adding or removing edges. Hence, its behavior resembles a feature elimination mechanism

which is essentially the same as setting the weight parameters of the deleted edges to zero.

We use the standard implementation of NEAT-Python [22] for running evolution. The initial population consists of p fully connected RankNet networks with random weights, which are evaluated and then split into species based on their topological similarities. The fitness of each individual is calculated by processing all pairs in the training set through the ViT and RankNet. Each pair consists of two frame sequences (x_i, x_j) and one ground truth preference (y_{ij}); each network is processed through the ViT and RankNet to derive $f(x_i)$ and $f(x_j)$ and finally to calculate the negative binary cross-entropy of the produced $\sigma(z_{ij})$. The mean of all cross-entropy scores for each pairing forms the fitness of the network and informs the selection of parents to mate and mutate.

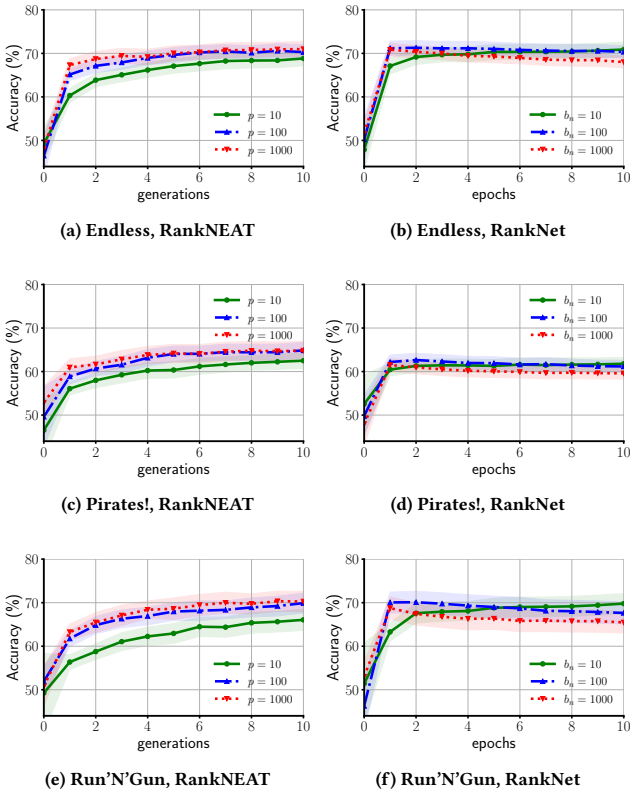
5 RESULTS

This paper aims to leverage neuroevolution for preference learning, assuming that its global optimization strategy may prove beneficial compared to gradient descent. Thus, the performance metric in our experiments is the accuracy in predicting the ranking between unseen pairs of gameplay footage windows. Specifically, we use a ten-fold cross-validation strategy for splitting the data into training and test sets. We ensure that data in the test set belongs to players that are absent from the training set. Therefore, we follow a leave-X-participants out method for cross-validation, where X is set between 6 and 11 participants depending on the game and fold. To address the randomness of weight initialization, genetic operators, and SGD, results are averaged across 5 independent runs [13] throughout the paper (including the 95% confidence interval between these 5 runs).

Due to the many hyperparameters of RankNet and RankNEAT, we perform a sensitivity analysis in Section 5.1 and report the main findings. Using the best parameters, Section 5.2 compares the performance of RankNet and RankNEAT for the three games, attempting to provide a fair ground of comparison in terms of computational effort. Throughout the experiments, we perform three tests per game by varying the preference threshold (P_t) between 0.15, 0.25 and 0.50. Higher threshold values can be more dependable in terms of the accuracy of the ranking but lead to significantly smaller datasets for training and testing.

5.1 Parameter Tuning

Several training hyperparameters control the behavior of both NEAT and SGD as optimizers. Parameter setting is often achieved through empirical trial-and-error processes. In terms of RankNet, we tune the batch size since the benefits of the adjustment of this parameter is two-fold. On the one end, the batch size is inversely proportional to the number of updates per epoch, affecting the speed of the training process. On the other end, the ratio of learning rate to batch size is a key element influencing the SGD dynamics [14]. When it comes to RankNEAT, there is no single correct choice of parameters for all problems due to interdependencies between hyperparameters such as population size and crossover [4]. Although the compatibility threshold ($c_t = 3$), elitism per species ($e_{ps} = 2$), and mutation rates (0, 0.5 for nodes and edges, respectively) were tuned according to some preliminary experiments, the population size p was adjusted based on a more systematic approach



(a) Endless, RankNEAT (b) Endless, RankNet

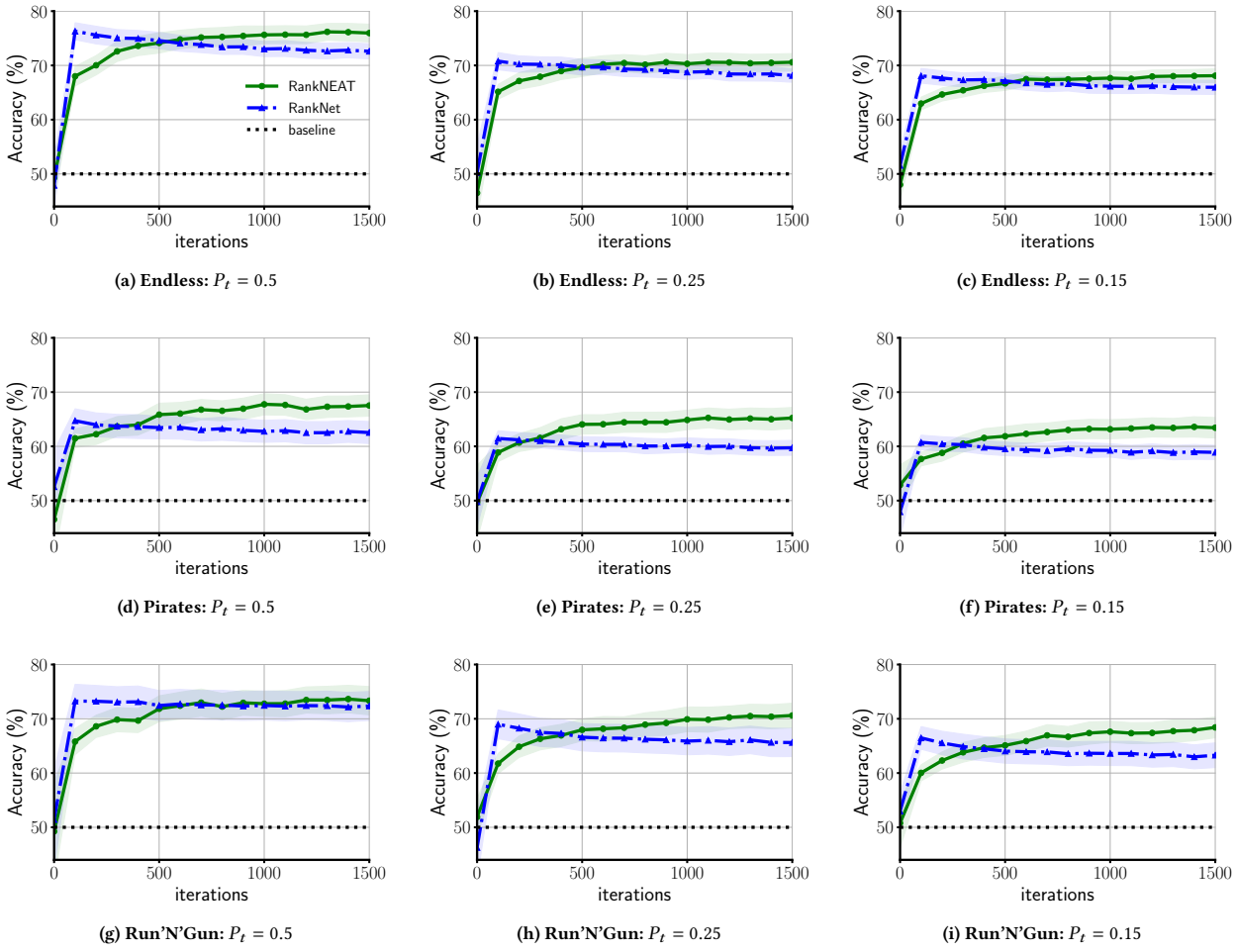
(c) Pirates!, RankNEAT (d) Pirates!, RankNet

(e) Run'N'Gun, RankNEAT (f) Run'N'Gun, RankNet

Figure 4: Impact of the population and batch size to the performance of the two algorithms.

since it influences both the training time and the robustness of the learner [31]. This section details our experiments on the three game test-beds for determining the optimal population size p and batch number b_n. It should be noted that other hyperparameters such as the learning rate for SGD, the compatibility coefficients and the survival threshold for NEAT were kept at their default values from their respective libraries. For space considerations we only present results with $P_t = 0.25$ in this section as experiments with the other two threshold values did not reveal any substantial differences for tuning the selected hyperparameters of RankNet and RankNEAT.

Figure 4 shows the progress of RankNet (SGD) and RankNEAT (neuroevolution) over 10 epochs and 10 generations, respectively. It should be noted that generations include more evaluations (depending on the population size p) than SGD epochs and thus the results between RankNet and RankNEAT are not comparable here. Evidence across all three games shows that large b_n values lead to a quick increase in accuracy for RankNet but subsequent epochs see a drop as the process overfits to the training set. Evidently, with small b_n values testing accuracy increases more slowly but has the potential to reach higher values. Based on this finding, we will use $b_n = 10$ as the best parameter in experiments of Section 5.2. Evolution on the other hand understandably benefits from larger populations: for instance with $p = 1000$ we see a quick optimization at the first generation but relatively small improvements after that.



(a) Endless: $P_t = 0.5$ **(b) Endless:** $P_t = 0.25$ **(c) Endless:** $P_t = 0.15$

(d) Pirates: $P_t = 0.5$ **(e) Pirates:** $P_t = 0.25$ **(f) Pirates:** $P_t = 0.15$

(g) Run'N'Gun: $P_t = 0.5$ **(h) Run'N'Gun:** $P_t = 0.25$ **(i) Run'N'Gun:** $P_t = 0.15$

Figure 5: Accuracy (and 95% confidence intervals) over evaluations for the RankNEAT and RankNet models. The black dotted line shows the (random) baseline accuracy of 50%.

Since with $p = 100$ the test accuracy reaches similar values as with $p = 1000$ within a few generations, we choose $p = 100$ in the experiments reported in the remainder of this paper for its significantly lower computational cost. We should note that optimization for $p = 10$ is slow but does not seem to converge within 10 generations, and it is possible that with more generations it could reach the performance of larger populations; however, we could not test this assumption in this paper.

5.2 RankNEAT versus RankNet

This section compares the best RankNEAT and RankNet models according to the hyperparameters investigated in Section 5.1. Following earlier comparative studies [18] we treat each training epoch and each individuals' fitness evaluation as having the same computational overheads and thus report test accuracy over *iterations* (with each generation of RankNEAT having p iterations, and each epoch in RankNet counting as 1 iteration). As in Section 5.1, we measure test accuracy based on a 10-fold leave-X-participants-out cross-validation, repeated and averaged from 5 independent runs. Based on Section 5.1, all RankNet experiments are performed with $b_n = 10$ (10 random pairs are sampled from the training set per epoch to calculate the gradient) and all RankNEAT experiments are performed with $p = 100$ (100 individuals in the population).

Figure 5 shows the progress over many iterations for the different datasets produced from different games and different preference thresholds (P_t). Even though we chose $b_n = 10$ because it did not overfit during the short training runs of Section 5.1, it is evident that as training progresses RankNet still is prone to overfitting. In all cases, test accuracy for RankNet drops after the first 100 iterations, often significantly (e.g. in Fig. 5a). On the other hand, evolution starts performing poorly but steadily increases at later generations. While evolution assesses its individuals in terms of accuracy in the training set and consistently improves there, it is evident that the models are also able to perform well (despite some fluctuations between generations) in the test set. At the same computational effort (1, 500 iterations), RankNEAT yields between 1% and 5% higher test

Table 1: Best accuracies (%) achieved by each model (RankNEAT vs RankNet) for the Endless, Pirates!, and Run'N'Gun test-beds, across three preference threshold values, P_t. Values are averaged across 5 independent runs. The average test accuracy of the best run (of 5) is also included within square brackets.

	$P_t = 0.5$		$P_t = 0.25$		$P_t = 0.15$	
	RankNEAT	RankNet	RankNEAT	RankNet	RankNEAT	RankNet
Endless	76.2 ±1.5 [77.3]	76.9 ±1.6 [77.9]	70.6 ±1.6 [71.7]	71.5 ±1.6 [72.1]	68.1 ±1.3 [69.2]	68.5 ±1.3 [69.1]
Pirates!	67.8 ±1.9 [69.6]	65.8 ±2.2 [66.9]	65.2 ±1.8 [66.5]	62.6 ±1.6 [63.5]	63.6 ±1.9 [64.7]	61.3 ±1.4 [62.1]
Run'N'Gun	73.6 ±2.5 [76.3]	73.7 ±3.0 [74.8]	70.6 ±2.3 [72.3]	70.2 ±2.3 [71.4]	68.4 ±1.9 [69.5]	67.8 ±2.0 [69.1]

accuracies from RankNet, on average, across the 9 experiments performed (with RankNEAT significantly outperforming RankNet in 5 of our 9 tests). Admittedly, in some of the experiments this is due to a noticeable drop in accuracy at later epochs for RankNet; in practice an early stopping criterion for RankNet would likely prevent this. Taking the best models discovered, on average, within these 1,500 iterations as a whole, we derive the results of Table 1. Here, we see that the results are comparable in several cases, although for the *Pirates!* game RankNEAT consistently performs better. It is worth noting that all models regardless of method underperform in Pirates! We hypothesize that RankNEAT may be able to perform better in more challenging problems. It is also worth noting that when we compare the best run of each algorithm, RankNEAT yields higher accuracies than RankNet in 7 out of 9 experiments.

Apart from the fact that RankNEAT performs global optimization, we expect that the custom operators that add or delete edges are especially powerful for this problem. As noted in Section 4.2, our version of RankNEAT does not allow for larger topologies to emerge but both speciation and topology changes in the edges are expected to have an impact. We expect that deleting an edge can act as a feature elimination mechanism and remove features that do not play a role in predicting arousal. Indeed, we observe that the best models of Table 1 for RankNEAT have between 5% and 6% fewer edges than the fully connected SGD network (RankNet with 768 edges). Due to the stochastic nature of the edge removal operator, this "feature selection" requires several generations to be impactful, but may largely be responsible for the good performance of the models.

We observe that models tend to be more accurate at higher preference thresholds. This is not surprising, and matches past findings [16], as ambiguous rankings are more aggressively cleaned. It is worth noting, however, that this comes at the cost of volume and generality of the dataset: indicatively, the datapoints at $P_t = 0.50$ are only 28% of the datapoints at $P_t = 0.15$ across all games.

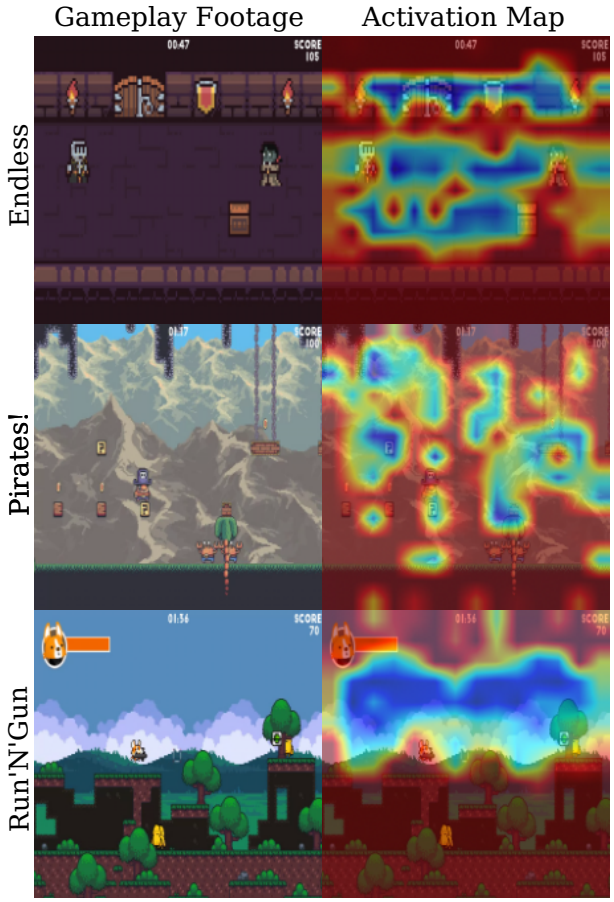


Figure 6: Eigen-CAMs for indicative frames of each game: saturated areas show pixels that are important predictors of arousal.

5.3 Qualitative findings

Results presented in the previous section show that player arousal can be modeled based on general-purpose representations such as video frames and, consequently, pixels. Drawing inspiration from the study of Makantasis et al. [16], we constructed the class activation maps (CAM) in order gain insights on which regions of the frames contributed the most to the final result. Our Eigen-CAM implementation relies heavily on the `PyTorch` library for CAM methods [10]. It should be noted that Eigen-CAM visualizes the principal components of the learned features, and thus it does not rely on the backpropagation of gradients or any other class relevance score [26]. We use RankNEAT, as it achieves the highest accuracies overall, to construct the visualization of Fig. 6. In these activation maps, warmer colors correspond to higher predictors of arousal value for a specific player in the test set. From the samples of Fig. 6, we observe that important predictors of arousal across games are regions containing information about the player, such as the avatar's position, life, game time, and score. Furthermore, the regions that contain information about the enemies' avatars are also very important for the model. In two out of three games, the model manages to mask out some of the redundant information

in the environment, such as empty space in Endless or the sky background in Run'N'Gun. For Pirates!, however, such patterns are less clear, and the model precludes the powerups from high importance regions. This may explain the relatively low accuracy value achieved on this game.

6 DISCUSSION

This work investigated the potential of neuroevolution for handling PL tasks when labels are defined in a subjective and ill-posed manner. We aimed to assess the power of NEAT as a preference learner by comparing the accuracy of NEAT and backpropagation in arousal prediction from general-purpose representations (gameplay videos) across three platformer games. To the best of our knowledge, this is the first time a NEAT algorithm has been used in a PL task. In particular, we studied the case of player affect modeling due to the fact that capturing the emotional manifestation of players is of great import for the domain of digital games [45][44]. The experiments indicate that RankNEAT can outperform RankNet by avoiding overfitting. There is evidence that RankNEAT's operators for deleting or adding edges is beneficial as a form of feature selection.

It should be noted that there is no straightforward way to compare evolution and SGD methods fairly. While past approaches have used CPU time [18, 42, 43], we instead matched epochs and individual evaluations as approximations of effort. That said, SGD selects a subset of the training data (in experiments in Section 5.2, this was $b_n = 10$) to derive a gradient while evolution evaluates cross-entropy in all pairings of the training set. Because we could multi-thread the evaluation of individuals in each generation, RankNEAT was between 57% and 72% faster in CPU times than RankNet per run, for the 1,500 iterations of Section 5.2 (tested on a CPU-only Intel Xeon, 132GB RAM). We could explore different ways of comparing the two methods in future work, as well as perform a more thorough tuning process for the other hyperparameters. In particular, parameters such as the survival threshold, elitism, and minimum species size can affect the crossover stage, increasing the diversity of the population. Thus, properly tuning these parameters may lead to better exploration of the search space.

Another worthwhile discussion is our choice of applying a more restrained version of NEAT for our experiment. The power of NEAT is arguably the fact that its operators can increase the network size with new nodes and more edges between these new nodes. While our version uses speciation as well as other operators of NEAT, speciation is more meaningful when networks differ in size. The initial population is fully connected while evolved individuals may have fewer edges (i.e. simpler topologies) but never more edges. Preliminary experiments with operators that could add nodes, however, led to an evolutionary process that quickly overfits to the training set while performing poorly on the test set. More experiments are necessary to investigate how this behavior can be countered, e.g. with different fitness evaluation schemes which assess a smaller subset of the training data similar to SGD's batch number. We will also consider and test alternative neuroevolutionary search methods such as covariance matrix adaptation evolution strategy, Differential Evolution, and their respective variants [27, 39] against the introduced algorithm in this paper. When it comes to RankNet there are a plethora of hyperparameters that might influence the performance of a model (e.g. network size, regularization) that need to be examined in a follow up study. The initial study presented here, however, contains a fair amount of hyperparameter tuning experiments for both algorithms as described in our results.

In terms of future research, there are several directions that we can follow to extend the goal of this work. An obvious next step on the scalability of this approach is testing the efficiency of RankNEAT to predict affect for the remaining six games of the AGAIN dataset, which includes racing games and shooter games [24]. A more important next step is testing whether the mapping between pixels and arousal found via neuroevolution can be general-purpose, for instance being able to predict arousal rankings in unseen games of the same genre. Earlier work [24] has shown that gameplay metrics (provided they are well-designed) can be robust predictors of arousal even in unseen games of the same genre. Establishing similar predictors through gameplay footage alone is arguably fundamental for general affect modelling [38]. Although this initial study used computer games as its test-bed domain, the proposed method is general and thus applicable to any affective computing and preference learning task; it remains to be found to which degree results hold for other tasks, datasets and domains of preference learning.

7 CONCLUSIONS

This paper introduced RankNEAT, an algorithm that transfers the benefits of the NEAT algorithm to learning-to-rank tasks in challenging domains with subjectively defined and biased labels such as affective computing. By leveraging pretrained computer vision models, we were able to evolve accurate models of arousal (with a test accuracy as high as 77% on average) using only gameplay footage. Comparing the performance of neuroevolution against stochastic gradient descent, which is the standard optimization method for PL, we observe that neuroevolution can overcome issues of overfitting. While SGD sometimes can find robust models early on, overfitting leads to a drop in accuracy that is difficult to control for. In contrast, RankNEAT continues to produce ever-more accurate models, and in some cases results had not converged at our ad-hoc cutoff point. Additional experiments, in more games and with more extensive exploration of hyperparameters (such as evolving larger topologies) are necessary to assess the true potential of this approach for player modeling, affective computing, and any machine learning domain that involves human demonstration and annotation.

ACKNOWLEDGMENTS

Kosmas Pinitas, Antonios Liapis and Georgios N. Yannakakis were supported by the European Union's H2020 research and innovation programme (Grant Agreement No. 951911). Konstantinos Makantasis was supported by the European Union's H2020 research and innovation programme (Grant Agreement No. 101003397).

REFERENCES

[1] Aitor Álvarez, Idoia Cearreta, Juan Miguel López, Andoni Arruti, Elena Lazkano, Basilio Sierra, and Nestor Garay. 2006. Feature subset selection based on evolutionary algorithms for automatic emotion recognition in spoken Spanish and

standard Basque language. In *Proceedings of the International Conference on Text, Speech and Dialogue*. Springer, 565–572.
[2] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*. 89–96.
[3] Christopher JC Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. Technical Report MSR-TR-2010-82. Microsoft Research.
[4] Kenneth A De Jong and William M Spears. 1990. An analysis of the interacting roles of population size and crossover in genetic algorithms. In *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature*. Springer, 38–47.
[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 248–255.
[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
[7] D Fair and BL Schlaggar. 2008. Brain development. In *Encyclopedia of Infant and Early Childhood Development*. Elsevier Inc., 211–225.
[8] Julian Frommel, Claudia Schrader, and Michael Weber. 2018. Towards emotion-based adaptive games: Emotion recognition via input and performance features. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play*. 173–185.
[9] Johannes Fürnkranz and Eyke Hüllermeier. 2011. Preference learning. In *Encyclopedia of Machine Learning*. Springer, 789–795.
[10] Jacob Gildenblat and contributors. 2021. PyTorch library for CAM methods. https://github.com/jacobgil/pytorch-grad-cam. Accessed 13 April 2022.
[11] Venu G Gudise and Ganesh K Venayagamoorthy. 2003. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the IEEE Swarm Intelligence Symposium*. 110–117.
[12] Jatinder ND Gupta and Randall S Sexton. 1999. Comparing backpropagation with a genetic algorithm for neural network training. *Omega* 27, 6 (1999), 679–684.
[13] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer New York Inc., New York, NY, USA.
[14] Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. 2017. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623* (2017).
[15] Phil Lopes, Georgios N Yannakakis, and Antonios Liapis. 2017. RankTrace: Relative and unbounded affect annotation. In *Proceedings of the International Conference on Affective Computing and Intelligent Interaction*. 158–163.
[16] Konstantinos Makantasis, Antonios Liapis, and Georgios N. Yannakakis. 2019. From pixels to affect: A study on games and player experience. In *Proceedings of the International Conference on Affective Computing and Intelligent Interaction*.
[17] Konstantinos Makantasis, Antonios Liapis, and Georgios N Yannakakis. 2021. The pixels and sounds of emotion: General-purpose representations of arousal in games. *IEEE Transactions on Affective Computing* (2021).
[18] Martin Mandischer. 2002. A comparison of evolution strategies and backpropagation for neural network training. *Neurocomputing* 42, 1-4 (2002), 87–117.
[19] Soroosh Mariooryad and Carlos Busso. 2013. Analysis and compensation of the reaction lag of evaluators in continuous emotional annotations. In *Proceedings of the IEEE Humaine Association Conference on Affective Computing and Intelligent Interaction*. 85–90.
[20] Hector P Martinez, Yoshua Bengio, and Georgios N Yannakakis. 2013. Learning deep physiological models of affect. *IEEE Computational intelligence magazine* 8, 2 (2013), 20–33.
[21] Héctor Pérez Martínez and Georgios N Yannakakis. 2010. Genetic search feature selection for affective modeling: a case study on reported preferences. In *Proceedings of the international workshop on Affective interaction in natural environments*. 15–20.
[22] Alan McIntyre, Matt Kallada, Cesar G. Miguel, and Carolina Feher da Silva. [n.d.]. neat-python. https://github.com/CodeReclaimers/neat-python. Accessed 13 April 2022.
[23] David Melhart, Antonios Liapis, and Georgios N Yannakakis. 2021. The Affect Game AnnotatIoN (AGAIN) Dataset. *arXiv preprint arXiv:2104.02643* (2021).
[24] David Melhart, Antonios Liapis, and Georgios N Yannakakis. 2021. Towards general models of player experience: A study within genres. In *Proceedings of the IEEE Conference on Games*.
[25] Gregory Morse and Kenneth O Stanley. 2016. Simple evolutionary optimization can rival stochastic gradient descent in neural networks. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 477–484.
[26] Mohammed Bany Muhammad and Mohammed Yeasin. 2020. Eigen-CAM: Class Activation Map using Principal Components. In *In Proceedings of the IEEE International Joint Conference on Neural Networks*.
[27] Ferrante Neri and Ville Tirronen. 2010. Recent advances in differential evolution: a survey and experimental analysis. *Artificial intelligence review* 33, 1 (2010), 61–106.
[28] Hong-Wei Ng, Viet Dung Nguyen, Vassilios Vonikakis, and Stefan Winkler. 2015. Deep learning for emotion recognition on small datasets using transfer learning. In *Proceedings of the ACM International Conference on Multimodal Interaction*. 443–449.
[29] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the ACM Conference on Information and Knowledge Management*. 257–266.
[30] Rosalind W Picard. 2000. *Affective computing*. MIT press.
[31] Stanley Rylander and Bart Gotshall. 2002. Optimal population size and the genetic algorithm. *Population* 100, 400 (2002), 900.
[32] Randall S Sexton and Robert E Dorsey. 2000. Reliable classification using neural networks: a genetic algorithm and backpropagation comparison. *Decision Support Systems* 30, 1 (2000), 11–22.
[33] Randall S Sexton, Robert E Dorsey, and John D Johnson. 1998. Toward global optimization of neural networks: a comparison of the genetic algorithm and backpropagation. *Decision Support Systems* 22, 2 (1998), 171–185.
[34] Takahiro Shinozaki and Shinji Watanabe. 2015. Structure discovery of deep neural network based on evolutionary algorithms. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*. 4979–4983.
[35] Nazmul M H Siddique and Mohammad O Tokhi. 2001. Training neural networks: backpropagation vs. genetic algorithms. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, Vol. 4. 2673–2678.
[36] Kenneth O Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.
[37] Madiha Tahir, Abdallah Tubaishat, Feras Al-Obeidat, Babar Shah, Zahid Halim, and Muhammad Waqas. 2020. A novel binary chaotic genetic algorithm for feature selection and its utility in affective computing and healthcare. *Neural Computing and Applications* (2020), 1–22.
[38] Julian Togelius and Georgios N. Yannakakis. 2016. General general game AI. In *Proceedings of the IEEE Conference on Computational Intelligence and Games*.
[39] Konstantinos Varelas, Anne Auger, Dimo Brockhoff, Nikolaus Hansen, Ouassim Ait ElHara, Yann Semet, Rami Kassab, and Frédéric Barbaresco. 2018. A comparative study of large-scale variants of CMA-ES. In *Proceedings of the International Conference on Parallel Problem Solving from Nature*. Springer, 3–15.
[40] Georgios N Yannakakis, Roddy Cowie, and Carlos Busso. 2017. The ordinal nature of emotions. In *Proceedings of the IEEE International Conference on Affective Computing and Intelligent Interaction*. 248–255.
[41] Georgios N Yannakakis, Roddy Cowie, and Carlos Busso. 2018. The ordinal nature of emotions: An emerging approach. *IEEE Transactions on Affective Computing* 12, 1 (2018), 16–35.
[42] Georgios N Yannakakis, John Levine, and John Hallam. 2007. Emerging cooperation with minimal effort: Rewarding over mimicking. *IEEE Transactions on Evolutionary Computation* 11, 3 (2007), 382–396.
[43] Georgios N Yannakakis, John Levine, John Hallam, and Markos Papageorgiou. 2003. Performance, robustness and effort cost comparison of machine learning mechanisms in FlatLand. In *Proceedings of the IEEE Mediterranean Conference on Control and Automation*.
[44] Georgios N Yannakakis and Manolis Maragoudakis. 2005. Player modeling impact on player's entertainment in computer games. In *In Proceedings of the International Conference on User Modeling*. Springer, 74–78.
[45] Georgios N Yannakakis and Julian Togelius. 2011. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing* 2, 3 (2011), 147–161.
[46] Georgios N Yannakakis and Julian Togelius. 2018. *Artificial intelligence and games*. Springer.
[47] Xin Yao and Yong Liu. 1997. A new evolutionary system for evolving artificial neural networks. *IEEE Transactions on Neural Networks* 8, 3 (1997), 694–713.
[48] Xingwen Zhang, Jeff Clune, and Kenneth O Stanley. 2017. On the relationship between the OpenAI evolution strategy and stochastic gradient descent. *arXiv preprint arXiv:1712.06564* (2017).
[49] Qiangfu Zhao and Tatsuo Higuchi. 1996. Evolutionary learning of nearest-neighbor MLP. *IEEE Transactions on Neural Networks* 7, 3 (1996), 762–767.