Neural Network Optimization on Depression Classification via Distinctiveness Pruning and Parameter Optimization

Jiayu Qin

Research School of Computer Science, Australian National University
u7138074@anu.edu.au

Abstract. Depression is an internalizing mental disorder but conventional diagnosis to detect depression may be subjective and time-consuming. In this paper, we start with a simple fully connected neural network to classify depression based on observers' physiological signals. We observe that having a large number of hidden units might decrease the overall efficiency, therefore we implement the distinctiveness pruning technique to remove insignificant and redundant neurons. We then apply Genetic Algorithm (GA) or Bayesian Optimization (BO) to tune the network by selecting a set of effective hyper parameters. We aim to slim down the model and improve the overall accuracy of neural network predictions. Our findings show that the overall accuracy predicted on the full set of features has increased from 29.2% (baseline) to 32.3% (distinctiveness pruning), 38.0% (distinctiveness pruning and GA), and 39.1% (distinctiveness pruning and BO). The application of pruning on the static weight matrix of the model and GA or BO on hyper parameters selection result in a significant increase in the model's accuracy.

Keywords: Neural Network, Depression Classification, Distinctiveness Pruning, Genetic Algorithm, Bayesian Optimization

1 Introduction

In this section we introduce the background of the issue, current situations, related work, and an outline of the improvements we made on the existing model in attempt to increase the effectiveness of the model.

1.1 Background

Depression is a mental health disorder characterized by persistent depressed mood or loss of interest in activities, and is causing significant impairment in daily life [1]. Depression is currently affecting more than 264 million people worldwide according to WHO statistics [1]. It is a leading cause of disability around the world and contributes greatly to the global burden of disease. The effects of depression can be long-lasting or recurrent and can drastically affect a person's ability to function and live a rewarding life.

1.2 Motivation

The causes of depression include complex interactions between psychological and biological factors [1]. Unfortunately, current diagnosis to detect depression may be time-consuming and subjective [3]. Recent advances made in affective computing technology demonstrate the possibility of using physiological signals to assist diagnosing depression [2]. By applying neural network (NN) learning in depression classification, it would increase the efficiency and accuracy compared to traditional diagnosing method. In this paper, we train neural networks on observers' physiological signals to detect depression levels of individuals in videos, then explore two potential improvements.

1.3 Dataset

Zhu et al. collected observers' physiological response signals such as Galvanic Skin Response (GSR), Skin Temperature (ST) and Pupillary Dilation (PD) to classify individuals' depression level [3]. For all these three physiological signals, they applied a min-max scaler normalization method to reduce the between-participant differences, which scaled signals to the range between 0 and 1. After the normalization process, they smoothed the signals to remove noise artifacts, and then extracted a number of features for each signal, including minimum, maximum, mean, standard deviation, variance, root mean square etc. They collected a total of 85 features from the three physiological signals: 23 (GSR) + 39 (PD) + 23 (ST), each signal with their extracted features saved in a separate excel files. The output depression level shown on the second column of each file (depr_label) has four depression categories: minimal, mild, moderate and severe depression which are encoded by integers 0, 1, 2, 3 respectively [3]. Some feature inputs may not be closely relevant to the

classification task and the dataset may benefit from feature selection process. Due to time constraint, we will discuss this data preprocessing in Future Works.

1.4 Investigations

We implement three potential improvement models on the baseline neural network: NN with two hidden layers, distinctiveness pruning applied on the single hidden layer, and hyper parameters optimizations via GA or BO to train the pruned network. We compare the performances of the existing model to ours on the same training and testing data. We see that the latter two variations have improved accuracies over the baseline.

2 Methodology

The original model by Zhu et al. is a simple one hidden layer fully connected neural network classification model with up to 85 features as input and the four classes of depression as output. This NN model has a sigmoid hidden layer of size 50. The output layer of four output neurons represents the four depression classes (They chose 50 as the number of hidden neurons after a series of accuracy tests because they found that between the range of 1-100 hidden unit sizes, 50 is optimal and gives the best overall performances). The NN uses Cross-Entropy loss function to evaluate deviations of the model predictions from ground truth and was trained using backpropagation with the Adam optimizer [3]. In our study we use the same implementation of this fully connected NN model as our baseline. We attempt three enhancements on this model. The first variation includes a second hidden layer in addition to the original. The second include only the original hidden layer, but some neurons are pruned via distinctiveness pruning method to reduce the model complexity. The third uses the same network as the second, but additionally uses GA or BO to tune all hyper parameters. For all the models, we use the same Zhu et al.'s pre-processed dataset. The models are adapted to 4 combinations of signals: 1) GSR+ST+PD; 2) GSR; 3) ST; and 4) PD. The first one, GSR+ST+PD concatenates all signal features into the full feature of size 85, while the rest uses features from the corresponding signal. We examine the same combinations for all models to stay consistent with the original model [3].

2.1 Neural Network with 1 hidden layer

Similar to Zhu et al.'s neural network model above, we construct our baseline NN model with 1 Sigmoid hidden layer of 50 neurons [3]. We train 100 epochs in total as training more epochs would likely result in overfitting which will be discussed later.

2.2 Neural Network with 2 hidden layers

As a supplementary to the NN with 1 hidden layer above, we also train a NN with one additional hidden layer. Adding one more hidden layer may be beneficial because the newly added hidden layer may capture additional information or information of different scopes from the features. It may specialize in transforming and combining features differently from the first hidden layer. While the original hidden layer has size of 50, the new hidden layer has 25. Giving the second hidden layer a smaller layer size compared to the first one ensures that the activations from the first hidden layer gets condensed and combined in the second hidden layer. The size of 25 also allows for a smooth feature transformation between the first hidden layer and the output layer, in that it forwards adequate amount of information between the two layers. The new hidden layer will also have Sigmoid as its activation function to enable non-linearity.

2.3 Distinctiveness Pruning

While the first variation explores increased complexity, the second goes in the opposite direction. This is for the consideration that an overly complex model may not always give better performances. As a Neural Network model grows bigger it occupies more memory and increases computational cost while the accuracy stops improving or even begins to decrease. Therefore, pruning redundant or less important hidden neurons from neural networks trained by back-propagation can be useful to improve the generalization performance. Pruning methods such as Brute Force or prune by inspection is used to reduce the hidden units. For this paper, we adopt the distinctiveness pruning method introduced by Gedeon, which is conceptually one of the simplest pruning methods [4]. Pruning to the neural network is done by the following steps: All training data is presented to the network one by one, and the activations (outputs of the Sigmoid function) of all hidden neurons are collected into a matrix such that each neuron would have a corresponding vector (of length the size of the training set) in the matrix. For each pair of the vectors in the matrix we calculate the angles between them which are limited to the range of 0° to 180° (An angle of 0° means that the two vectors are pointing towards the exact same direction, while the angle of 180° means the exact opposite). We prune neurons that are similar or

complementary to others. Specifically, a pair of neurons are considered similar when the angle between their vectors in the matrix is less than 15°, and complementary when more than 165°. For each similar neuron pairs, one of the two neurons is removed (by fixing its weights to 0), and optionally, the associated vector of the removed neuron can be added to that of the preserved neuron. This could be beneficial because if the two neurons are learning similar weights, their behaviors would be similar. Also, merging the weights could ensure the merged neuron gives output of similar magnitudes. As for complementary neurons, both are removed from the network. This could be beneficial as the two vectors are likely cancelling each other out [4]. The aim of this pruning is to remove undesired and redundant neurons, potentially reducing confusions and increasing performance of our model.

2.4.1 Parameter Optimization via Genetic Algorithm

So far, we have been tuning model hyper parameters manually. With this variant we try to optimize the parameters automatically using GA. GA selects solutions to a problem by mimicking the natural evolution process. Potential solutions within the population go through natural selection, reproduction, and mutation [5]. In our case, the solution is an array consisting of 6 hyper parameters for the 1-hidden layer pruned neural network. The parameters include number of epochs to train, number of epochs between pruning, batch size, learning rate, and the two pruning angle thresholds. We start with initialization where a set of solutions (the population) is generated. Each parameter of the solutions is chosen at random from a set of predefined values. For example, batch size would be one of 8, 16, 32, 64, 128. The evolution process then begins. In each generation, we compute fitness of each solution. The fitness is the performance of the models trained using the parameters encoded in the solution. With the fitness as probabilities, we select solutions in population such that solutions deemed fitter are more likely to be preserved while solutions deemed less effective are slowly eliminated. We implement Elitism in this stage by always preserving the best few solutions between generations. We then randomly pick pairs of solutions and perform crossover. This means random parts of the two solutions are chosen to form new solutions. Finally, we randomly mutate parts of some solutions by altering the encoded parameter. The above evolution would continue until either the max number of generations is reached, or if the solutions in the population has converged, meaning they look and perform similarly. It is expected from GA that it could combine good parts of different solutions into a fittest solution.

2.4.2 Parameter Optimization via Bayesian Optimization

Bayesian Optimization (BO) is an alternative to GA and is also used here to optimize hyper parameters. Parameter optimization problem is difficult to model so BO is especially helpful, as it approximates the objective function. It does so by sampling in the input space, first randomly but then with heuristics, to obtain observations at different locations of the objective function. With the estimated objective function, it then discovers a solution, a best guess, to the problem [6]. One aspect where BO may be better than traditional GA is that instead of choosing parameters from a set of predetermined values, BO guesses solutions on demand. In brief, GA is more discrete while BO is more continuous. We would provide BO the ranges of parameter values to sample, and let the observations be performances of the models trained with the guessed parameters. BO would produce a set of parameters that is most likely to maximize the performance. We use BO because GA is too computationally expensive, and BO may yield similar or better results while performing less computations.

2.5 Evaluation

We use the leave-one-participant-out cross-validation method described in Zhu et al.'s paper. This validation method leaves out all features belonging to one participant as validation set and group the rest to form the training set. To ensure fairness and integrity of the evaluation, each variation is trained 12 times (because there are 12 participants), each with a different participant left out for validation. This method resembles 12-fold validation, so test performance of the model is the average of the 12 models. In order to correctly perceive the performances of the models above, Zhu et al.'s used precision, recall and F1-score as evaluation measurements. F1-score takes the harmonic mean of precision and recall. And as there are four classes of depression output, we calculate the average precision, recall and F1-score for all output levels and give a view on general prediction performance. Also, the overall accuracy, the number of individuals correctly predicted with their corresponding depression levels over the total number of individuals, is calculated to evaluate the overall performance [3]. And according to these measurements, ideally a value closer to 1 indicates more accurate predictions.

3 Result

In this experiment, we are interested in the accuracy of depression level prediction based on observers' GSR, ST and PD features. The assessment of performance for each single signal is also important, so we do the comparison using NN (1 and 2 hidden layers), distinctiveness pruning, GA or BO on pruned model under four conditions: 1) GSR+ST+PD: using combined features extracted from all three signals; 2) GSR; 3) ST; and 4) PD, with the last three only using extracted features from the respective signal. When each single signal is trained, only the input feature size and model input layer is changed while everything else remains the same.

3.1 Neural Network with 1 and 2 hidden layers

Figure 1 shows the evaluation measurements of precision, recall and F1-score for the four depression levels under the four conditions (the three signals respectively and the three combined). It also compares the result of the baseline NN model which has only 1 hidden layer, and the viarient with 2 hidden layers. We can see that by combining all these 85 features as inputs, the baseline model, NN with 1 hidden layer, has an overall accuracy of 0.292 which means 29.2% of individuals have their corresponding depression levels correctly predicted. The 2 hidden layer NN model also gives the overall accuracy of 0.292. We conclude that compared to the baseline model, adding one more hidden layer has no improvement on the overall performance and is not a meaningful improvement.

Also, for the other three conditions with single signals as input only, we can see that NN with 2 hidden layers has a significant increase in the overall prediction accuracy with ST signal alone. But for GSR and PD, overall accuracy are even worse than NN with 1 hidden layer. Therefore, we conclude again that adding another hidden layer is not very effective.

As shown in the Figure, these evaluation measurements (average precision, recall and F1-score and overall accuracy) are all below 0.5 (far from the ideally value of 1), meaning that the combination of the training data and our NN model may not be perfectly adequate for this depression classification dataset. As the improvements of the 2 hidden layer model is minimal, future models should try other means to improve the evaluation results.

	NN											
	GSR			PD			ST			GSR+PD+ST		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
None	0.282	0.229	0.253	0.290	0.188	0.228	0.200	0.188	0.194	0.324	0.229	0.268
Mild	0.326	0.313	0.319	0.327	0.354	0.340	0.305	0.375	0.336	0.318	0.292	0.304
Moderate	0.217	0.271	0.241	0.265	0.271	0.268	0.200	0.229	0.214	0.188	0.250	0.214
Severe	0.404	0.396	0.400	0.317	0.396	0.352	0.333	0.229	0.272	0.380	0.396	0.388
Average	0.307	0.302	0.303	0.300	0.302	0.297	0.260	0.255	0.254	0.302	0.292	0.294
Overall accuracy		0.302			0.302			0.255			0.292	
	NN 2 hidden layer											
	GSR			PD			ST			GSR+PD+ST		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
None	0.222	0.208	0.215	0.250	0.146	0.184	0.250	0.250	0.250	0.257	0.188	0.217
Mild	0.351	0.271	0.306	0.295	0.375	0.330	0.304	0.354	0.327	0.268	0.313	0.288
Moderate	0.200	0.229	0.214	0.283	0.271	0.277	0.315	0.354	0.333	0.281	0.333	0.305
Severe	0.400	0.458	0.427	0.298	0.354	0.324	0.382	0.271	0.317	0.364	0.333	0.348
Average	0.293	0.292	0.290	0.281	0.286	0.279	0.313	0.307	0.307	0.292	0.292	0.289
Overall accuracy		0.292			0.286			0.307			0.292	

Fig. 1. Evaluation measures for depression levels on all features for NN with 1 and 2 hidden layers.

3.2 Distinctiveness pruning

As mentioned above, NN with 2 hidden layers does not increase the overall performance much. Therefore, we only applied distinctiveness pruning on NN with 1 hidden layer. After the 100 epochs of training, we prune the redundant and useless neurons and train another 30 epochs to evaluate the performance. If we train more than 30 epochs, performance starts to decline which may be due to the overfitting problem and more detail will be covered in the discussion section.

Figure 2 shows the average number of neurons pruned in the NN along with the overall evaluations on all features. As shown below, the ratio differences of average pruned neurons for the four conditions are relatively large. For the combined signals condition, we pruned 34.5% of the total of 50 hidden units. And for other three conditions with single signals alone as inputs, the average neurons pruned are even larger, being 50%, 74% and 51% respectively. This indicates that the hidden layer size of 50 is more than enough to extract information useful for classifications from the combined features which have size 85 (GSR+PD+ST). This is even truer for single signals with smaller sizes 23 (GSR), 39 (PD), 23 (ST).

Compared to Figure 1 above, we see that pruning does increase the generalization performance a little. For the combined case, distinctiveness increases the overall accuracy from baseline of 0.292 to 0.323. This model with pruning also does better than the NN variant with 2 hidden layers. The same is also true for single signals conditions, where distinctiveness pruning increases the evaluation measures marginally (2.6% increase in GSR, 1.6% in PD, and 2.1% in ST).

	Distinctivesness Pruning											
	GSR			PD			ST			GSR+PD+ST		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
None	0.261	0.250	0.255	0.279	0.250	0.264	0.192	0.208	0.200	0.263	0.208	0.233
Mild	0.372	0.333	0.352	0.355	0.458	0.400	0.315	0.354	0.333	0.362	0.354	0.358
Moderate	0.222	0.250	0.235	0.256	0.229	0.242	0.224	0.229	0.227	0.259	0.313	0.283
Severe	0.469	0.479	0.474	0.364	0.333	0.348	0.405	0.313	0.353	0.408	0.417	0.412
Average	0.331	0.328	0.329	0.313	0.318	0.313	0.284	0.276	0.278	0.323	0.323	0.321
Overall accuracy		0.328			0.318			0.276			0.323	
avg. neurons pruned		25.25			37			25.67			17.25	
avg. pruned ratio		0.505			0.74			0.5134			0.345	

Fig. 2. Evaluation measures for depression levels on all features for distinctiveness pruning

3.3.1 GA on pruned network

Additionally, we use GA to select hyper parameters on pruned network to improve the accuracy of the pruned model above and the evaluation results are given in Figure 3. Compared to the distinctiveness pruning model, for the combined signals condition, the overall accuracy has increased from 0.323 to 0.38 (as shown below in blue). This 5.7% of increase gives the support that GA has had an improvement on pruned model. Likewise, notable amount of improvements are also observed for single signals GSR, PD, and ST. This is very true for ST which sees an 8.9% improvement over baseline.

In all of our tests, the combined features (GSR+PD+ST) always yield the highest performance in our evaluations because in this case having more information translates into better accuracy.

	GA											
	GSR			PD			ST			GSR+PD+ST		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
None	0.375	0.063	0.107	0.263	0.208	0.233	0.381	0.333	0.356	0.356	0.333	0.344
Mild	0.000	0.000	0.000	0.400	0.333	0.364	0.327	0.375	0.350	0.435	0.417	0.426
Moderate	0.333	0.708	0.453	0.313	0.417	0.357	0.373	0.396	0.384	0.316	0.375	0.343
Severe	0.366	0.625	0.462	0.380	0.396	0.388	0.386	0.354	0.370	0.432	0.396	0.413
Average	0.269	0.349	0.256	0.339	0.339	0.335	0.367	0.365	0.365	0.384	0.380	0.381
Overall accuracy		0.349			0.339			0.365			0.380	

Fig. 3. Evaluation measures for depression levels on all features for GA on pruned network

As shown in Figure 4, we compare the overall accuracy on all features for NN (shown in yellow), Distinctiveness pruning (green) and GA on pruned network (red). Clearly from the figure, compared with the baseline model shown in yellow, both distinctiveness pruning and GA on pruned model have improved the overall performance. Also, using GA to select hyper parameters for pruned network further increases the performance. Nevertheless, GA is very computationally expensive. Even though we observe that the population generally converges around generation 7, a generation of 7 still requires the evaluation of at least 3360 models (12 participants × 40 population size × 7 generations).

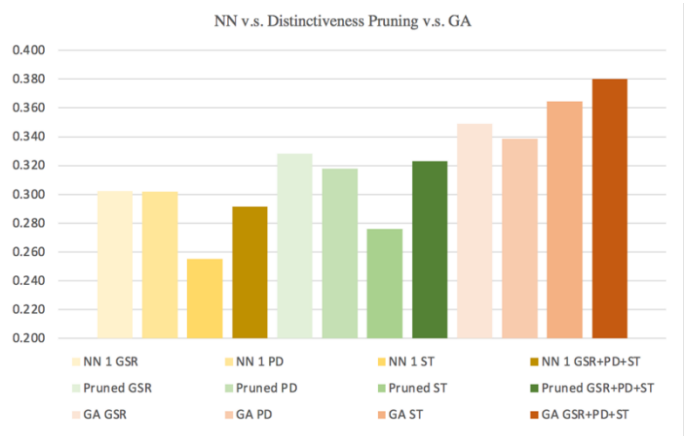

Fig. 4. Overall accuracy on all features for NN, Distinctiveness Pruning, and GA on pruned network

3.3.2 BO on pruned network

As a substitution method for selecting hyper parameters, BO does similar things as GA. And considering about its performance, we can see from Figure 5, the overall accuracy for the combined features reaches 0.391, slightly higher than GA (0.38). Even though GA and BO yield similar overall accuracies, BO largely reduces computational time.

	GA			BO		
	Precision	Recall	F1-score	Precision	Recall	F1-score
None	0.356	0.333	0.344	0.471	0.333	0.390
Mild	0.435	0.417	0.426	0.364	0.417	0.388
Moderate	0.316	0.375	0.343	0.345	0.417	0.377
Severe	0.432	0.396	0.413	0.422	0.396	0.409
Average	0.384	0.380	0.381	0.400	0.391	0.391
Overall accuracy		0.380			0.391	

Fig. 5. Overall accuracy on GSR+PD+ST for GA and BO on pruned network

3.4 Discussion

As shown in Figure 6 below, we compare performances of all models. We see that NN 2 hidden layer does not improve baseline much. Furthermore, distinctiveness Pruning not only simplified the model but also improved the performance. In addition to the pruned model, GA and BO both select optimal hyper parameters to further push performances to the limits.

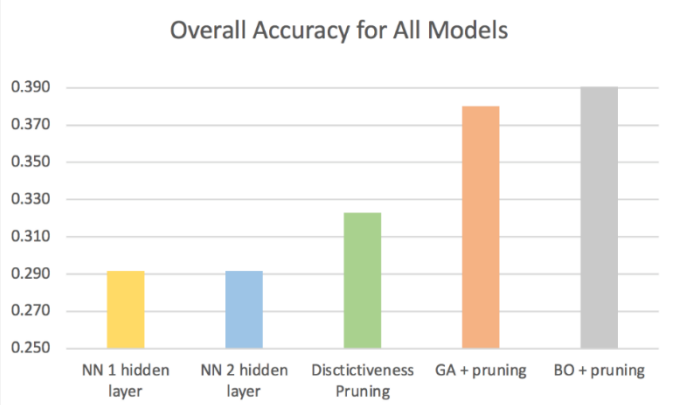

Fig. 6. Overall accuracy for all models with all signals combined

Moreover, in NN with 1 hidden layer, due to the small dataset size of 192 combined with the use of all 85 input features, this model encountered the overfitting problem. As shown in the Figure 7 below, the training accuracy is much higher than testing accuracy in NN baseline model, which means that overfitting occurs during the training process. NN baseline model with all three signals combined reaches an average training accuracy of 55.7% while the average testing accuracy stays around 29.2%. This 26.5% difference shows that our model performs decently on the training dataset, but when the model starts to predict on the test data, the accuracy remains low. It is likely the model becomes specialized in predicting the training set, and is confused on test set, the unseen data. We implement improvements such as distinctiveness pruning, and GA or BO on pruned model. As we can see that GA and BO reduce the difference between training and testing performance to 8.6% and 10.2% respectively. Indeed, the differences become smaller for GA and BO. The overfitting problem is mitigated.

	NN 1 hidden layer	Distinctiveness Pruning	GA on pruned network	BO on pruned network
Avg. training accuracy	0.557	0.583	0.466	0.493
Avg. testing accuracy	0.292	0.323	0.380	0.391

Fig. 7. Average training and testing accuracies of NN with 1 hidden layer, distinctiveness pruning, GA and BO on pruned network

4 Conclusion

In this paper, we investigate the performance of NN with 1 and 2 hidden layers, the distinctiveness pruning, and hyper parameter optimization via GA or BO. In terms of the NN model, adding one more hidden layer does not increase the

overall accuracy much. Next, by applying distinctiveness pruning method to this NN baseline model, in average 17.25 hidden neurons are pruned, making the model simpler and more efficient. The overall accuracy increases by 3.1% over baseline. Furthermore, after optimizing hyper parameters, GA or BO improve performances by 8.1% and 9.9% respectively.

From the results above, we conclude that distinctiveness pruning improves the model to some extent (at this stage all parameters are manually chosen). With GA or BO, the performance can be improved further because they test many more combinations of parameters and is able to find the optimal sets. While GA produces reliable results, it depends heavily on good population initializations because final parameters must be chosen from the original population. Also, GA requires plenty of time and resources for simulations because better results usually undergo many crossovers and selections. On the other hand, BO is more flexible because it chooses parameters from a range instead of predefined values so the parameters can be more precise. For example, while GA can only choose batch size from one of 8, 16, 32, 64, and 128, BO can produce a value such as 54.5. Besides, BO is not as computationally expensive as GA, and usually reaches solutions as good as that of GA in much less simulations.

4.1 Future Works

To further improve performance, the overfitting problem which still exists after all attempts needs to be remediated. Two possible solutions are to gather more experiment data to train the model or using dropout method to better balance and tune the weights in the model. Early stopping may also be applied in this work to improve the overfitting problem.

Also, we use all features of the four conditions in the NN models with pruning, including each of these three psychological signals separately and all three combined. It is likely that some of those extracted features from the signals may be redundant. These redundancies may bring unnecessary complexity to the model and have an adverse impact on the performance. Therefore, implementing feature selection process may simplify the model input and reduce confusion, thereby improving performance. GA with Linear Discriminant Analysis (LDA) fitness would be a good candidate method for this task.

References

1. World Health Organization. (2021). Depression. Retrieved from https://www.who.int/health-topics/depression#tab=tab_1
2. S. Potvin, G. Charbonneau, R.-P. Juster, S. Purdon, and S. V. Tourjman. (2016). Self-evaluation and objective assessment of cognition in major depression and attention deficit disorder: Implications for clinical practice. Compr. Psychiatry, vol. 70, pp. 53–64.
3. Zhu, X., Gedeon, T., Caldwell, S., & Jones, R. (2019). Detecting emotional reactions to videos of depression. In INES'19: IEEE 23rd International Conference on Intelligent Engineering Systems, vol. 1, pp. 6.
4. Gedeon, T. (1995). Indicators of hidden neuron functionality: the weight matrix versus neuron behaviour. Proceedings 1995 Second New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, vol. 1, pp. 26.
5. A.E. Eiben, J.E. Smith. (2015). Introduction to Evolutionary Computation. (2nd ed.). Springer.
6. Jason Brownlee. (2019). How to Implement Bayesian Optimization from Scratch in Python. Retrieved from https://machinelearningmastery.com/what-is-bayesian-optimization/