Fuzzy Aggregated Topology Evolution for Cognitive Multi-tasks

Fuzzy Aggregated Topology Evolution for Cognitive Multi-tasks

Noname manuscript No.
(will be inserted by the editor)

Fuzzy Aggregated Topology Evolution for Cognitive Multi-tasks

Iti Chaturvedi · Chit Lin Su · Roy Welsch

Received: date / Accepted: date

Abstract Evolutionary optimization aims to tune the hyper-parameters during learning in a computationally fast manner. For optimization of multi-task problems evolution is done by creating a unified search space with a dimensionality that can include all the tasks. Multi-task evolution is achieved via selective imitation where two individuals with the same type of skill are encouraged to crossover. Due to the relatedness of the tasks, the resulting offspring may have a skill for a different task. In this way, we can simultaneously evolve a population where different individuals excel in different tasks. In this paper, we consider a type of evolution called Genetic Programming (GP) where the population of genes have a tree like structure and can be of different lengths and hence can naturally represent multiple tasks.

Methods : We apply the model to multi-task neuroevolution that aims to determine the optimal hyper-parameters of a neural network such as number of nodes, learning rate and number of training epochs using evolution. Here each gene is encoded with the hyper parameters for a single neural network. Previously, optimization was done by enabling or disabling individual connections between neurons during evolution. This method is extremely slow and does not generalize well to new neural architectures such as Seq2Seq. To overcome this limitation, we follow a modular approach where each sub-tree in a GP can be a sub-neural architecture that is preserved during crossover across multiple tasks. Lastly, in order to leverage on the inter-task covariance for faster evolutionary search we project the features from both tasks to common space using fuzzy membership functions.

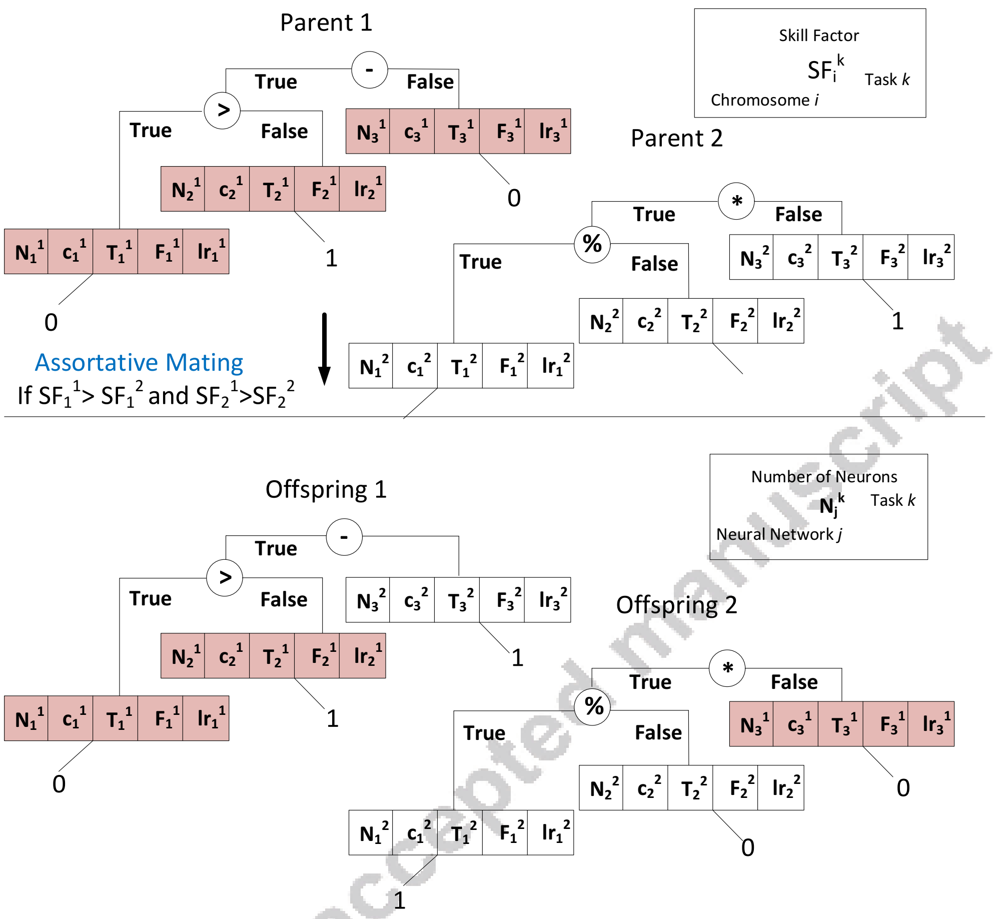
Conclusions : The proposed model is used to determine the optimal topology of a feed-forward neural network for classification of emotions in physiological heart signals and also a Seq2seq chatbot that can converse with kindergarten children. We can outperform baselines by over 10% in accuracy.

I. Chaturvedi
SCSE, Nanyang Technological University
E-mail: iti@ntu.edu.sg

1 Introduction

Recent studies show that cognitive functions such as memory or heart are strongly effected by emotions. Due to the complex nature of emotions we need balance multiple attributes such as Valence and Arousal simultaneously [1], [2], [3]. Multi-task optimization aims to leverage on annotated samples from different tasks simultaneously, since skills acquired by an individual for one task may be useful for another inter-related task [4]. Such a transfer optimization aims to harness past search experiences on one task to enhance the convergence efficiency of another task [5]. A limitation of the model is that an improvement in accuracy of one task may be accompanied by a deterioration in accuracy for another task. This happens because the covariance between the data samples for the same class label in different classes is low [6]. For example, we consider the classification of ECG into emotions. Here one task is classification of Valence into Joy or Sad and another task is the classification of Arousal into Fear or Calm. However, an increase in heart rate during Joy may not be well correlated with an increase in heart rate when experiencing Fear. In recent years, population based optimization such as an evolutionary search algorithm

Fig. 1 Crossover operation during neuroevolution using genetic programming. Here two different topologies are combined using an additional node that represents a mathematical operator such as $+$ or $*$. Hence, genetic programming results in an additional layer of neurons where the weights are determined by non-linear operators.



have been used to solve multi-task problems. Evolutionary multi-tasking aims to optimize each constitutive task absolutely, instead of having to establish any kind of tradeoff between individual tasks. To achieve this we evolve both the tasks in the same population of individuals, however we allow the offspring to imitate the skill factor (cultural trait) of the parents [7]. Following the principle of assortative mating where individuals prefer to mate with those belonging to the same cultural background, we only allow crossover among individuals skilled in a particular task [8]. The skill is calculated using the accuracy on the training dataset. Such a framework is intuitively more likely to result in new individuals that are competent in at least one task. The crossover operation will constantly generate new individuals from two existing elite individuals. The new individuals may have a higher skill factor for any of the tasks. Hence, even if we crossover two individuals with high skill factor for one task, we might generate a new individual with a higher skill factor for another task. This happens because the two tasks are related to each other.

Social data analysis is focusing on emotion recognition in natural language text. Annotating commonsense in terms of 'Pleasantness' and 'Attention' is a difficult and time-consuming task, hence unsupervised neural models are being used [9], [10]. Traditionally, neural networks are trained using backpropagation and the optimization of hyper-parameters is done using a validation set. In this paper, we apply this idea to neuroevolution where a population of neural networks with different topologies is evolved [11]. In particular, we wish to determine hyper-parameters such as learning rate and number of neurons in each layer. The shift from evolving fixed topologies to increasingly complex ones creates new challenges like crossing over structures (that is, combining the structures of two parent networks to create a parsimonious offspring network).

In [12], the authors used a database of safe mutations that only incurs a small cost that enables the learning of even hundreds of layers. Here, we leverage on the fact that the dictionary of features learned for one task maybe used in another task. Hence, we propose an aggregate model of sub neural networks where each component is trained on a specific task [13]. This allows us to even model temporal neurons (such as those used in a chatbot that can simulate how a human would behave as a conversational partner). Another limitation of previous evolutionary approaches is that they assume that all chromosomes are of equal length. However, each task may require a different number and type of neural layers hence a variable length chromosome is ideal for evolving multiple tasks together.

Genetic programming (GP) has a flexible variable length tree representation [7]. They have shown good accuracy in transfer learning tasks such as image classification and multi-lingual product classification. Hence, in this paper we consider neuroevolution in a genetic programming framework. GP evolves a population of potential models, each structured in a tree-like fashion with mathematical functions linking input nodes and constants. The mathematical functions enable highly nonconvex optimization compared to a fixed objective function. Our approach to neuroevolution involves three steps: (a) first we create a population of neural networks with a single output node (b) each input node in the GP corresponds to the output node of a single topology (c) we try to combine sub-component neural networks using an additional layer of mathematical operators and a single output neuron. Figure 1 illustrates the crossover operation during neuroevolution using genetic programming. Here two different topologies are combined using an additional node that represents a mathematical operator such as '+' or '*'. Hence, genetic programming results in an additional layer of neurons where the weights are determined by non-linear operators. In [8] the authors achieved a sparse solution for multiple tasks by thresholding, instead we can achieve sparsity in a Genetic Program by simply selecting a chromosome with fewer input nodes.

The organization of the paper is as follows: Section 2 reviews related works and dataset on image translation; Section 3 provides the preliminary concepts necessary to understand the present work; Section 4 details the proposed model for generating videos; in Section 5 we validate our method on two real world dataset and finally we provide conclusions in Section 6.

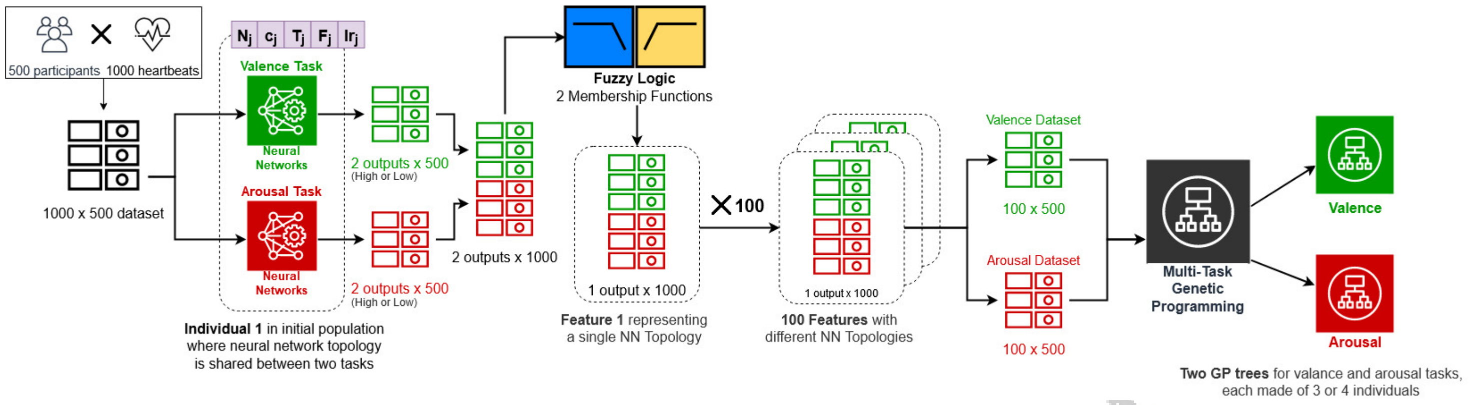
2 Related Works and Contributions

Multi-task optimization aims to optimize one or more tasks simultaneously so that we can get an improvement in accuracy over the 'no-transfer' case. We can leverage on the hidden covariances between related tasks. The inter-task covariance matrix provides a mapping from one task to another. In [14] the authors described multi-task optimization based on a Gaussian Process prior which has inter-task covariance specified by the product of the inter-task covariance matrix and the intra-task covariance matrix . However, computation of the covariance matrix becomes extremely slow with increase in the number of samples and tasks. Instead, we propose the use of a sampling based evolutionary approach where a knowledge base is created to gather information from different tasks. The samples in the knowledge base are accessible to all the tasks in the multi-tasking environment. The idea is similar to multi-objective optimization that aims to obtain a set of equally good solutions that hold a tradeoff among multiple conflicting objectives. However, multi-task optimization will obtain one solution for each task that are related but not necessarily conflicting each other. Hence, there is no trade-off between different tasks.

In [15], the authors reviewed the multi-task problem is different domains, however they don't consider evaluation on any specific dataset. Evolutionary multi-tasking in optimization leverages on the implicit knowledge transfer across different optimization tasks thereby achieving faster convergence. This additional factor for each task that is concurrently optimized results in a multi-factorial problem. To achieve implicit genetic transfer the modified child of task 1 is evaluated for task 2 and vice versa. For example, task 1 could be the path planning of two UAVs through a barrier and task 2 can be a more complex case where four UAV have to fly concurrently through a barrier. A limitation of the approach is that a unified genotype is used for both tasks that results in an overlap in phenotype. However, this can limit the accuracy on a single task. Instead, in this paper we consider sub neural networks that are completely optimized for a single task and there is no opportunity for a negative transfer.

Previous authors have used assumed a linear relationship between parents during crossover where the covariance matrix between two individuals is constant. In contrast, in this paper, we allow for mating between individuals that excel in different tasks hence the covariance matrix will keep changing. In order to account for this additional factor in this paper, we use assortative mating to achieve 'multi-task evolutionary optimization'. Evolutionary multi-tasking was also proposed

Fig. 2 FATE process on ECG data


500 participants
1000 heartbeats
1000 x 500 dataset
N_j c_j T_j F_j lr_j
Valence Task
Neural Networks
Arousal Task
Neural Networks
2 outputs x 500 (High or Low)
2 outputs x 500 (High or Low)
Individual 1 in initial population where neural network topology is shared between two tasks
2 outputs x 1000
Fuzzy Logic
2 Membership Functions
1 output x 1000
Feature 1 representing a single NN Topology
×100
1 output x 1000
100 Features with different NN Topologies
Valence Dataset
100 x 500
Arousal Dataset
100 x 500
Multi-Task Genetic Programming
Valence
Arousal
Two GP trees for valance and arousal tasks, each made of 3 or 4 individuals

in [16]. Here explicit transfer was achieved between the different tasks using a denoising auto-encoder. Instead of using Gaussian noise here they assumed that the solution of one task is the corrupted version of another task. To transfer a solution they multiply with the mapping matrix learned by the auto-encoder. The limitation of their approach is that the chromosomes are of equal length. In this paper, we use GP where chromosomes can be of variable length. Hence, we use fuzzy membership functions to perform explicit transfer between the tasks.

When doing optimization of a neural network we need to determine the optimal hyper-parameters such as number of neurons and learning rate. Traditionally, Bayesian optimization is used that makes the use of past evaluations when choosing the hyper-parameters set to evaluate next [17]. In this way a Bayesian prior is used to select the best hyper-parameters given the training accuracy. Bayesian optimization is computationally efficient; however the prior and posterior distribution is fixed and is unable to adapt to new tasks. Another limitation of this method is the cold start problem where it requires to evaluate several thousand solutions before convergence. Here we consider an evolutionary approach to hyper-parameter tuning that is flexible and highly parallelizable. In [18], the authors evolved neural networks to play the game of 'GO' where they maintain a population of neurons with a fixed number of connections but may allocate them arbitrarily among the units in the input and the output layers.

In order to apply this type of neuroevolution to multi-task optimization we consider the use of Genetic Programming. The modular nature of a Genetic Program tree makes it ideal for transfer learning across different tasks. Next, Fuzzy logic is used to model the inter-task covariance matrix. The resulting Fuzzy Aggregated Topology Evolution (FATE) framework has immense applications in symbolic regression, classification, automatic model design and real parameter optimization. In our model, the solution will be a GP tree where each leaf node is a single neural topology. Here, we can efficiently merge or delete layers in a neural network using simple mathematical operators. In order to represent a neural topology at a single leaf node in the GP we consider neural networks with a single output neuron. In this way, the feature learned at the output neuron for each topology can be concatenated into a vector of input features. The task of evolution is to select the best sub-set of topologies and aggregate them into a single neural topology.

In [19], the authors encode the genetic program as a directed acyclic graph to represent the neural network architecture. The limitation of this method is that the feature vector has to be padded with zeros so as to combine different architecture outputs. Evolution was used to train an ensemble of neural networks in [20] such that the covariance between the models in minimal. By decomposing a problem into sub-problems such a model is more robust compared to a single model. Multi-objective evolution can also be used to create a set of competing classifiers [21] where different models perform well on different objectives.

We can summarize the main contributions of this paper as follows:

1. We use Fuzzy logic to predict the inter-task covariance matrix where each membership function corresponds to a single task.
2. To evolve the neural topologies for different tasks, we consider Genetic Programming where each chromosome can be of a different length that is ideal to model multiple tasks.
3. Each leaf node of the optimal GP is a neural topology with different hyper-parameters. In this way, we

can control the training of sub-components in the aggregated model for different tasks.

4. In contrast to previous authors we are also able to optimize temporal models with inter-connected neurons.

Emotions can be represented in a two dimensional space of 'Arousal' and 'Valence'. Arousal is a state of being alert, on the other hand 'Valence' quantifies the attractiveness of an event. For example, high arousal would result in 'Fear' emotion and high valence would result in a feeling of 'Joy'. Heart signals are annotated for emotions in multi-modal studies of a person's personality and mood. Here all participants watched a set of short videos and were then profiled according to their personality traits and their mood based on the heart signal recorded [22]. The annotation is done by each participant using a questionnaire rating the intensity of emotions they feel after watching the video. Figure 2 illustrates the complete FATE process for ECG data.

We also evaluate the proposed model on a personal digital assistant for children. Personal digital assistants can act as secretary in doing activities such as scheduling of tasks, sending emails, making reservations, etc. For example, a 6-year old Dallas girl prompted Alexa to order her a Dollhouse [23]. Secondly, it can help people who suffer from social isolation such as the elderly. Rule based chatbots such as Siri are easy to train, however they lack memory and variations in response. In [24], the authors moved from symbolic AI to sub-symbolic AI (such as memory models) to enhance the efficiency of dialog systems. In retrieval based models we can take into account the related common-sense knowledge to select the appropriate response [25]. Such a model can become dull and a generation based model with an encoder-decoder can be used. It is ideal to combine rule-based and AI models for the best results [26]. In this paper, we consider the multi-task learning of a chatbot that can converse on different topics.

3 Preliminaries

In this section, we provide the preliminary theoretical concepts needed to understand the model. We begin with a description of Genetic Programming (GP) where chromosomes can be of variable lengths. Next, we describe two neural topologies that are optimized in this paper. Lastly, we describe the multi-task evolutionary framework for optimization using GP.

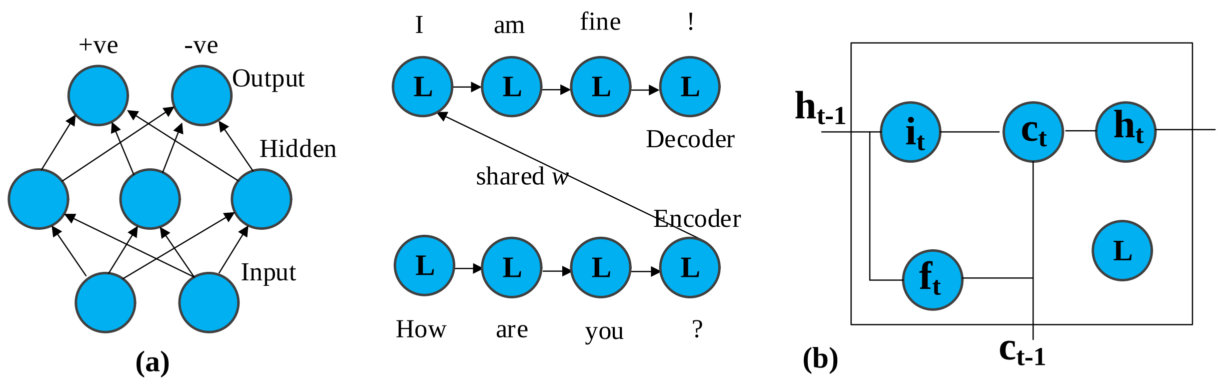
3.1 Genetic Programming

Genetic Programming (GP) evolves a population of potential models, each structured in a tree-like fashion, with mathematical functions linking input nodes and constants. The probability of a given tree surviving into the next generation depends on its classification accuracy on the training set. Fitness proportional selection combined with genetic operators such as crossover and mutation produce a new generation of offspring solutions. Here, each leaf node is the output node of a certain neural topology and all other nodes are functions such as the set $\boldsymbol{F} = \{+, -, *, /, sin, cos, exp, <, >, sqrt, cube\}$ where / denotes protected division that returns 1 if the denominator is 0. We first initialize a large population of GP's randomly.

Algorithm 1 describes a simple steady state GP classifier. We start with a root p and n_p children for the root. Next, for each child we randomly generate a new sub-tree until the maximum number of nodes is reached. The next generation is reproduced through the crossover operation. Here, we selectively rank the entire population of GPs based on classification accuracy on the training dataset. Next, two elite parent GP's are selected, and we randomly replace a sub-tree in one parent GP with a sub-tree from the second parent GP resulting in two new children GPs for the next generation. This process of evolution continues until convergence when the accuracy of elite GP in each generation does not improve any further. In order to achieve neuroevolution using genetic programming we initialize each leaf node with the output neuron from a single neural topology. The elite program in the last generation is used to determine the optimal topology that is made up of an aggregation of individual topologies. In order to combine the different sub-topologies we use an additional layer of neurons where the weights are determined using mathematical operators. For example, in Figure 1 offspring 1 is made up of three individual topologies and two mathematical operators namely $>$ and $-$. Each topology has its own set of hyper-parameters such as N_1^1 for number of neurons and lr_1^1 for learning rate. The input to the neuron is random class labels. The accuracy of the complete GP can be obtained by solving the entire decision tree.

A GP can get stuck in local optimum solution if the population similarity in each generation is low ($<0.6\%$). Hence, we keep the maximum number of nodes to 20 and depth to 4 levels, so that the similarity between trees is high and it can easily converge to the global optimum solution. Irrespective of the number of input features, each GP tree will have 20 nodes and discard the remaining features.

Fig. 3 Topology for (a) Feedforward NN (b) Seq2Seq network



Algorithm 1 Genetic Programming classification

```
1: **Input :**
2: Training and test data $(x_{ij})_{n\times T}$ for $n$ topologies and $T$ samples
3: Corresponding class labels $(y)_{1\times T} \in \{+ve, -ve\}$
4: **Output** :
5: GP ensemble topology
6: Class labels of Test Samples
7: % Initialize a population of random GP's
8: Initialize GP root $p$ and children p.children with length $n_p$
9: **repeat**
10:    $p.children[k] \leftarrow$ randomly generated subtree
11: **until** $k < n_p$
12: % Crossover Operations to generate new population
13: **repeat**
14:    Select Two Elite GP $p1$ and $p2$ based on Accuracy
15:    Select subtree1 = $p1.children[1 : k1]$ for any $k1$
16:    Select subtree2 = $p2.children[k1 + 1 : n_p]$
17:    Merge subtree1 and subtree2 resulting in two new children
18:    $l = l + 2$
19: **until** convergence
20: Each test sample is classified using predicted GP
21: **Accuracy** : # of correctly classified test samples
```

3.2 Neural Networks Topologies

We consider neuroevolution for two different types of neural networks. However, the model can be easily applied to optimize any other neural model. Figure 3 illustrates the feedforward and the Seq2Seq neural models. The feedforward neural networks are used to classify ECG samples as positive or negative. The Neural Network (NN) has at least three layers, namely the input layer of heartbeat samples, the hidden layer and the output layer for the class label as shown in Figure 3 (a). The weights of the edges connecting different layers are learned using backpropagation algorithm.

For example, we consider a feedforward neural network with an input, one hidden and an output layer. The continuous state $\hat{h}_j$ of the hidden neuron j, with bias b_j, is a weighted sum over all continuous input nodes $\boldsymbol{v}$ and is given by:

$$\hat{h}_j = b_j + \sum_i v_i w_{ij}, \tag{1}$$

where w_{ij} is the connection weight to hidden neuron j from a visible node v_i. Similarly, we can predict the value of neurons in the output layers using (1). Next, the binary state h_j of the hidden neurons in the output layer can be defined by a sigmoid activation function:

$$h_j = \frac{1}{1 + e^{-\hat{h}_j}}. \tag{2}$$

Lastly, we compute the change in weights as the difference between the predicted outputs and the target outputs :

$$\triangle w_{ij} = \alpha(< v_i h_j >_{output} - < v_i h_j >_{target}), \tag{3}$$

where α is the learning rate and $< v_i h_j >$ is the expected frequency with which visible unit i and hidden unit j are active together when the visible vectors are sampled from the training set and the hidden units are determined by (1).

The Seq2seq neural network maps an input sentence to an output sentence with a tag and attention value. The idea is to use two Long-Short Term memory (LSTM) that will work together with a special tying or sharing of weights to predict the next state sequence from the previous sequence. Unlike the feedforward NN, the neurons in an LSTM are inter-connected so that they can remember the past sequence of words in a sentence using memory states. The first LSTM is the encoder that learns to predict the next word in the input chat sentence and the second LSTM is the decoder that is trained to predict the next word in the response as shown in Figure 3 (b).

Each neuron in an LSTM is a cell made up gates that control which information to remember and which information to forget. Each gate has two associated weight matrices : input weights w_1 and memory weights w_2. The input from the previous hidden neuron h_{t-1} in a sequence of words is used to learn the weights of the next word. In addition, the cell has a state node c_t that determines if the current word should be remembered or forgotten. The value of c_t is determined using the input gate i_t, the forget gate f_t and the previous state c_{t-1} as shown in Figure 3 (b).

The model is trained using gradient descent similar to the one described for feedforward NN. The continuous state of each gate is determining as a weighted sum over all input nodes:

$$\begin{aligned}
i_t &= v_t w_i^1 + h_{t-1} w_i^2 \\
f_t &= v_t w_f^1 + h_{t-1} w_f^2 \\
c_t &= v_t w_c^1 + h_{t-1} w_c^2 \\
h_t &= v_t w_h^1 + h_{t-1} w_h^2 \qquad (4)
\end{aligned}$$

where the gates are updated using the error computed at the last hidden neuron and known target labels as shown in (3). The forget gate uses the sin activation function that transforms the input in the range $[0, 1]$ where a value 0 indicates that the information is forgotten.

3.3 Multi-Task Evolutionary Optimization

Multi-task optimization aims to model two or more task simultaneously via transfer of solutions between the domains of the two tasks. The features of each task will have domain specific contextual meaning to the optimization of that task. Hence, we need to model both the independent and the shared information between the tasks. In this paper, Multi-task evolutionary optimization aims to achieve this via 'assortative mating'.

Figure 1 illustrates a Genetic Program for multi-task evolution for two tasks. Let us denote the skill factor of a chromosome i for a particular task k as SF_i^k. Here we can compute the skill factor as the accuracy on the training dataset. Let us consider the optimization of a feedforward neural network j for task k with five hyper-parameters : (1) Number of hidden neurons N_j^k, (2) Regularization constant c_j^k, (3) Number of training epochs T_j^k, (4) Activation function F_j^k and (5) Learning rate lr_j^k. We consider a neural network with a single output neuron. Hence, for each training sample of task k we can create an output vector corresponding to the neural network with the specified hyper-parameters. For example, the chromosome of Parent 1 in Figure 1 is an ensemble three neural network topologies connected by two mathematical operators $>$ and $-$. Here we can use $>$ boolean operator on the output of neural network 1 and 2. If the output of 1 is greater than output of 2, then we select the value and class label for the 'True' branch and otherwise we select the value and class label for the 'False' branch. Next, we subtract the value from the output of topology 3 and check if its greater than a predefined threshold resulting in the final class label for the sample.

We start with a random population of chromosomes. The skill factor of each chromosome is evaluation for both the tasks. In the next generation, we can use assortative mating of two parent chromosomes resulting in two new offspring's. Assortative mating states that individuals prefer to mate with those belonging to similar cultural background. Hence, two parents with a higher skill in a particular task are mated. For example, in Figure 1 we compare the skill factors of both the parents for the two tasks. If $SF_1^1 > SF_1^2$ and $SF_2^1 > SF_2^2$ then we can conclude that both chromosomes are suitable for task 1 and hence are mated to generate two new offspring's that might have higher fitness for task 1. However, if the offspring has a higher skill for task 2, then in the next generation it will be mated with another offspring with higher skill for task 2. In this way, multi-task evolution uses a combined search space to optimize both tasks completely.

Figure 7 illustrates an example of multi-task GP. The first tree has a higher accuracy for Valence than Arousal. The second tree has higher accuracy for Arousal than Valence. The trees take a decision tree structure where all the leaf nodes are random binary class labels. The operators are all boolean tests that determine which class label is propagated to the leaf node. For example, in Figure 7 the predicted GP tree for Valence is made of neural topologies P12, P77 and P40. We can see that P12 was also used in the Arousal GP tree. In contrast to the traditional multi-task evolutionary optimization where a unified genotype is used to generate different phenotypes for different tasks. Hence a single chromosome encodes both the tasks. Here, the genotype for each task is different since each chromosome is only skilled in a specific task. However, sub-trees could be transferred between tasks during crossover.

4 Fuzzy Multi-task Optimization

In this section, we describe the Fuzzy classifier to detail the inter-task covariance matrix. Next, we describe the complete Fuzzy Multi-Task Optimization Framework.

4.1 Fuzzy Inter-task Covariance Matrix

Individual modeling of each task will not be very suitable for multi-task problems due to the ignorance of the inter-task latent covariance. For a pair of input samples i and $i+1$ from the same or different tasks we can define an inter-task utility function $q(i)$ as follows:

$$q_i = \begin{cases} 1, & \text{if } \sigma \geq 0.5 \\ 0, & \text{if } \sigma < 0.5 \end{cases}$$

where σ is the covariance between i and $i+1$. Hence, q_i is 1 if the covariance is high and q_i is 0 if the covariance is low. We can also define a long-term utility function over all samples as follows :

$$y_i = q_{i+1} + \beta q_{i+2} + \beta^2 q_{i+3} + \dots \tag{5}$$

where β are the unknown parameters. In this paper we employ Fuzzy Neural Networks to estimate the utility function.

We now detail the fuzzy model for shared inter-task dataset, the model for a single task is a special case of the former. The input is a vector of features where each feature is the output of a pre-trained sub-topology. We use one fuzzy membership function to represent each task. For example, one function can represent the valence and the other can represent the arousal of an individual. We consider four emotional dimensions for each ECG $e(t)$, calmness $m_c(t)$, fear $m_f(t)$, happiness $m_h(t)$ and sadness $m_s(t)$. The emotions have uncertainties which can vary in the range $m_c(t) \in [m_{c\min}, m_{c\max}]$, $m_f(t) \in [m_{f\min}, m_{f\max}]$, $m_h(t) \in [m_{h\min}, m_{h\max}]$ and $m_s(t) \in [m_{s\min}, m_{s\max}]$. It is to say that the uncertainty of the calmness $m_c(t)$ is bounded by its minimum value $m_{c\min}$ and its maximum value $m_{c\max}$. Similarly, the other emotional dimensions are bounded by their minimum and maximum values.

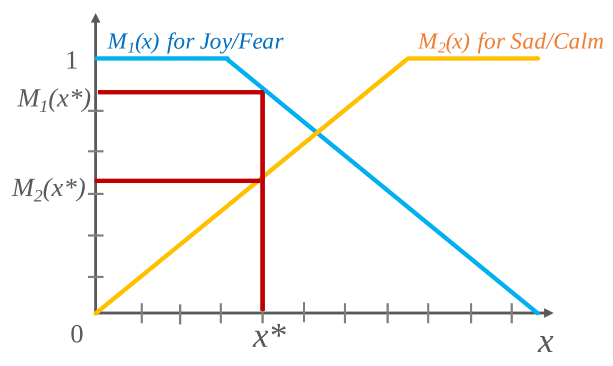


Fig. 4 Sample Membership functions for two ECG tasks

In order to determine the inter-task covariance matrix, we consider two membership functions. The partial membership to both the functions M_1 and M_2 such as 'Very Low, Low, High, Very High' can be determined using :

$$\begin{aligned} M_1(x) &= \frac{x - m_{h\max}}{m_{h\max} - m_{h\min}} \\ M_2(x) &= \frac{x - m_{s\max}}{m_{s\max} - m_{s\min}} \\ M_1(1/x) &+ M_2(1/x) = 1 \end{aligned} \tag{6}$$

The member functions are labeled "High" and "Low" as shown in Figure 4. The membership function M_1 is high for Joy and low for Sad. On the contrary the membership function M_2 is high for Sad and is low for Joy. For a given input sample x^* we can compute two values corresponding to $M_1(x^*)$ and $M_2(x^*)$. Lastly, fuzzy blending allows us to infer the overall fuzzy sentiment model using the following rules :

$$\begin{aligned} &\mathit{Rule}1: \text{ IF Joy is High and Sad is Low} \\ &\qquad\qquad \text{THEN } y = 1 \\ &\mathit{Rule}2: \text{ IF Joy is Low and Sad is High} \\ &\qquad\qquad \text{THEN } y = 0 \\ &\mathit{Rule}3: \text{IF Fear is High and Calm is Low} \\ &\qquad\qquad \text{THEN } y = 1 \\ &\mathit{Rule}4: \text{IF Fear is Low and Calm is High} \\ &\qquad\qquad \text{THEN } y = 0 \end{aligned} \tag{7}$$

where $y \in \{0, 1\}$ is the emotion class for a single task. Similar rules can be designed for the chatbot dataset.

For a set of input variables $x = (x_1, x_2, \dots, x_p)$ and output labels y, we consider a union of K fuzzy membership functions $M_1, M_2, \dots, M_K$, then the defuzzifier that maps the fuzzy set to a single output $\hat{y}$ is defined as :

$$\hat{y}_i = \sum_{l=1}^{K} w^l \left(\prod_{i=1}^{p} M_l(x_i) \right) / \sum_{l=1}^{K} \prod_{i=1}^{p} M_l(x_i) \tag{8}$$

where $w^l = m_{s\max}$ is the point at which $M_l(y) = 1$.

Figure 2 provides the flow diagram for computing the inter-task covariance matrix. We train a common neural topology with both the tasks dataset : Valence and Arousal. Next, the two output features extracted from both tasks are combined and used to train the Fuzzy Logic classifier. We use two membership functions to capture the labels for two tasks. The output of the Fuzzy classifier is a single output feature vector y. In order to train the multi-task GP we again separate the output feature into two tasks dataset.

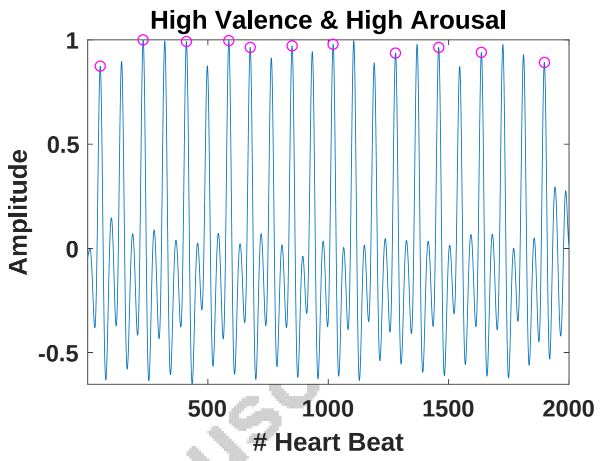
4.2 Aggregated Multi-task Neuroevolution Framework

Next, we can use the previously described multi-task optimization to collectively evolve several related tasks. Here, each objective is a separate skill or task. For example, in the ECG data one task is Valence classification and the second task is Arousal classification. Similarly, for the Children conversation data set, children with 1 year of English exposure is one task and children with 2 years of English exposure is the second task.

For each training sample with a known class label, we predict the output class node using different neural topologies. The genetic program uses mathematical functions to determine the optimal ensemble of topologies that can maximize the accuracy for all tasks simultaneously as described in the previous section. Each individual is now evaluated for each of the tasks. Next, we can assign a skill to the individual as the task with the highest fitness score after solving the genetic program. Assortative mating is used where two individuals with the same skill undergo crossover to produce new individuals in the next generation. By definition: A solution $x' \in X$ is called Multi-task optimal if there is no solution $x \in X$ such that $f_i(x) \leq f_i(x')$ for all $i = 1, 2, \ldots, k$ and $f_j(x) < f_j(x')$ for all objective functions with index j.

We first train a starting population of topologies for each task. Each topology is trained using a different set of hyper-parameters. The output features of each topology from both tasks are combined to determine the inter-task covariance matrix using a Fuzzy system. The transformed outputs for each task are used to train a multi-task genetic program using assortative mating. Figure 2 illustrates the entire process of FATE for ECG dataset. For each of the 500 participants we consider 1000 time samples in the ECG dataset and the labels for level of Arousal and Valence. A common topology is shared between the two tasks. We train the neural network independently for both tasks. The neural network has two outputs corresponding to high or low emotion. We can combine the output for both tasks resulting in a matrix of two outputs and 1000 participants. Next, we use the combined data to train the Fuzzy Logic Classifier. Here the output of the Fuzzy model is a single neuron that represents the shared neural topology. This process repeated for 100 different topologies resulting in a matrix of 100 outputs for 1000 participants. Finally, we again separated the features for the two tasks and train the Mutli-task GP model.

Fig. 5 Here we consider Task 1 where Valence is high (Joy) and Task 2 where Arousal is high (Fear). The R-R peaks are shown as circles. The Q peaks first reduce and then increase in amplitude.



5 Experiments

Validation of the proposed FATE (available on GitHub[1]) is done on two real world dataset : (1) Heart ECG data using Feed-forward NN (2) Chatbot for kids using Seq2seq model. Following previous authors we report the improvement in accuracy over baselines.

5.1 Physiological Signals ECG Dataset

The ECG signals were classified according to four emotional states: calm (low arousal), fear (high arousal), joy (high valence) and sad (low valence). First, we trained the model through manually annotated ECG signals from the DREAMER [27], Amigos [28] and Ascertain [29] databases. In order to collect ECG samples for a particular emotion, a video clip was shown to the participant and ECG was recorded at the end of the clip. Each clip targeted one of the following nine emotions: amusement, excitement, happiness, calmness, anger, disgust, fear, sadness and surprise. To avoid contaminating data recordings with multiple emotions, only the recordings captured during the last 60 seconds of each film clip were used for further analysis. A 5 second baseline recording showing a fixation cross was shown before each film clip in order to help the subject return to a neutral emotional state. Each participant performed an initial self-assessment for the emotion felt ranging from

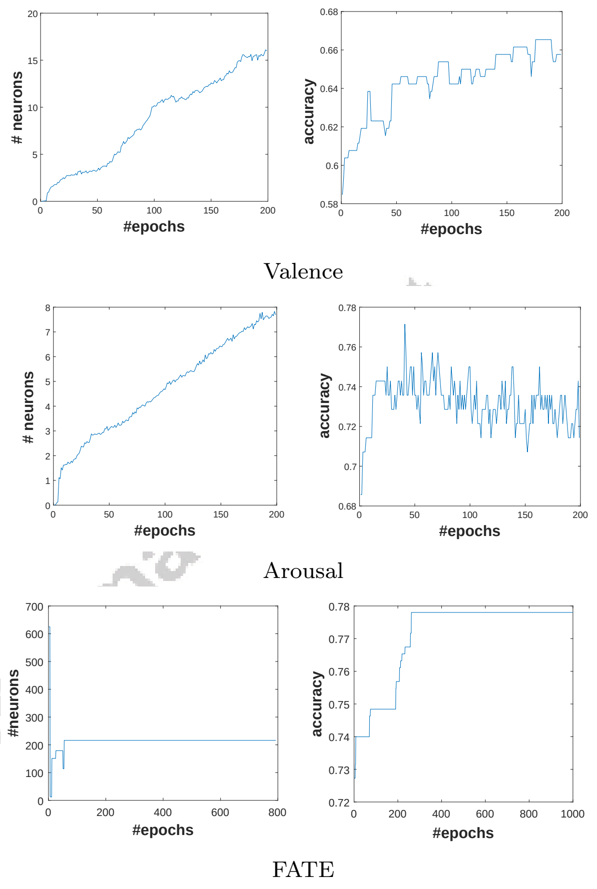
[1] http://github.com/cogncomp/fate

1 (unpleasant/stressed) to 0 (happy/calm). We consider 2 leads and up to 500 samples from each lead and binary valence labels. The Amigos database contains ECG recordings from 40 subjects and 16 movie clips. Dreamer has recordings of 23 subjects and 18 movie clips. Lastly, Ascertain with 58 student samples for 36 movies. Figure 5 illustrates a sample ECG where the valence and arousal are both high. Each heart-beat spans R-R peaks shown in pink circles. The shorter peaks correspond to Q and S peaks. When viewing movie clips for different emotions such as ‘Sad’ or ‘Fear’, the amplitude and frequency of the wave form will be different. Table 1 compares the classification accuracy of the proposed FATE with the baseline GP without multi-tasking and NeuroEvolution of Augmenting Topologies (NEAT) algorithm. FATE is able to outperform both the methods with up-to 15% in accuracy.

Table 1 Comparison of different baselines with proposed model on different types of Dataset

Dataset	GP	NEAT [11]	FATE
ECG	74.2	71.5	**84.6**
Chatbot	60.0	59	**68.4**

Fig. 6 We compare the evolution of number of neurons and accuracy for NEAT and the proposed FATE. The first figure is the increase in number of hidden neurons and the second figure is the corresponding improvement in accuracy.



5.2 Childrens Conversation Dataset

The Paradis corpus consists of sample conversations with 25 children who are learning English as a second language [30]. Transcription is in English orthography only; phonetic transcription was not included in this research. Any real names of people or places in the transcripts have been replaced with pseudonyms. When the study started, the children were, on average, 5;6 years old with a mean of 9.5 months of exposure to their English L2 in a preschool or school program. Data was collected approximately every 6 months for 5 rounds. We can convert the problem into a classification problem where the response generated by the model can be labeled as ‘adult’ or ‘child’. We consider two tasks for children with five months and 34 months of exposure to English. Here we compare the response for two questions ‘When’s your birthday?’ and ‘What’s your favourite food?’ for both task. We can see that at five months the child uses support sounds such as ‘mmhm’ and children with a longer exposure need fewer prompts from the adult. In order to evaluate the chatbot we consider the recursive replies to an input question. Here each response is used as the input in the next time stamp (see Table 2). The response can be classified into 10 categories based on the conversation input question such as ‘Food’ or ‘Birthday’. Table 1 compares the classification accuracy of the proposed FATE with the baseline GP without multi-tasking and NEAT algorithm. FATE is able to outperform both the methods with up-to 10% in accuracy.

5.3 Neuroevolution Parameters

In order to determine the optimal hyper-parameters we create an initial population of neural networks for each of the types described in Figure 3. Table 3 provides the list of tunable parameters for both the types of neural networks and the range of values considered in the experiments. For the case of feedforward neural networks we vary the number of neurons, the number of epochs, the learning rate, the regularization constant and the activation function. For the case of Seq2seq we vary the batch size, the number of hidden layers, the number of attention units, the number of heads in the multi-attention model and the number of epochs. Fig-

Table 2 We illustrate samples generated conversations between a child and an adult using Seq2seq for 5 months and 34 months exposure to English

5 months
Question : How old are you?
Reply (t1) : six
Reply (t2) : youre six
Reply (t3) : thats pretty old
Reply (t4) : and when it was my birthday
Reply (t5) : I was five
34 months
Question : do you go to school ?
Reply (t1) : yes
Reply (t2) : what grade are you in
Reply (t3) : kindergarten
Reply (t4) : youre in kindergarten still
Reply (t5) : mmhm

ure 6 shows the evolution of the proposed FATE and the baseline NEAT algorithm on ECG dataset. We had to run NEAT for both tasks Valence and Arousal. The neural network predicted by NEAT has only up-to 20 hidden neurons instead by using an ensemble of neural networks the proposed FATE uses up-to 260 neurons. Lastly, the proposed FATE is exponentially faster than the baseline NEAT algorithm.

Table 3 Tuning of Hyper-parameters

Feedforward NN	Parameters
Number of Neurons	[1, 2,. . . , 10]
Number of Epochs	[1, 2,. . . , 50]
Learning Rate	[0.1, 0.2,. . . , 1]
Regularization Constant	[0.1, 0.2,. . . , 1]
Activation Function	['trainbr' 'trainbfg' 'trainlm']
Seq2Seq	**Parameters**
Batch Size	[1, 2, . . . , 5]
Number of Layers	[1, 2, 3]
Number of Neurons	[1, 2, . . . , 5]
Number of Heads	[1, 2, . . . , 5]
Number of Epochs	[50, 100, . . . , 500]

5.4 Decision Tree Representation

In this paper, each genetic program takes the form of a decision tree as it is causal, transparent and intuitive. In a decision tree each operator is a boolean test that compares a numeric attribute against a threshold value or a nominal attribute against a set of possible values. For example in Figure 7 we show the predicted GP tree for the two ECG tasks : Valence and Arousal. The optimal trees for both tasks have three neural topologies. Valence is an aggregate of P12, P77 and P40 and Arousal is an aggregate of P14, P12 and P21. Hence, we can conclude that some components such as P12 are reused in both tasks. To solve the tree we have to first solve the +ve and -ve sub-tree and the result is propagated from the leaf nodes upwards to the root node. For example in the tree for Valence, if the condition $\sqrt[3]{P40} > 0.78$ is true then the solution is +ve child value 0 and if the condition is false then the solution is the -ve child value that is 1. Next, we go up one level, here again if the condition $P77 < 0.49$ is true then the solution is the value from +ve child that is $P12$ and if the condition is false then the solution is from the -ve child that is $P40$. The class label in this task is binary where 0 corresponds to Sad and 1 corresponds to Joy.

In Figure 1 we illustrate two sample chromosomes that both have a higher accuracy on task 1. After cross over we swap the sub-trees (shaded) between the two parents. The resulting chromosomes may have higher skill on task1 compared to both parents. However, it may happen that an offspring is created that has a higher accuracy in task 2. This is also observed in Figure 7. Here, we consider the two tasks : Valence and Arousal in ECG data. We can see that individual P12 occurs in both tasks optimal trees. This happens due to implicit genetic transfer between the two tasks.

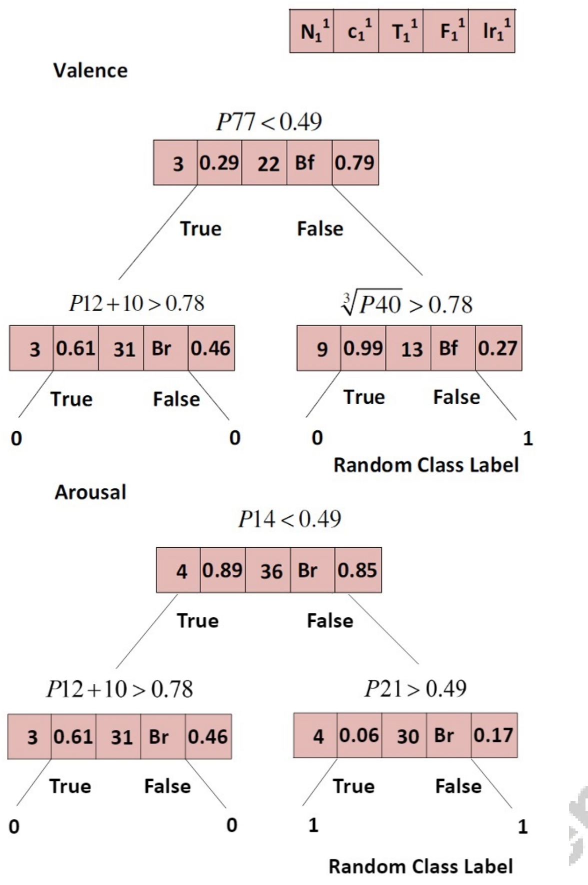
6 Conclusion

We show that evolutionary approaches can be used to optimize the hyper-parameters of complex neural architectures. Furthermore, we can use assortative mating to evolve several tasks in the same model. To enable explicit transfer between different tasks, we can use genetic programming where each individual has a tree like structure and crossover is possible by simply swapping sub-trees. To determine the optimal neural topology, we train several different models in parallel. Next, we use Fuzzy logic to determine the inter-task covariance matrix and transform the features from both tasks to a common space. The transformed data is used to train a genetic program classifier. The elite genetic program is an ensemble of two or more topologies that are combined together using mathematical functions. We evaluate our model on two real world problems and outperform baselines by over 10% in accuracy.

Compliance with Ethical Standards : This work is internally funded by the College of Science and Engineering at Jame Cook University. This work is also partially supported by the Data Science and Artificial Intelligence Center (DSAIR) at the Nanyang Technological University.

Conflict of Interest : Iti Chaturvedi declares that she has no conflict of interest. Chit Lin Su declares that she has no conflict of interest. Roy Welsch declares that he has no conflict of interest.

Fig. 7 Predicted Optimal Trees for Valence and Arousal using FATE.



Ethical approval : This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. E. Cambria and A. Hussain, "Sentic album: Content-, concept-, and context-based online personal photo management system," *Cognitive Computation*, vol. 4, pp. 477–496, 2012.
2. E. Cambria, T. Mazzocco, A. Hussain, and C. Eckl, "Sentic medoids: Organizing affective common sense knowledge in a multi-dimensional vector space," in *ISNN*, 2011, p. 601–610.
3. E. Cambria, A. Hussain, C. Havasi, and C. Eckl, "Sentic computing: Exploitation of common sense for the development of emotion-sensitive systems," in *Development of Multimodal Interfaces: Active Listening and Synchrony*, 2010, pp. 148–156.
4. I. Chaturvedi, Y. S. Ong, and R. V. Arumugam, "Deep transfer learning for classification of time-delayed gaussian networks," *Signal Processing*, vol. 110, pp. 250–262, 2015.
5. B. Da, A. Gupta, and Y. S. Ong, "Curbing negative influences online for seamless transfer evolutionary optimization," *IEEE Trans. Cybernetics*, vol. 49, no. 12, pp. 4365–4378, 2019.
6. T. He, Y. Liu, T. H. Ko, K. C. C. Chan, and Y. S. Ong, "Contextual correlation preserving multiview featured graph clustering," *IEEE Transactions on Cybernetics*, pp. 1–14, 2019.
7. J. Zhong, L. Feng, W. Cai, and Y. S. Ong, "Multifactorial genetic programming for symbolic regression problems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–14, 2018.
8. H. Li, Y. S. Ong, M. Gong, and Z. Wang, "Evolutionary multitasking sparse reconstruction: Framework and case study," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 733–747, 2019.
9. L. Oneto, F. Bisio, E. Cambria, and D. Anguita, "Semi-supervised learning for affective common-sense reasoning," *Cognitive Computation*, vol. 9, pp. 18–42, 2016.
10. N. Majumder, S. Poria, H. Peng, N. Chhaya, E. Cambria, and A. Gelbukh, "Sentiment and sarcasm classification with multitask learning," *IEEE Intelligent Systems*, vol. 34, pp. 38–43, 2019.
11. R. Miikkulainen, *Neuroevolution*. New York: Springer, 2010.
12. K. O. Stanley, B. D. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the nero video game," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 6, pp. 653–668, 2005.
13. Y. Li, Q. Pan, T. Yang, S. Wang, J. Tang, and E. Cambria, "Learning word representations for sentiment analysis," *Cognitive Computation*, vol. 9, pp. 843–851, 2017.
14. E. V. Bonilla, K. M. Chai, and C. Williams, "Multi-task gaussian process prediction," in *NIPS*, J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds., 2008, pp. 153–160.
15. Y. S. Ong and A. Gupta, "Evolutionary multitasking: A computer science view of cognitive multitasking," *Cognitive Computation*, vol. 8, no. 2, pp. 125–142, 2016.
16. L. Feng, L. Zhou, J. Zhong, A. Gupta, Y. S. Ong, K. Tan, and A. K. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE Transactions on Cybernetics*, vol. 49, no. 9, pp. 3457–3470, 2019.
17. J. Snoek, H. Larochelle, and R. P. Adams, in *NIPS*, 2012, pp. 2960–2968.
18. N. Richards, D. E. Moriarty, and R. Miikkulainen, "Evolving neural networks to play go," *Applied Intelligence*, vol. 8, no. 1, pp. 85–96, Jan 1998.
19. M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *IJCAI*, 7 2018, pp. 5369–5373.
20. X. Yao and M. M. Islam, "Evolving artificial neural network ensembles," *IEEE Computational Intelligence Magazine*, vol. 3, no. 1, pp. 31–42, 2008.
21. E. Fernandes, A. C. De Carvalho, and X. Yao, "Ensemble of classifiers based on multiobjective genetic sampling for imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2019.
22. I. Chaturvedi, R. Satapathy, S. Cavallari, and E. Cambria, "Fuzzy commonsense reasoning for multimodal sentiment analysis," *Pattern Recognition Letters*, vol. 125, pp. 264 – 270, 2019.
23. H. Chung, M. Iorga, J. Voas, and S. Lee, ""Alexa, can i trust you?"," *Computer*, vol. 50, no. 9, pp. 100–104, 2017.
24. M. Ghazvininejad, C. Brockett, M.-W. Chang, B. Dolan, J. Gao, W. tau Yih, and M. Galley, in *AAAI*, 2018, pp. 5110–5117.

25. C.-Y. Chen, D. Yu, W. Wen, Y. M. Yang, J. Zhang, M. Zhou, K. Jesse, A. Chau, A. Bhowmick, S. Iyer, G. Sreenivasulu, R. Cheng, A. Bhandare, and Z. Yu, "Gunrock: Building a human-like social bot by leveraging large scale real user data," 2018.
26. T. Young, E. Cambria, I. Chaturvedi, M. Huang, H. Zhou, and S. Biswas, "Augmenting end-to-end dialog systems with commonsense knowledge," in *AAAI*, 2018, pp. 4970–4977.
27. S. Katsigiannis and N. Ramzan, "Dreamer: A database for emotion recognition through eeg and ecg signals from wireless low-cost off-the-shelf devices," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 1, pp. 98–107, 2018.
28. J. A. Miranda Correa, M. K. Abadi, N. Sebe, and I. Patras, "Amigos: A dataset for affect, personality and mood research on individuals and groups," *IEEE Transactions on Affective Computing*, pp. 1–1, 2018.
29. R. Subramanian, J. Wache, M. K. Abadi, R. L. Vieriu, S. Winkler, and N. Sebe, "Ascertain: Emotion and personality recognition using commercial sensors," *IEEE Transactions on Affective Computing*, vol. 9, no. 2, pp. 147–160, 2018.
30. H. Golberg, J. Paradis, and M. Crago, "Lexical acquisition over time in minority first language children learning english as a second language." *Applied Psycholinguistics*, vol. 29, no. 1, pp. 41 – 65, 2008.