National University of Computer & Emerging Sciences (FAST-NU)

Project Name

Classification of Actionable Genetic Mutations using Nature Inspired Algorithmic Approach

Project Supervisor
Mr. Muhammad Shahzad

Project Team

Ahmed Raza	**16K-3986**
Sanif Ali Momin	**16K-3966**
Sohail Zia	**16K-3974**

Submitted in the partial fulfilment of the requirements for the degree of Bachelor of Science

Submitted on: June 07, 2020

Department of Computer Science National University of Computer & Emerging Sciences (FAST-NU) Main Campus, Karachi

June 2020

National University of Computer & Emerging Sciences (FAST-NUCES)

Project Supervisor	**Mr .Muhammad Shahzad**
Project Manager	**Ahmed Raza (16K-3986)**
Project Team	**Sanif Ali Momin (16K-3966), Sohail Zia (16K-3974)**
Submission Date	**June 07,2020**

Supervisor

Mr. Muhammad Shahzad _____

Head of Department

Dr. Muhammad Atif Tahir _____

Department of Computer Science National University of Computer & Emerging Sciences (FAST-NU) Main Campus, Karachi

Classification of Actionable Genetic Mutations using Nature Inspired Algorithmic Approach

Ahmed Raza
Computer Science
National University of Computer and Emerging Sciences
Karachi, Pakistan
k163984@nu.edu.pk

Sanif Ali Momin
Computer Science
National University of Computer and Emerging Sciences
Karachi, Pakistan
k163966@nu.edu.pk

Sohail Zia
Computer Science
National University of Computer and Emerging Sciences
Karachi, Pakistan
k163974@nu.edu.pk

***Abstract*—In this paper, we propose a PSO based neural network for a challenging classification problem. This classification problem was posed as a kaggle competition named *Personalized Medicine: Redefining Cancer Treatment* in which the task was to classify genetic mutations using expert annotated knowledge base containing text literatures. In order to investigate the efficacy of using PSO for training neural networks especially on extremely challenging problems, we conducted this research. We first used conventional approaches to tackle the problem, and so we used a GRU model for this problem. Then we used our proposed method to train the GRU model with the same configurations (i.e number of layers and neurons). These models were evaluated on the competition's test set and therefore the final results were compared relative to kaggle's public and private scores. The scores for the models were very similar. The results also show that initially the PSO based GRU model rapidly converged to good solutions faster than our conventional GRU model, but then got stuck at those solutions for a while. The particles then were able to find better solutions but with an extremely slower rate. This shows the potential of using PSO for training neural networks. Hence, we can claim that it's worthwhile to use PSO for the training of neural networks.**

I. Introduction

In recent years, the interest in incorporating nature inspired algorithms in neural networks has been growing rapidly. Amongst a long list of nature inspired meta-heuristics in literature, the most common ones are Genetic Algorithm (GA), which is a search method based on the abstraction of Darwinian evolution and natural selection of biological systems and Particle Swarm Optimization (PSO) which is based on swarm behavior in nature, such as fish and bird schooling.

For the training of neural networks, backpropagation is the most widely used method which uses gradient descent, a local search algorithm. Although, gradient descent and some of its variants suffer from two common problems, slow convergence rate and getting trapped in local minima due to its local search nature, current success of gradient descent based back propagation trained neural networks has proven the critics wrong as also mentioned in [20] that the latter case is infact a rare problem.

Similarly, back propagation algorithm is also used in training of recurrent neural networks (RNN). However, due to recursive loops in its structure, the gradient computations tend to make RNN more computationally expensive. Moreover, because of the feedback loops in RNN and due to its weight parameters being shared throughout the time steps, the search space is of higher dimension and often has a large set of parameters for the RNN to work on. This makes the network quite sluggish in attaining convergence. Apart from gradual convergence rate, its more prone to getting stuck in saddle points especially in extremely high dimenstions, as mentioned before. As NIA works great at exploring a large and complex search space, therefore this is one of the reasons why incorporating them into neural networks is still an open research problem. Many researchers have applied and studied the performance of meta-heuristic algorithms for the training of neural networks and reported that the meta-heuristic approaches outperform all the conventional methods by a huge margin [1-18], but it is often the case that they use a standard benchmark dataset to evaluate their model on. As recent developments in the field of deep learning aim to solve challenging problems namely speech recognition, face recognition, text summarizing, sentiment analysis etc, it's not sagacious to claim that using nature inspired metaheuristics for training neural networks is practical because NIA incorporated neural networks haven't been evaluated on arduous and challenging problems sets which is exactly the aim of our work.

However, there is a different school of thought, who argue that incorporating NIA for training neural networks is not beneficial and that researchers should aim to solve other related issues, for example, neural network hyperparameter optimization. Experience tells us that backpropagation's convergence is somewhat sensitive to the hyperparameters of neural networks whose settings vary from problem to problem, so there also has been some solid research work on this problem. Therefore, automated hyperparameter tuning using NIA is another approach that can be taken into consideration related to our problem set.

II. Related Work

One of the early contributions to learn the weights of a feed forward neural network using Nature Inspired Metaheuristic Algorithm was done by Whitley [1], in which he used Ge-

netic Algorithm on the exclusive-or (XOR) problem, the 424 encoder problem, the minimal 2-bit adder problem, and fully connected 2-bit adder problem, achieving the convergence rate above 90%. Many researchers propagated this notion with added improvements namely connectivity optimization [2], architecture optimization[3-6] and even hyperparameter optimization [7] [8]. In [9], the Firefly algorithm was incorporated into backpropagation algorithm and tested on standard datasets (Iris, Wine and Liver datasets) , showed that the proposed method demanded far less number of iterations for the optimization algorithm to converge, thus improving convergence rate even with minimal neural network design. Similarly, Nazri Mohd. Nawi et al. [10] proposed a method which incorporated Cuckoo search into backpropagation with extremely good results, but the dataset that they used to assess the performance of their proposed method was XOR and OR datasets. They compared the performance of their proposed method with 3 algorithms, Artificial Bee Colony - Levenberg Marquardt (ABC-LM), Artificial Bee Colony - Backpropagation (ABC-BP) and standard Backpropagation, and showed that their proposed method achieved faster convergence rate and better accuracy, but the problem with ABC is that they are poor to exploitation. More recently, Whale optimization algorithm is applied to adjust the weight of ANN for smart grid cyber intrusion detection [11]. The model is trained on Mississippi State University and Oak Ridge National Laboratory databases of power-system attacks which itself consists of the 3 levels of difficulties, namely, Binary-class, Triple-class and Multi-class. The results of the experiment showed that the accuracy gradually increased with number of iterations, and at 64th iteration the accuracy stabilized at 99%.

Furthermore, for the improvement of the training of LSTMs, 4 different metaheuristic optimizers are used [12]. These are: Harmony Search (HS), Gray Wolf Optimizer (GWO), Sine Cosine (SCA) and Ant Lion Optimization Algorithms (ALOA). The proposed work is applied on standard datasets: Breast Cancer Wisconsin Dataset (BCWDS) and Epileptic Seizure Recognition Dataset (ESRDS). The experimentation suggested that, after keeping the parameter of the search algorithms (population size and iteration number) constant, and by using different number of neurons, the models trained on BCWDS produced an average accuracy of 86.67% and the models trained on ESRDS produced an average accuracy of 77.14%. Similarly, Andre Quintiliano Bezerra Silva [13] trained the LSTM network on cervical cancer dataset using five different nature inspired metaheuristic algorithms, Genetic Algorithm (GA), Gravitational Search (GS), Particle Swarm Optimization (PSO), Gray Wolf Optimizer (GWO) and Cuckoo Search (CS). The results showed that LSTM with PSO performed better than LSTM with other algorithms, achieving an accuracy rate of 96% after testing out different population sizes.In [14], a modified PSO in training of backpropagation algorithm of RNN is used for the identification of frequency-dependent impedances of power electronic systems. It was used in the sense that instead of only choosing "gbest" value based on MSE of particles, the gradient of each particle was also taken into consideration. The results obtained from modified PSO were then used for weight initialization in RNN, and the performance of this hybrid approach gave significantly better results than individual BP and PSO. However, this approach did not guarantee to achieve global minima during the backpropagation step. Moreover, Rashid A.Tarik et. al. [15] proposed a modified GWO in training of multi hidden RNNs for forecasting students' outcomes. The modification to the GWO algorithm was such that, instead of considering three sets for best solutions, Alpha, Beta and Delta, another best solution Gamma is also involved, which are now used by Omega wolves to update their positions.The results showed that the proposed method achieved an accuracy of 98% while other models, such as GWO-based RNN without any modifications achieved 94%, whereas BPNN achieved only 76%. Additionally, Tae-Young Kim et. al. [16] proposed a PSO-based CNN-LSTM neural network for the prediction of energy consumption. They also used different models like RandomForest, Linear Regression, Decision Tree, Multi-layer perceptrons and manually adjusted CNN-LSTM, the results showed that their proposed method yielded least MSE among all models.

There also have been some research work on hyperparameter optimization. Steven R. Young et. al. [17] performed hyperparameter optimization for automating network selection on computational clusters by genetic algorithms. They evaluated their method on CIFAR-10 dataset, and the results found were the best hyperparameters for the model. B. Qolomany et. al. [18] used PSO for hyperparameter optimization of deep learning model on dataset collected from campus Wifi network. They compared their method with grid search and their obtained results show that the training time decreased by 77% - 85%. In [8], WOA is applied with CNN for texture classification on two different levels. Firstly, in convolutional layer, to optimize the values of filters, and then in fully-connected layer to optimize the weights and biases. The proposed method is applied to Kylberg v1.0 , Brodatz and Outex_TC_00012 datasets, and gave better accuracy than the existing methods.

Over the two decades, researchers have transformed their general perspective of the application of Nature Inspired Algorithms to being more problem-specific, so as to deal with every possible room for improvements in order to get better model performances. Similarly, our proposed method incorporates PSO in training of neural networks, specifically a GRU model on a Kaggle dataset where the problem is to classify genetic mutations from clinical literature [19].

III. Proposed Work

A. PSO

PSO is a nature inspired evolutionary and stochastic optimization technique that was proposed by Ebenhart and was inspired by the collective behavior of animals that live in large colonies such as birds, fishes etc. and therefore it is originally based on the concept of swarm intelligence. In

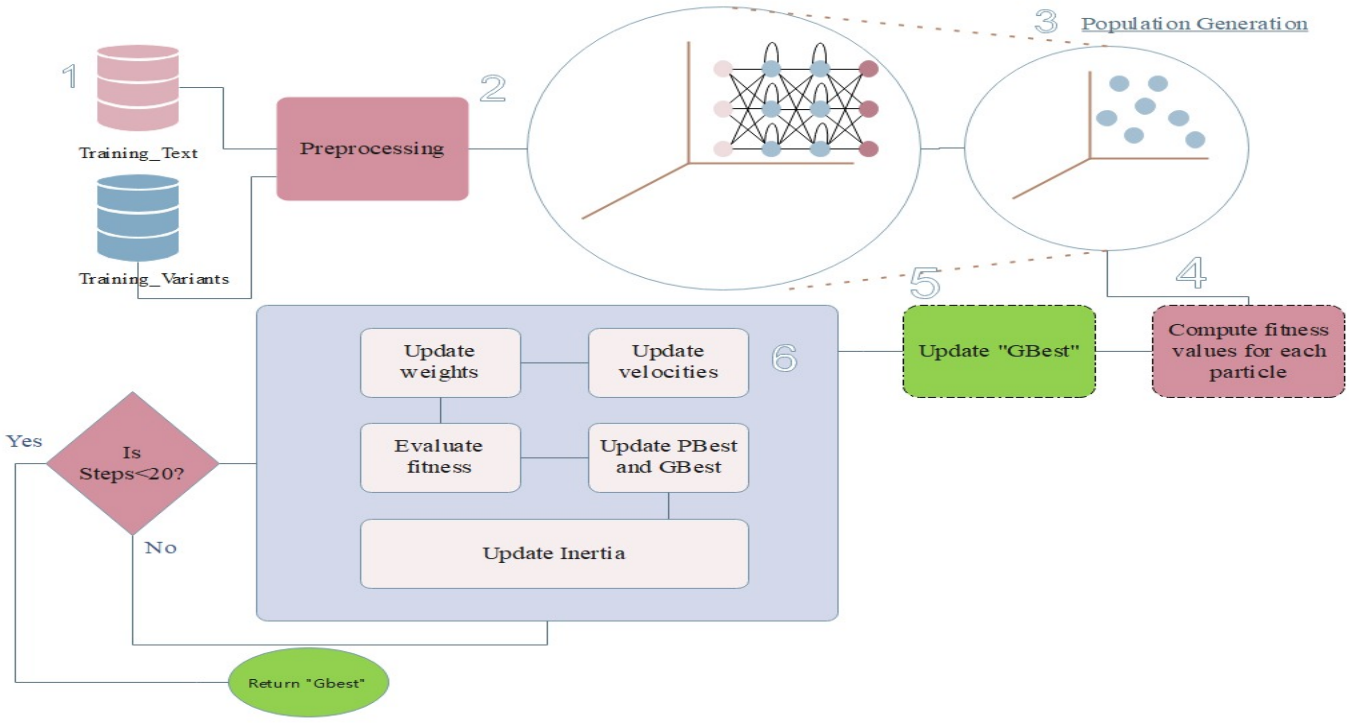

Fig. 1. PSO based Neural Network

PSO, a swarm of particles communicate with one another using search directions. Each particle is a possible candidate solution. These particles in the swarm traverse the search space to locate a global optimum. During each iteration, each particle updates its position according to its previous history (local best position) and also its neighbor's history (global best position). More formally, a particle is composed of 3 vectors:

- x-vector: records the current position vector of the particle in the search space.
- p-best-vector: records the position vector of the best solution found so far by this particular particle.
- v-vector: contains the direction to which particle will travel in.

In each iteration, the particle updates its position using the equation (1).

$$x_i = x_i + v_{i+1} \qquad (1)$$

$$v_{i+1} = w_i + C_1.R_1.(p_i^{best} - x) + C_2.R_2.(p_i^{global} - x) \qquad (2)$$

where v_{i+1} is the updated velocity. The p_i^{best} is the best individual position and p_i^{global} is the best position found by the i^{th} particle so far. The C_1 and C_2 are the acceleration coefficients which are empirically recommended to be set to 2. w is inertia weight with value recommended within 0.4 to 0.9 range. R_1 and R_2 are two random numbers in the range [0,1].

Once a particle computes the updated position, it then evaluates the solution using a fitness function, and in the case of machine learning problems, it can be as simple as log loss or MSE. However, as these metaheuristics are gradient free, we are not bound to use differentiable loss functions.

The pseudocode for PSO is provided in the Figure 2.

Algorithm 1 Particle Swarm Optimization Algorithm

Input: $Problem_{size}, Population_{size}$
Output: P_{g_best}

$Population \leftarrow 0$
$P_{g\_best} \leftarrow 0$
for $i = l$ **to** $Population_{size}$ **do**
    $P_{velocity} \leftarrow RandomVelocity()$
    $P_{position} \leftarrow RandomPosition(Population_{size})$
    $P_{p\_best} \leftarrow P_{position}$
    if $(Cost(P_{p\_best}) \leq Cost(P_{g\_best}))$ **then**
        $P_{g\_best} \leftarrow P_{p\_best}$
while $(!StopCondition())$ **do**
    for $(P \in Population)$ **do**
        $P_{velocity} \leftarrow UpdateVelocity(P_{velocity}, P_{g\_best}, P_{p\_best})$
        $P_{position} \leftarrow UpdatePosition(P_{position}, P_{velocity})$
        if $(Cost(P_{p\_best}) \leq Cost(P_{g\_best}))$ **then**
            $P_{g\_best} \leftarrow P_{position}$
            if $(Cost(P_{p\_best}) \leq Cost(P_{g\_best}))$ **then**
                $P_{g\_best} \leftarrow P_{p\_best}$
return P_{g_best}

Fig. 2. Pseudocode for particle swarm optimization

Algorithm 2 PSO-GRU

Def $Optimizer()$:

$model \leftarrow Model.Build()$
$structure \leftarrow model.to\_json()$
for i **in** $num\_particles$ **do**
    $Particles[i] \leftarrow keras.model.model\_from\_json(structure)$
    $Particle[i].compile()$
for each $particle$ **do**
    $Get\ local_{score}$
    if $local_{score} < Global_{score}$) **then**
        $Update\ Global_{score}$
for i **in** $iter$ **do**
    $Update\ inertia()$
    for each $Batch$ **in** $batches$ **do**
        for each $particle$ **in** $particles$ **do**
            $particle.UpdateVelocities()$
            $particle.UpdateWeights()$
            $Evaluate\ loss\ as\ local_{score}$
            if $local_{score} < Global_{score}$) **then**
                $Update\ Global_{score}$
                $Global_{best\_Particle} \leftarrow particle$
return $Global_{best\_Particle}$

Fig. 3. Pseudocode for our proposed strategy

B. PSO based neural network

Our proposed PSO strategy works as follows: Each particle in the swarm is treated as a weight matrix (or a model with different initialized weights). Therefore, each particle is itself a whole model with different weights. The weights are initialized normally. During training, the model's training error acts as a fitness function for each particle. Specifically, 'categorical cross-entropy' is used as the metric. Iteratively, these particles, during training, would record their personal best p_i^{best} and also, the swarm would record the p_i^{global}. The p_i^{global} would hold the best obtained weights till i^{th} iteration. To update the position of the particles, we used another PSO variant that was proposed by [21]. We tried the standard PSO, and also other old variants, but they either took very long to train, or simply did not work. The equation for this variant uses an additional formula that linearly decreases the inertia weight during training. The formula is:

$$w = (w_s - w_e)(t_{max} - t)/t_{max} + w_e \qquad (3)$$

where w_s and w_e are the initial and final values of inertia respectively, t_{max} represents the maximum number of iterations and t denotes the current iteration. The values of w_s and w_e are 0.9 and 0.4 respectively. Similarly, all other particles then evaluate their position (weight solution) and then update their position using the equations (1).
The figure 1. describes the holistic view of our proposed strategy and Figure 3. provides the pseudocode of our proposed work.

IV. Experimental Setup

A. Data

For our research, the experimental dataset was obtained from a Kaggle competition named, *Personalized Medicine: A treatment for Cancer*, which contains text-based clinical literature, and manually labeled genes and its variations. . The challenge was to classify genetic mutations based on the literature.
This dataset has 3342 training samples, distributed among 9 classes, which were predefined by the organizer. After importing the data, the first step was to merge the files into a single data frame, for this task and other similar data manipulation operations we used Python's Pandas library. The dataframe then contained 3 features(text, gene and variation) and one label. Next, we inspect the data, by first visualizing the class distribution. As shown in Figure 3 , the distribution is severely imbalanced.

B. Preprocessing

The next phase is to refine the textual features. For this, NLTK, Spacy and scikitlearn libraries were used. We used regular expressions to remove bibliography and unrecognizable tokens from text, then we tokenized each text into tokens.
For each token, if it appears in our set of stopwords formed from the union of stopwords sets taken from aforementioned libraries' stopwords, and the words that we found not useful through inspection, we removed those words. Further, we used Spacy to do part of speech tagging in order to lemmatize these words into their root words. In this way, the document having a maximum words count of 76708 was reduced to 47557. Now, to treat the gene and variants, we use scikit learn, to label encode each of them, later we use SVD to truncate these encoded vectors to lower dimensions.
In our last step of preprocessing, we use Doc2Vec proposed by Le and Mikolov et al. [24] to represent the textual features to document vectors of dimension 200, and concatenate it with the truncated gene and variation vector.

C. Model

We used our proposed PSO-GRU. We first created neural network based on 3 layers of Gated recurrent units(GRUs), implemented using Keras library. The input layer dimension is set to 250, as the dimension of our feature matrix. The next two hidden layers have successive input dimensions of 250 and 200, while the output layer at end uses a Dense layer, and output dimension of 9, since it is a multiclass problem. At the output layer, we use Softmax as an activation function, to predict the output probability of classes for each feature set input. For this case, Categorical cross-entropy is used as the loss function.
Now we compiled the model, the configurations of the model is summarized in Table I. The model's initial weights are then used to initialize 80 particles with the same model structure.

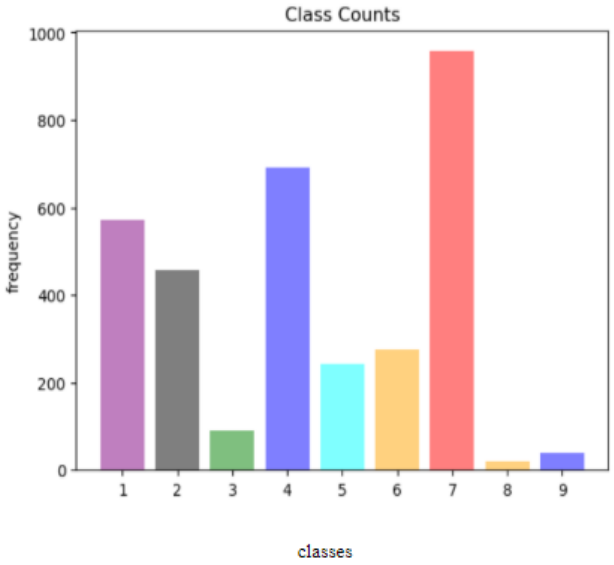

Fig. 4. *Class distribution*

TABLE I
GRU MODEL SUMMARY

Layer(s)	*Units*	*Parameters*
GRU_1	250	375750
GRU_2	250	375750
GRU_3	250	375750
Flatten	200	0
Dense	9	1809
	Total Parameters: 1,023,909	

These particles are then scattered over the search space around our initial particle to represent a swarm. The initial value of inertia, $local_{acc}$, and $global_{acc}$ are set to 0.7929, 1.49445, 1.49445 successively. We train each model i.e. particle, using $batch_{size}$ of 500, and number of iterations to 20. To deal with the class imbalance problem, we used Synthetic Minority Oversampling Technique (SMOTE), and for validation we used stratified k-folds, while keeping the number of folds to 3.

V. RESULTS

In the previous section, we discussed the dataset and model structure. To evaluate the performance of our proposed work, we first trained a GRU model using the conventional methods on this problem-set to be able to perform a comparative analysis between the GRU and the PSO based GRU model. Hence, this GRU model was used as a baseline and then PSO was also used to train this baseline model. These models were evaluated on kaggle's official evaluation metric, that uses cross entropy given by equation (4).

$$-\sum_{c=1}^{M} y_{o,c} log(p_o, c) \qquad (4)$$

The summary of kaggle scores for both models are given in II

TABLE II
KAGGLE SCORES SUMMARY

Model(s)	*Kaggle Private Score*	*Kaggle Public Score*
GRU without PSO	4.37	1.48
GRU with PSO	2.32	2.48

In addition to Kaggle's evaluation metrics, we visualized our model's performances using confusion matrix. Figure 5 and 6 shows the confusion matrix and classification report respectively, of PSO based GRU model,

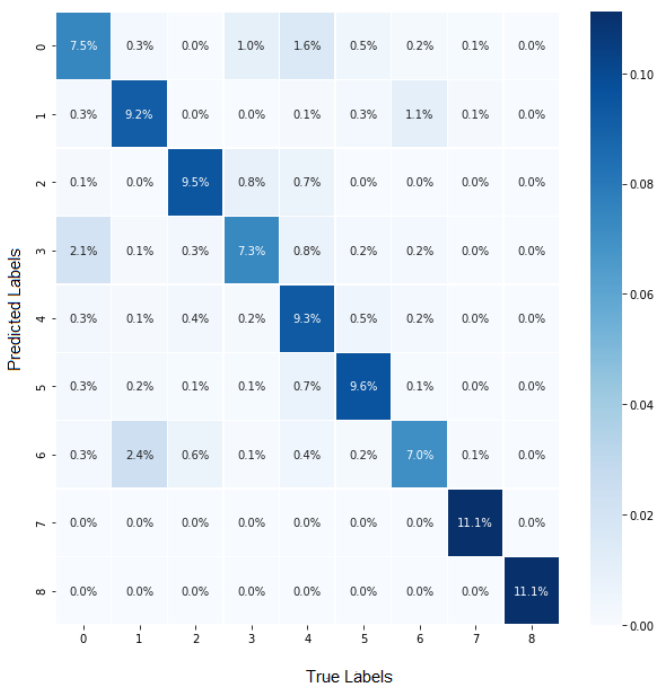

Fig. 5. *Confusion Matrix for PSO based GRU*

Furthermore, we compare the loss on each successive iteration for both models as shown in Figure 7 and 8 , which clearly shows that though the particles were randomly initialized and were far from optimum solutions, they rapidly converged at a faster rate than GRU in less than 100 generations. The particles then got somewhat trapped at local solution i.e. local optimium solutions, but due to the PSO variant that we used which linearly decreases the inertia weight, the particles were able to find better solutions but with an extremely slower rate. This clearly shows the potential of using PSO for training neural networks. These results can further be improved by experimenting with the different approaches mentioned in the future work section.

Classification Report of Fold: 3

	precision	recall	f1-score	support
class 0	0.69	0.67	0.68	319
class 1	0.74	0.82	0.78	319
class 2	0.86	0.85	0.86	319
class 3	0.77	0.66	0.71	319
class 4	0.68	0.84	0.75	319
class 5	0.85	0.87	0.86	319
class 6	0.79	0.63	0.70	319
class 7	0.97	1.00	0.99	319
class 8	0.99	1.00	1.00	319
accuracy			0.82	2871
macro avg	0.82	0.82	0.81	2871
weighted avg	0.82	0.82	0.81	2871

Fig. 6. *Classification Report for PSO based GRU*

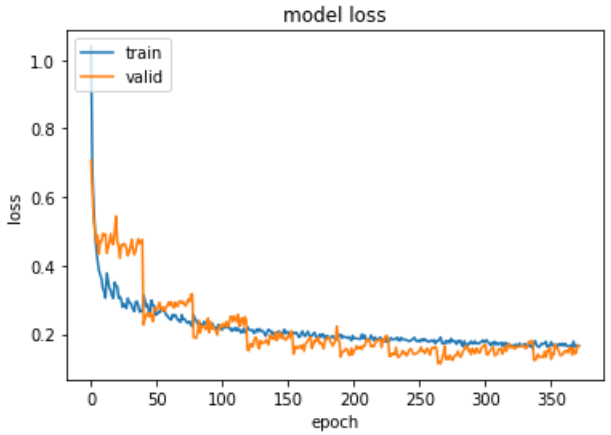

Fig. 7. *Model loss*

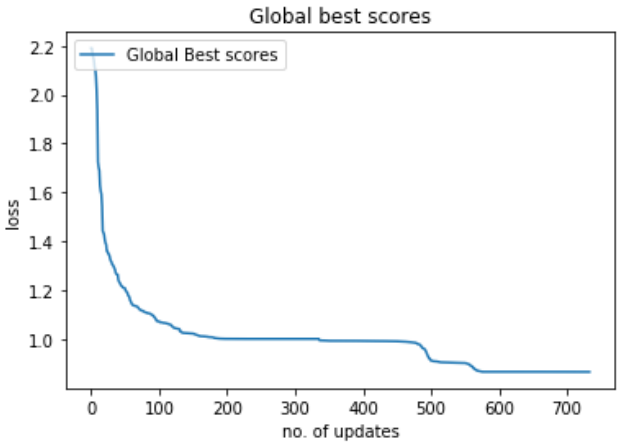

Fig. 8. *Plot for number of updated of Global best position*

VI. Discussion of Results

As PSO is not usually applied in extremely high dimensional problems i.e. dimensional space of parameters greater than 10,000 , therefore using it in neural networks is even more challenging. The baseline model that we used for classification was parameterized by more than a million parameters. To be able to achieve better performance with PSO, the parameters associated with PSO also need to be decided optimally. One parameter of PSO is the population size (number of particles). Though, choosing this parameter is often a trial and error procedure and has no systematic way of deciding it, yet there does exist a common heuristic amongst PSO practitioners that is to set the population size three times the dimension of your problem. But, alas, we could not work with a very large population size due to hardware limitations. Despite this, we still were able to achieve somewhat considerable performance. Apart from this, due to time constraints, we could not implement a parallelized version of PSO to be able to run a PSO based model on GPU. Therefore, the PSO based neural network took 6 hours to train while GRU took 1.5 hours as it was trained on GPU. The takeaway from the results is that, although the GRU based model has achieved slightly better performance, the public and private score difference is quite big while the public and private score difference for PSO based model is significantly quite low.

VII. Future Work

Although PSO based neural network performed relatively better than our baseline models in terms of kaggle submission scores, however the results empirically did not live up to our expectations. Our purpose of the research was to test if incorporating PSO may yield faster convergence than the orthodox methods that are predominantly used to train neural networks. However, we cannot claim that using PSO to train neural networks is not worthwhile because unfortunately there are a number of ideas that have been left unexplored due to limited time capacity. We will further continue with our research on using those alternative ideas specifically on our problem set and especially under controlled settings. We recommend others to try out these mentioned approaches as well. We will briefly discuss those approaches in this section.

- One approach is to use the PSO based optimized weights for initialization in GD based neural network. Purpose is to use PSO to do global search, and to further optimize those obtained solutions by using local search algorithms GD, ADAM etc. This approach is usually known as the Memetic Approach in literature, a paradigm of hybrid algorithms,which gives a methodological concept for combining two or more metaheuristics/algorithms (global and local) which efficiently explores the whole search space.

- Another approach that has not widely been used is known as the ‘Lamarckian Evolution’ approach which works in a complementary way to the Memetic approach. In this approach the weights after learning using GD.etc,

are kept in the population for PSO.

- We have only used the available corpus provided in the competition for learning word vector representation. However, there is medical literature data, e.g. PubMed, that is available on the internet, and therefore it would be worthwhile to use those large corpus to learn better word embeddings.

- In the literature, there have been numerous variants of PSO proposed in order to improve PSO's performance as it has been applied in many different applications. For example, regarding the concerns of PSO parameter choice, [22] comparatively summarized over 15 proposed strategies to set inertia weight. Also, because of PSO's rapid convergence, it still is prone to getting trapped in local optimum especially in high dimensional problems where there are many local solutions, [23] proposed a PSO swarm formulation method to overcome this drawback, where the swarms are formed according to the distance between the particles. Therefore, there are a plethora of variants that can be experimented with.

VIII. Conclusion

In this paper, we proposed a PSO based neural network for a challenging classification problem. This classification problem was posed as a kaggle competition named *Personalized Medicine: Redefining Cancer Treatment* in which the task was to classify genetic mutations using expert annotated knowledge base containing text literatures. In order to investigate the efficacy of using PSO for training neural networks especially on extremely challenging problems, we conducted this research. We first used conventional approaches to tackle the problem, and so we used a GRU model for this problem. Then we used our proposed method to train the GRU model with the same configurations (i.e number of layers and neurons). These models were evaluated on the competition's test set and therefore the final results were compared relative to kaggle's public and private scores. The scores for the models were very similar. There's still a lot of experiments to be carried out to provide a more detailed comparative analysis, e.g. trying out other PSO variants, playing with the population size, implementing a parallelized version of PSO etc. However, our provisional experiment did show that PSO is a possible candidate algorithm for training neural networks.

References

[1] D. Whitley, T. Starkweather, and C. Bogart, *Genetic algorithms and neural networks: optimizing connections and connectivity* , Parallel Comput., vol. 14, no. 3, pp. 347 – 361, Aug 1990.

[2] J. R. McDonnell and D. Waagen, *Determining network connectivity using evolutionary programming*, 25th Asilomar Conf. on Signals, Sys., and Computers, Monterey, CA, 1992.

[3] J. Yu, L. Xi, S. Wang, *An improved particle swarm optimization for evolving feedforward artificial neural networks*, Neural. Process. Lett. 26 (3) (2007) 217–231.

[4] J. Yu, S. Wang, L. Xi, *Evolving artificial neural networks using an improved PSO and DPSO*, Neurocomputing 71 (4–6) (2008) 1054–1060.

[5] H. Quan, D. Srinivasan, A. Khosravi, *Particle swarm optimization for construction of neural network-based prediction intervals*, Neurocomputing 127 (2014) 172–180.

[6] N.S. Jaddi, S. Abdullah, A.R. Hamdan, *Optimization of neural network model using modified bat-inspired algorithm*, Appl. Soft. Comput. J. 37 (2015) 71–86.

[7] N.S. Jaddi, S. Abdullah, A.R. Hamdan, *A solution representation of genetic algorithm for neural network weights and structure*, Inf. Process. Lett. 116 (1) (2016) 22–25.

[8] U.Dixit, A.Mishra, A.Shukla, R.Tiwari, *Texture classifcation using convolutional neural network optimized with whale optimization algorithm*, SN Applied Sciences (2019)

[9] S. Nandy, P. P. Sarkar, and A. Das, *Analysis of a nature inspired firefly algorithm based back-propagation neural network training* , Int. J. Comput. Appl., vol. 43, no. 2, pp. 8 – 16, Apr 2012.

[10] A. Khan, N. M. Nawi, and M. Z. Rehman, *A new back-propagation neural network optimized with cuckoo search algorithm* , in Computational Science and Its Applications—ICCSA 2013: Proceedings of the 13th International Conference, Ho Chi Minh City, Vietnam, June 24-27, 2013, Part I, vol. 7971 of Lecture Notes in Computer Science, pp. 413–426, Springer, Berlin, Germany, 2013.

[11] Haghnegahdar, L., and Wang, Y. (2019). *A whale optimization algorithm trained artificial neural network for smart grid cyber intrusion detection*. Neural Computing and Applications.

[12] Tarik A. Rashid, Polla Fattah, Delan K. Awla, *Using Accuracy Measure for Improving the Training of LSTM with Metaheuristic Algorithms*, Procedia Computer Science, Volume 140, 2018, Pages 324-333

[13] Andre Quintiliano Bezerra Silva: *Predicting Cervical Cancer with Metaheuristic Optimizers for Training LSTM*. ICCS (5) 2019: 642-655

[14] P. Xiao, G. K. Venayagamoorthy, K. A. Corzine, *Combined training of recurrent neural networks with particle swarm optimization and backpropagation algorithms for impedance identification*, 2007 IEEE Swarm Intelligence Symposium, pp. 9-15, 2007.

[15] Rashid A. Tarik, Abbas and K. Turel, 2019, *A multi hidden recurrent neural network with a modified grey wolf optimizer*, PLOS ONE.

[16] Kim, T.-Y., and Cho, S.-B. (2019). *Particle Swarm Optimization-based CNN-LSTM Networks for Forecasting Energy Consumption*. 2019 IEEE Congress on Evolutionary Computation (CEC).

[17] Steven R. Young , Derek C. Rose , Thomas P. Karnowski , Seung-Hwan Lim , Robert M. Patton, *Optimizing deep learning hyper-parameters through an evolutionary algorithm* , Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments, p.1-5, November 15-15, 2015.

[18] B. Qolomany, M. Maabreh, A. Al-Fuqaha, A. Gupta, and D. Benhaddou, *Parameters optimization of deep learning models using particle swarm optimization* In Int. Wireless Communications and Mobile Computing Conference, pp. 1285-1290, 2017.

[19] https://www.kaggle.com/c/msk-redefining-cancer-treatment

[20] LeCun, Y., Bengio, Y., Hinton, G, *Deep Learning*, Nature 521, 436 (2015)

[21] J. Xin, G. Chen, and Y. Hui, *A particle swarm optimizer with multistage linearly-decreasing inertia weight* in Proc. Int. Joint Conf. Comput. Sci. Optim. (CSO), vol. 1. Sanya, China, 2009, pp. 505–508.

[22] J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon and A. Abraham, *Inertia Weight strategies in Particle Swarm Optimization*, 2011 Third World Congress on Nature and Biologically Inspired Computing, Salamanca, 2011, pp. 633-640, doi: 10.1109/NaBIC.2011.6089659.

[23] J. Tsuji and M. Noto, *Distance based multiple swarms formulation method in particle swarm optimization*, 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest, 2016, pp. 001573-001578, doi: 10.1109/SMC.2016.7844463.

[24] Quoc Le and Tomas Mikolov. 2014. *Distributed representations of sentences and documents*, In Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 (ICML'14). JMLR.org, II–1188–II–1196.

NIA PAPER - PLAG REPORT

ORIGINALITY REPORT

12%	6%	10%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	2%
2	Tae-Young Kim, Sung-Bae Cho. "Particle Swarm Optimization-based CNN-LSTM Networks for Forecasting Energy Consumption", 2019 IEEE Congress on Evolutionary Computation (CEC), 2019 Publication	1%
3	Tarik A. Rashid, Polla Fattah, Delan K. Awla. "Using Accuracy Measure for Improving the Training of LSTM with Metaheuristic Algorithms", Procedia Computer Science, 2018 Publication	1%
4	link.springer.com Internet Source	1%
5	Ujjawal Dixit, Apoorva Mishra, Anupam Shukla, Ritu Tiwari. "Texture classification using convolutional neural network optimized with whale optimization algorithm", SN Applied Sciences, 2019	1%

Publication

6	www.naturalgenesis.net Internet Source	1%
7	Submitted to Higher Education Commission Pakistan Student Paper	<1%
8	Marini, Federico, and Beata Walczak. "Particle swarm optimization (PSO). A tutorial", Chemometrics and Intelligent Laboratory Systems, 2015. Publication	<1%
9	Andre Quintiliano Bezerra Silva. "Chapter 62 Predicting Cervical Cancer with Metaheuristic Optimizers for Training LSTM", Springer Science and Business Media LLC, 2019 Publication	<1%
10	Yang, Xin-She. "Optimization and Metaheuristic Algorithms in Engineering", Metaheuristics in Water Geotechnical and Transport Engineering, 2013. Publication	<1%
11	Jianbo Yu. "An Improved Particle Swarm Optimization for Evolving Feedforward Artificial Neural Networks", Neural Processing Letters, 10/23/2007 Publication	<1%

12	S. K. Goudos. "Dielectric filter optimal design suitable for microwave communications by using multiobjective evolutionary algorithms", Microwave and Optical Technology Letters, 10/2007 Publication	<1%
13	T Mahmood, S O Ahmed, M H Swaleh, S H Nayyer. "A-Eye: Automating the Role of the Third Umpire in the Game of Cricket", 2011 International Conference on Information Science and Applications, 2011 Publication	<1%
14	Studies in Fuzziness and Soft Computing, 2016. Publication	<1%
15	Lida Haghnegahdar, Yong Wang. "A whale optimization algorithm-trained artificial neural network for smart grid cyber intrusion detection", Neural Computing and Applications, 2019 Publication	<1%
16	Nazri Mohd Nawi, Abdullah Khan, M.Z. Rehman. "A New Levenberg Marquardt based Back Propagation Algorithm Trained with Cuckoo Search", Procedia Technology, 2013 Publication	<1%
17	D Whitley, T Starkweather, C Bogart. "Genetic algorithms and neural networks: optimizing connections and connectivity", Parallel	<1%

Computing, 1990
Publication

18	"Proceedings of the 16th International Conference on Hybrid Intelligent Systems (HIS 2016)", Springer Science and Business Media LLC, 2017 Publication	<1%
19	Michal Pluhacek, Roman Senkerik, Donald Davendra. "Chaos particle swarm optimization with Eensemble of chaotic systems", Swarm and Evolutionary Computation, 2015 Publication	<1%
20	Peng Xiao, Ganesh K. Venayagamoorthy, Keith A. Corzine. "Combined Training of Recurrent Neural Networks with Particle Swarm Optimization and Backpropagation Algorithms for Impedance Identification", 2007 IEEE Swarm Intelligence Symposium, 2007 Publication	<1%
21	www.hindawi.com Internet Source	<1%
22	Submitted to Nottingham Trent University Student Paper	<1%
23	media.neliti.com Internet Source	<1%
24	Submitted to University of Sheffield	

	Student Paper	<1%
25	Submitted to The University of Manchester Student Paper	<1%
26	Submitted to University of Florida Student Paper	<1%
27	doc.lagout.org Internet Source	<1%
28	Submitted to University of Birmingham Student Paper	<1%
29	Jinn-Moon Yang, Cheng-Yan Kao. "A Robust Evolutionary Algorithm for Training Neural Networks", Neural Computing & Applications, 2001 Publication	<1%
30	Khalid M. Salama, Ashraf M. Abdelbar. "Learning neural network structures with ant colony algorithms", Swarm Intelligence, 2015 Publication	<1%
31	Amr M. Ibrahim, Noha H. El-Amary. "Particle Swarm Optimization trained recurrent neural network for voltage instability prediction", Journal of Electrical Systems and Information Technology, 2017 Publication	<1%

32	"Computational Science and Its Applications – ICCSA 2013", Springer Science and Business Media LLC, 2013 Publication	<1%
33	Submitted to University of New England Student Paper	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		