Evolutionary Data Synthesis for Scarce Data Depression Classification

Lanqin Yuan

Research School of Computer Science
The Australian National University
Canberra, Australia
lanqin.yuan@anu.edu.au

Abstract. Humans when observing individuals suffering from depression have been identified to exhibit physiological reactions that can be captured and analysed to classify the severity of depression of the observed individuals. However, the difficulty and high cost of gathering a sufficiently large training dataset, as well as the noisy nature of human physiological data can make it hard for classification models to learn effectively. In this work, we attempt to address data scarcity by synthesising training data using an evolutionary algorithm based algorithm, using a partially trained neural network model to compute the fitness metric. We further train the neural network model on synthesised data and apply network reduction techniques to remove inhibiting and poor performing hidden units via a Contributions scoring metric. Our results do not show statistically significant improvement and fall short of the results described in existing literature. We conclude that the evolutionary algorithm and network pruning is overall ineffective at improving classifier performance for scarce and noisy data.

Keywords: Neural Networks · Data Synthesis · Network Reduction · Timeseries Summary Statistics · Depression

1 Introduction

Depression or Major Depressive Disorder (MDD) is a common mental health disorder that affects more than 216 million people of all ages [11]. Individuals suffering from MDD tend to exhibit a persistent depressed mood, a general loss of interest in activities, low self esteem, and in severe cases suicidal thoughts. Recent works in depression recognition and classification show promising results and suggest that humans subconsciously sense depression when observing depressed individuals and observers of depressed individuals subconsciously respond with observable physiological reactions [10]. By capturing these reactions, we can identify and classify depression in the observed individuals at an accuracy higher than human performance [6] [12].

However, the difficulty and high cost of obtaining such data, as well as the inherent noisy nature of gathering human physiological data can limit the amount of quality data available for training which in turn can limit the classification performance of classification models as well as cause problems for the deployment of such technology in the real world at a reasonable cost. Existing works such as [4] suggest the synthesis of data to address such problems by increase the training set using data points generated by generative machine learning models such as Restricted Boltzmann machines [8] and Variational Autoencoders [1]. In this work, we take a different direction to the aforementioned works and explore the use of an Evolutionary Algorithm based data synthesis model to address data scarcity by generating new training data while using a trained model as our fitness function. We further explore *Contribution Scoring*, a metric identified in [2] as a means to further improve our model by removing poor performing hidden units.

2 Dataset

The dataset described in [12] was used in this work. The dataset is collected from observations of 12 participants with no prior knowledge of depression identification viewing a set of 16 recorded videos of the behaviours of depressed and non-depressed individuals. The set of videos featured 4 individuals with no depression, 4 with mild depression, 4 with moderate depression, and 4 with severe depression. The individuals in the videos communicated in German, a language which none of the observing participants understood, and were asked to answer various questions. All observers were exposed to the same set of videos, each observing participant contributing 12 observations with 4 observations of level of depression. This resulted in a total of 192 observations. The participants Galvanic Skin Response (GSR), Pupillary Dilation (PD), and Skin Temperature (ST) data were recorded as time series data while watching these videos. Feature extraction was done after filtering and normalising the data to extract summary statistic features from this data. In total, 85 numeric features were extracted from the GSR, PD and ST data. Each data instance was also associated with a label which was the depression level of the individual in the video.

3 Method

The classification model used in this work can be separated into two components: a Neural Network Classifier, and a Evolutionary Algorithm based generator. Construction of the model can can be separated into three stages:

1. Training the classifier on the original training data.
2. Synthesising new training data using a simple genetic algorithm based generator using the classifier's performance on the generated data as the fitness function.
3. Further Training the classifier using the synthesised training data in addition to the original training data.

Network Reduction in the form of Contribution pruning was also explored as a means to improve classifier performance by reducing the number of hidden units in the Neural Network Classifier.

3.1 Neural Network Architecture

The choice of architecture for the neural network classifier was heavily based on that used in [12] due to the high reported classification accuracy. A simple two layer Neural Network classification model was initially used, featuring a single hidden layer of 50 units with a sigmoid activation function and an output layer consisting of 4 units with a Softmax activation function, each corresponding to a category of depression (none, mild, moderate, severe). The architecture is illustrated in figure 3.1. After a thorough exploration of the hyperparameters, the results of which are discussed in section 5, the final number of hidden units was increased to 100. Implementation of the model was done in Pytorch [5].

The dataset used for training was the previously discussed dataset from [12]. To process the dataset, each observation's GSR, PD, and ST features were normalised and concatenated together to form a single vector of dimension $\mathbb{R}^{85}$. This was further aggregated into a data matrix $\mathbf{X} \in \mathbb{R}^{192\times185}$, where each row corresponds to a single observation instance. All features were considered for classification following what was done in [12] for which the classifier also considered all 85 features. As the problem is expected to not be fully convex in nature, mini-batch training was utilised in order to address the potential problem of slow convergence due to repeated becoming stuck in local minima. Training was done by feeding a mini-batch of data instances with all 85 features into the network, then adjusting the network parameters using back propagation with the Cross-Entropy loss function and the Adam stochastic optimisation algorithm [3].

Other network architectures for the neural network classifier were also explored, such as the utilisation of more hidden layers (from 2 to 4), the introduction of a dropout layer, using different activation functions, and the introduction of weight decay. Overall, all these architectures produced either similar results with higher costs, or worse results compared to the simple two layer architecture. The results of these experiments are discussed in further detail in section 5.

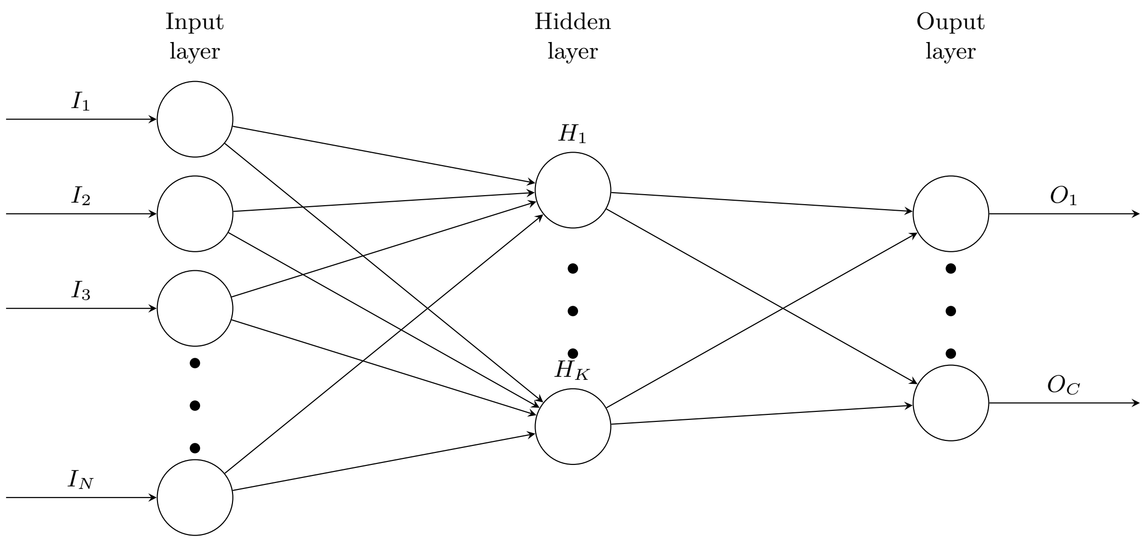

Fig. 1. The Neural Network architecture used in this work. Each input was fed into a hidden layer with a Sigmoid activation function, then fed into the output units with Softmax activations. In this work 85 features were used ($N = 85$) with 100 hidden units ($K = 100$) to classify 4 levels of depression ($C = 4$).

3.2 Evolutionary Data Synthesis

To address the problem of data scarcity, an Evolutionary Algorithm based generator is used to synthesise new training data that attempt to be similar to extant training data. A diagram of the steps of the Evolutionary Algorithm generator is shown in Figure 3.2.

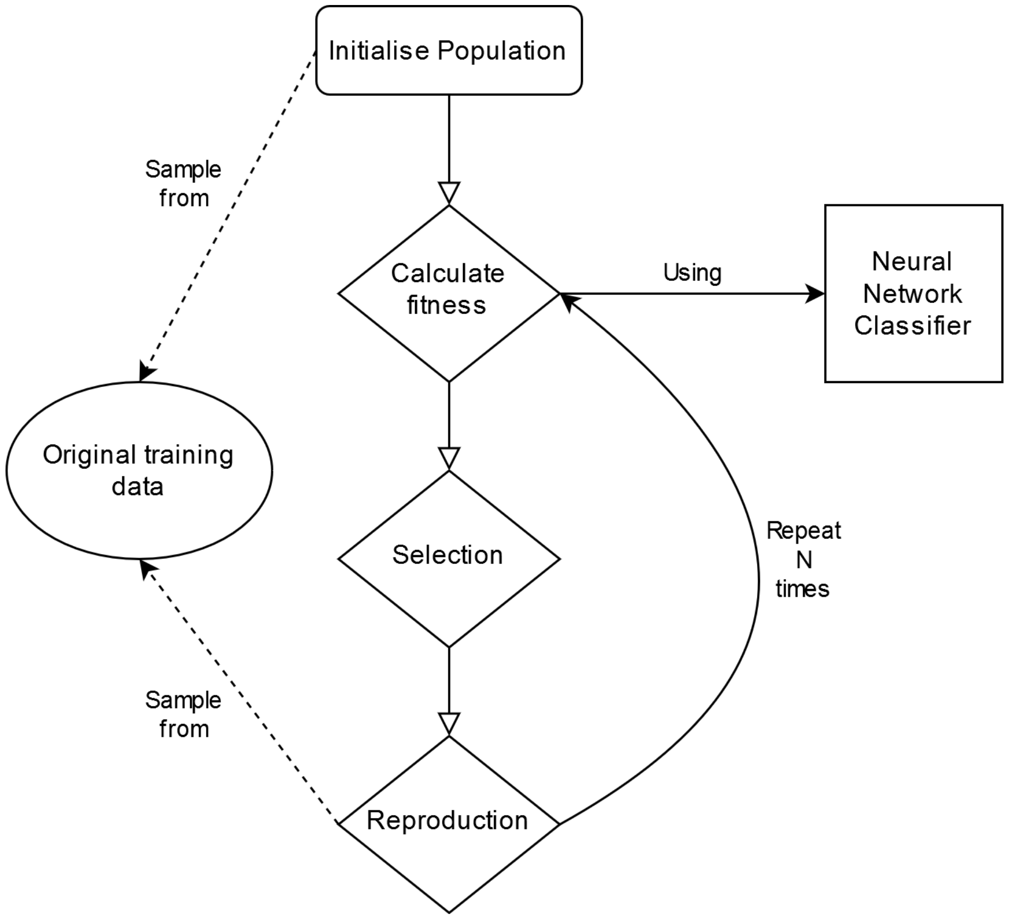

Fig. 2. Diagram of the Evolutionary Algorithm based generator model.

We construct a separate generator for each class label. For a generator of class label l, we initialise our population by randomly sampling n many existing data pairs from our training data that are of class l (with replacement) to act as parents for our initial population, where n is a user set parameter determining the number of generated training data points for each class. The Uniform crossover operator is then applied on each pair with equal probability to select from both parents to obtain our initial population. The fitness of an individual is evaluated using the neural network classifier. The logit value of the individual's true class classification returned by the output layer of the neural network classifier for the individual is used as the individual's fitness score. The raw logit value is used over the softmax probabilities to give greater distinction between individuals as the logit scores are not bound between 0 and 1.

Selection is done using ranked based sampling selection [9]. 50 % of the individuals in old population are kept on in the new population. These kept individuals are sampled at random from the old population with the probability for each individual being selected being weighted based on their fitness. This probability is computed by ranking each individual using their fitness in descending order, and assigning a probability based on this ranking to favour individuals with high fitness. Formally the probability of an individual $u_i \in U$, where U is the set of all individuals in the old population is defined as:

$$P(u_i) = \frac{||U|| - r(u_i) - 1}{\sum_{u \in U} r(u)} \tag{1}$$

where $r(u_i)$ is the rank of the individual u_i sorted descendingly based on their fitness starting at 1, with 1 being the most fit individual. The remaining 50% of the new population is created using the reproduction operator with the first parent being sampled from the old population using the same rank based sampling method and probabilities as the selection process, and the second parent being sampled from the original training data points that are of class l.

The reproduction operation between two individuals consists of Uniform Crossover between the two individuals with mutation. Each feature in the new individual is selected with equal probability from either parent. Uniform crossover was chosen to maintain high genetic diversity and variation among candidate individuals, as well as retain some level of noise as do not wish to overfit our generated data to the partially trained Neural Network model. Mutation consists of adjusting a single feature by $\pm 0.1 * f$ with f being the mutation factor between 0 and 1, with equal probability for the sign to be positive or negative. This adjustment is bounded between 0 and 1 and cannot move the feature value outside this range. Mutation is applied to the child with p probability after all features are selected, where p is a hyperparameter which determines the probability of a mutation occurring. As all features are normalised, the mutation operator applied on an individual essentially corresponds to changing one of the individual's feature's value by 10% of the mutation factor.

The evolutionary algorithm is run for i iterations and the final population is taken as the generated training dataset. Once each class generator has generated a dataset, the datasets are then concatenated to the original training dataset and used to further train the model. Reseting the model completely and training from scratch using the generated + original training data was also explored but was found to give worse results than further training the pretrained model. The final hyperparameters for the evolutionary algorithm are as follows: the number of iterations to run the evolutionary algorithm i, the mutation factor f, the mutation probability p, and n the number of generated data points per class. A wide range of these hyperparameter values are explored in this work and this is further discussed in section 5.

3.3 Network Reduction

Network reduction via pruning was explored as a means as a means to reduce model complexity as well as to attempt to prune counter productive units. Techniques and metrics identified in [2] discuss how networks can be simplified by removing unneeded or poorly performing hidden units from the model. Identification of these hidden units to prune can be done using various metrics such as relevance, sensitivity, distinctiveness, and badness. In this work, pruning was done after fully training the model and before evaluation. In the case of the Evolutionary Algorithm model, pruning was done after the first training stage then reverted before further training. The pruned model was not used as the fitness function due to its overall unstable results. We focus on a particular technique in *Contribution* to reduce our network and attempt to improve model performance.

Contribution Pruning Contribution is a relatively simple scoring metric which can be easily calculated. Informally the contribution of a hidden unit is defined as the sum of the product of the activation of the unit with the weight of each output unit, multiplied by a sign correction scalar depending on whether the unit's contribution to the final output moved the final output towards the correct prediction. Formally it is defined for a single input instance as the following:

$$Contrib(\boldsymbol{x}_i, t_i) = \sum_{c=1}^{C} \sum_{k=1}^{K} s_j(t_i)\ z_k(\boldsymbol{x}_i)\ w_{ck} \tag{2}$$

where: $\boldsymbol{x}_i$ is an input vector of dimension $\mathbf{R}^m$ where m is the number of features representing a single input instance (in this case $m = 85$) and t_i the associated associated target, w_{ck} is the weight associated with the c-th output unit and the k-th hidden unit, $z_k(\boldsymbol{x}_i)$ is the logit of the k-th hidden unit given input $\boldsymbol{x}_i$, s_j is the sign correction scalar given target t_i defined as positive one if the j-th output unit corresponds to the value of the target t_i, and negative one otherwise. Formally s_j is defined as:

$$s_j(t_i) = \begin{cases} +1, j = t_i \\ -1, otherwise \end{cases} \tag{3}$$

The total contribution score for each is be defined as the sum over all instances in the training dataset.

After the computation of the contribution score, the bottom $n\%$ (a pruning size hyperparameter) units with the lowest score were pruned by setting the weights connecting the units to the output layer to 0, effectively removing the unit from the network. Pruning sizes of 10,20,25,30, and 50 percent of all hidden units were explored in this work.

4 Evaluation

For evaluation, a process similar to that described in [12] was followed. Information identifying the participant was not included in the datasets and no features explicitly contained any information regarding which participant the observation came from. Due to the nature of having multiple data instances from each participant, a random split for training and test datasets was not viable due to the potential for bias to be introduced if a participant's data was in both the training and test sets. As such, a leave-one-participant-out k-fold validation scheme, where all data from one participant was omitted from training and utilised for testing while all others were utilised for training, was used for evaluation. This extended to the evolutionary algorithm based generator which only sampled from the training set.

Classification accuracy of the level of depression was the main evaluation measure considered in this work. Two accuracy measures were initially considered: classification accuracy with regards to all 4 levels of depression, and classification accuracy with regards to having depression and not having depression. Both measures were computed using the same model by calculating directly from the confusion matrix. It was quickly identified that the classification performance w.r.t having depression vs not having depression was extremely sub par, with results well below the expected result from random guessing (<25%). As a result, this metric was abandoned in favour of only focusing on classification accuracy w.r.t all 4 levels of depression.

Hyperparameters were selected after experimentation based on results over all participants. The neural network hyperparameters selected are as follows: learning rate = 0.0001, mini-batch size = 30, number of epochs = 500, prune percentage = 10%. These hyperparameter values are used in all results discussed in this work unless otherwise stated. The evolutionary algorithm parameters used for the data generator are as follows: number of iterations = 50, mutation factor = 0.25, mutation probability = 0.75, and number of generated data points per class = 48, effectively doubling the amount of training data.

Evaluations consisted of the averaged results of 10 experiments conducted with the same hyperparameters. Each experiment consisted of a 12-fold validation over all participants, training the neural network model for 500 epochs, generating new training data using the evolutionary algorithm based model, then further training the neural network model for 250 epochs. Evaluation of network reduction's effects on model performance was done after the completion of the first and second training stages. The Cross-Entropy Loss and classification accuracy was recorded for the neural network model with and without network reduction after training only on the original data for 500 epochs, and after further training for 500 epochs on the original data and the generated data. Network reduction did not affect training as the weights were reverted back to their pre-pruned values after obtaining results and before further training. The recorded results were then averaged over all 12 participants to produce the experiment results, which are further averaged over 10 experiments to produce the final results reported.

5 Results and Discussion

Classifier performance both with and without using generated training data was overall underwhelming, with test classification accuracies for individual participants ranging between 10-50%, and the overall mean accuracy across all participants hovering around 25-30%. Detailed results are shown in tables 1 and 2. We were not able to reproduce the results described in [12] in any way, achieving well below the reported 88% accuracy in all cases.

5.1 Baseline

Different architectures and hyper parameters for the Neural Network classifier were explored in this work in order to establish a stable baseline classifier from which we can improve upon. This exploration was conducted using only the Neural Network classifier without the use of the Evolutionary Algorithm based generator as we must first find a stable classifier configuration to ensure the fitness function for the generator is representative of the true feature distribution. We also wished to use the baseline as a point of comparison to observe the effects of further training using synthesised data versus training only on the original training data.

Initially, attempts were made to reproduce the results described in [12] to act as the baseline for improvement. [12] states that a 88% classification accuracy was achieved using a simple neural network architecture of a 2 hidden layers with 50 hidden neurons. A similar neural network model was constructed with 50 hidden neurons and sigmoid activations. Initial testing with learning rates of 0.01, and a max epoch of 100 gave a very poor result of sub 25% average classification accuracy, which is worse than that of the expected performance from random guessing given

Participant	Baseline	With EA
p02	23.75%	10.00%
p03	25.00%	27.50%
p04	13.75%	15.00%
p06	41.25%	37.50%
p07	25.00%	25.00%
p08	50.00%	47.50%
p09	37.50%	37.50%
p10	58.75%	60.00%
p11	31.75%	31.25%
p12	10.00%	6.25%
p13	22.50%	25.00%
p14	32.50%	36.25%

Table 1. Table showing mean accuracy over 10 experiments for all participants at final epoch. Baseline model was trained for 1000 epochs while the EA model was trained for 500 epochs, then further trained for 500 epochs after generating data using the EA generator.

the fact that the 4 classes in the dataset are balanced.

As such, a variety of classifier model architectures was explored. Increasing the number of hidden layers found to give with poor results, increasing training time while not affecting classifier performance. Examining the confusion matrix and average losses, it was found that the model was drastically overfitting to the training data. As such, Dropout was explored as a technique to introduce noise into the network and prevent overfitting. A three layer neural network featuring two hidden layers of size 100 and 80, with a dropout layer of 0.5 after the first hidden layer was found to give similar performance to the simple two layer neural network after training for 5000 epochs with a learning rate of 0.0001. However, due to the variance introduced from the stochastic nature of dropout, this results were not stable nor consistent, making the classifier unsuitable to be used as a fitness function. The three layer network was also significantly more complex and took noticeably longer to train without any real improvement. The leaky ReLU activation function was also explored, both as an activation function in before the second hidden layer after dropout, and as the activation before the output layer, both with no noteworthy results. Introducing weight decay to the Adam optimiser was also explored as a means to introduce noise to the network to prevent overfitting. Decay factors of 0.001, 0.01, and 0.1 were explored, all with classification results worse than the dropout model.

	Baseline	With EA
Final Mean Loss Training	1.209	1.174
Final Mean Loss Test	1.393	1.415
Final Mean Loss Test W/ Reduction	1.414	1.451
Final Mean Accuracy Training	50.62%	51.34%
Final Mean Accuracy Test	30.93%	29.90%
Final Mean Accuracy Test W/ Reduction	30.52%	32.40%

Table 2. Table showing mean accuracy and loss over all participants at final epoch. Baseline model was trained for 1000 epochs while the EA model was trained for 500 epochs, then further trained for 500 epochs after generating data using the EA generator. 10% network reduction is applied for the reduction results. Loss is calculated using the Cross Entropy loss function.

The final architecture which is used in the reported the results used 100 hidden neurons with a learning rate of 0.0001. This classifier was trained for 1000 epochs without the use of the evolutionary algorithm based generator to obtain our baseline results. Overall classifier accuracy was still very underwhelming, with accuracy generally around 30%, though still consistently better than naïve baselines such as the expected performance from random class selection (25%) and always selecting one class (25%). The baseline model's final mean test results over 10 experiments using the hyperparameters discussed in section 4 are shown in Figure 3 and Table 2.

5.2 Evolutionary Data Synthesis

Exploration of hyperparameters for the data generator was mainly focused on the number of generated data points per class. Generating 24,48, and 96 new training data points per class, which corresponds to increasing the test dataset by 50%, 100%, and 200% of its original size, was explored with the results displayed in Table 3.

Classifier	Accuracy	Loss
Baseline	30.93%	1.393
+24*4 data points	28.33%	1.380
+48*4 data points	29.90%	1.415
+96*4 data points	28.75%	1.377

Table 3. Table comparing final mean test accuracy and loss after training on different amounts of generated data. Baseline model was trained for 1000 epochs while the EA model was trained for 500 epochs, then further trained for 500 epochs after generating data using the EA generator. Loss is calculated using the Cross Entropy loss function.

The mean results for each individual participant is shown in Figure 1. We can see that overall accuracy were slightly below that of the baseline classifier, indicating. This suggests that the evolutionary algorithm was overall not effective at generating useful training data points. Other hyperparameters were also explored but were found to not have significant impact. Increasing the the number of iterations the evolutionary was run was found to decrease training loss, however this did not translate into improvements in testing accuracy. Increasing the mutation factor, mutation probability, and the number of features mutated per mutation were all found to simply introducing more noise to the classifier, giving more unstable classification results.

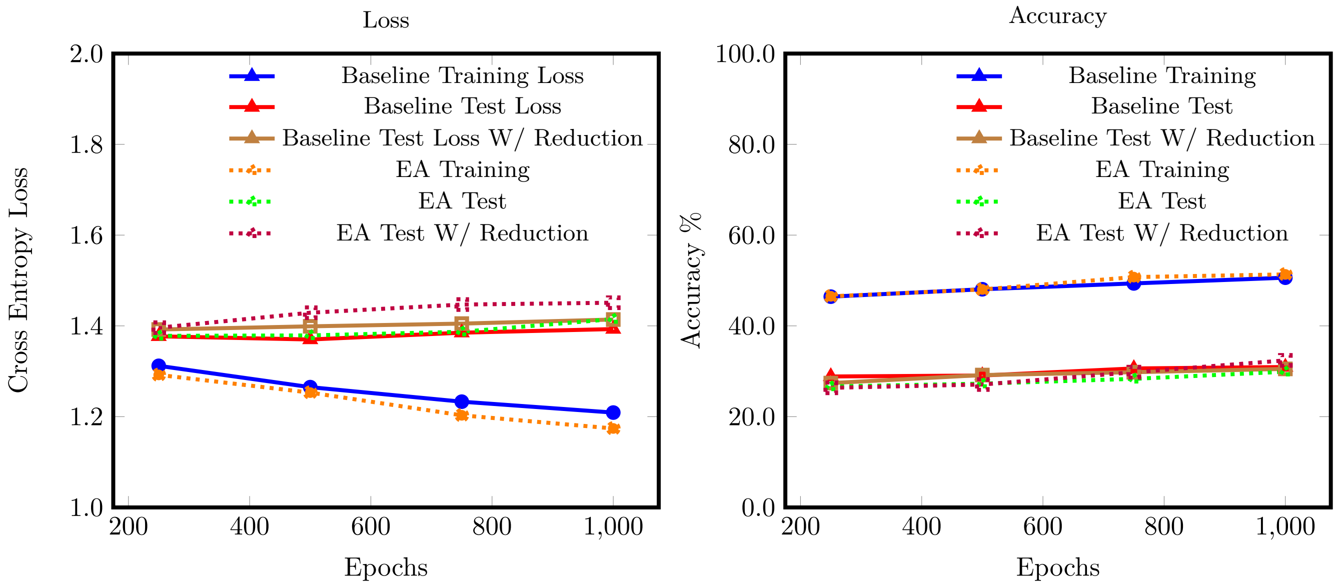

Fig. 3. Line graphs showing mean classification accuracy and loss over all participants at 250 epoch intervals for the Baseline model and the EA model with and without network reduction. Loss is calculated using the Cross Entropy loss function.

5.3 Network Reduction

Overall, network reduction via contribution pruning was found to not be effective in improving the model in any meaningful manner that was consistent and generally tended to introduce more noise to the model, giving worse test classification performance despite generally improving training classification accuracy. Pruning several different proportions of hidden units, from the bottom scoring 10% to 50% was explored with the results are shown in Table 4. All pruning experiments showed non-conducive results with regards to test classification accuracy compared to not pruning the network for the baseline model, and gave variable results for the evolutionary algorithm model. Pruning 10% of the network after training the model trained on the EA generated data was found to produce slightly better results than the baseline. However, this may simply be due to the introduction of additional noise to the classification model from both the generated data and network reduction increasing variance in results. A two sample t-test was conducted between the baseline model's results and the evolutionary algorithm + network reduction results, giving a p-value of 0.9272, indicating that the results are not statistically significant.

In general, we found that pruning any more than 20% of the hidden units would cause the model to classify only one class for most of the participants. Due to the already abysmal classification accuracy results, the accuracy drops were not significantly reflected when simply comparing the overall mean classification accuracy across all participants. However, upon inspection of the confusion matrices of each participant, it is obvious that pruning over 20% causes the model to show significant tendency to classifying one class for all data points regardless of the feature values. Pruning after completing 50% training then continuing to train using the pruned model was also explored with no improvements to results for both the baseline model and for the evolutionary algorithm model.

Reduction %	Baseline Loss	EA Loss	Baseline Accuracy	EA Accuracy
0%	1.393	1.415	30.93%	29.90 %
10%	1.414	1.451	30.52%	32.40 %
20%	1.417	1.467	29.61%	27.87 %
30%	1.462	1.501	30.87%	29.46 %
50%	1.456	1.521	28.98%	27.48 %

Table 4. Table showing final mean accuracy and loss when reducing the number of hidden neurons by different proportions on the test set for the baseline model and the EA model. Loss is calculated using the Cross Entropy loss function.

5.4 Discussion and Observations

A interesting observation was that certain participants produced significantly worse test results than others. Participants “p02”, “p04”, and “p12” results appear to be noticeably different from the other participants across all experiments, with the model performing far worse than the expected result from random guessing (25%). For participant “p07” the model has strong tendencies to classify all observations from this participant as a single class. Increasing the number of epochs, number of hidden units, number of hidden layers, adjusting the learning rate, or any other adjustments to hyper parameters and network architecture was not found to change this tendency unless the weights were so skewed that the model degenerates to erratic random assignment, suggesting that there may be a low degree of seperation between the classes for this particular participant. This could perhaps suggest that certain participants are outliers and have different physiological reactions when observing depressed individuals.

Another observation was the difference in peak accuracies across different epochs for different participants. As mean classification accuracy across all participants was the main focus in this work, the same hyperparameters were used across all participants for testing. However, different participants exhibited peak testing accuracy at different epochs and as such, early stopping [7] could be explored as a potential direction for future works.

We also notice that overall, the test loss has a positive gradient across all experiments despite the training loss always having a negative gradient across all experiments, which may indicate that the model itself is unable to generalise for this particular dataset.

6 Future Works

Several extensions can be made to the current model:

The use of generative neural network models such as the Variational Auto Encoder or Generative Adverserial Network in some manner to produce and/or mutate individuals may be considered as an potential improvement to the evolutionary algorithm based generator. Currently, the methods used for mutation are quite crude and in general unguided. As such, using a VAE or GAN for mutation may improve the quality of the generated training data.

As the problem of depression identification and depression level classification can be considered two separate problems, a potential direction for future works is to utilise two separate models: one detecting the presence of depression and another the level of depression. Classification would be done by first using the first model to classify if the individual has depression, and if they do, subsequently using the second model to classify the level of depression. By using two separate models, we simplify the individual problems which may allow for better training given the same amount of data. The evolutionary algorithm based generator can likewise be applied to the two problems separately, using one model to generate more data for the depression identification problem, and the other model to generate generated more data for the depression level classification problem.

As mentioned in Section 5, significant differences in performance were observed for certain participants which may suggest that their recorded physiological responses to depression are in some way different compared to the other participants and can potentially be characterised as outliers. As such, outlier detection could also be explored in identification of these participants which significantly differ from the rest of the sample. By filtering out outlier participants, we may be able to achieve better results using the same model. A downside to removing these outliers is the loss of training data which may be problematic given the already small dataset. Also previously mentioned in Section 5 is the exploration of early stopping for different participants rather than using a static set of epochs.

7 Conclusion

In this work, we explored the synthesis of training data using an evolutionary algorithm based generator and network reduction using a Contribution scoring metric as a means to improve classifier performance for scarce and noisy data problems such as depression classification based on physiological data of an observer. We constructed a simple two layer neural network to classify 4 levels of depression based on 85 time series summary features of physiological data collected from individuals observing videos of the depressed individuals, then applied the two aforementioned techniques to obtain results. We obtain statistically insignificant results, as well as results indicating that both techniques do little more than introduce more noise to the data. We conclude that both expanding the training data set using the evolutionary algorithm based generator and network reduction are overall ineffective at improving classifier performance for this problem .

References

1. Doersch, C.: Tutorial on variational autoencoders (2016)
2. Gedeon, T., Harris, D.: Network reduction techniques. In: Proceedings International Conference on Neural Networks Methodologies and Applications, AMSE„ AMSE 1991 (1991)
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014)
4. Molano, J.M., Paredes, R., Ramos, D.: Generative models for deep learning with very scarce data. CoRR **abs/1903.09030** (2019), http://arxiv.org/abs/1903.09030
5. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
6. Potvin, S., Charbonneau, G., Juster, R.P., Purdon, S., Tourjman, S.V.: Self-evaluation and objective assessment of cognition in major depression and attention deficit disorder: Implications for clinical practice. Comprehensive Psychiatry **70**, 53 – 64 (2016). https://doi.org/https://doi.org/10.1016/j.comppsych.2016.06.004, http://www.sciencedirect.com/science/article/pii/S0010440X16300062
7. Prechelt, L.: Early stopping - but when? In: Neural Networks: Tricks of the Trade, volume 1524 of LNCS, chapter 2. pp. 55–69. Springer-Verlag (1997)
8. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: Proceedings of the 24th International Conference on Machine Learning. p. 791–798. ICML '07, Association for Computing Machinery, New York, NY, USA (2007). https://doi.org/10.1145/1273496.1273596, https://doi.org/10.1145/1273496.1273596
9. Shukla, A., Pandey, H.M., Mehrotra, D.: Comparative review of selection techniques in genetic algorithm. 2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE) pp. 515–519 (2015)
10. Tomaka, J., Blascovich, J., Kelsey, R.M., Leitten, C.L.: Subjective, physiological, and behavioral effects of threat and challenge appraisal. (1993)
11. Vos, T., Allen, C., Arora, M., Barber, R., Bhutta, Z., Brown, A., Carter, A., Casey, D., Charlson, F., Chen, A., Coggeshall, M., Cornaby, L., Dandona, L., Dicker, D., Dilegge, T., Erskine, H., Ferrari, A., Fitzmaurice, C., Fleming, T., Murray, C.: Global, regional, and national incidence, prevalence, and years lived with disability for 310 diseases and injuries, 1990–2015: a systematic analysis for the global burden of disease study 2015. The Lancet **388**, 1545–1602 (10 2016). https://doi.org/10.1016/S0140-6736(16)31678-6
12. Zhu, X., Gedeon, T., Caldwell, S.B., Jones, R.: Detecting emotional reactions to videos of depression. In: IEEE 23rd International Conference on Intelligent Engineering Systems, INES 2019. IEEE (2019)