Assessing evolutionary algorithms and input analysis techniques for explaining EEG alcoholism readings

Tim Hill

Research School of Computer Science,
Australian National University,
Canberra Australia
u2557066@anu.edu.au

Abstract. We use a simple neural net to predict alcoholism in patients from summarised EEG measurements. We compare a grid-based hyperparameter tuning approach to an evolutionary algorithm with regularized fitness, looking at time and accuracy. We find the latter a significant improvement from a zero-knowledge start, giving a model with 61.11% test accuracy. Then we use a range of input analysis techniques to see if we can understand which inputs are relevant for predicting alcoholism from an EEG. We use a weight-base magnitude measure, a functional measure that compares inputs as vectors and a modern sensitivity measure, LIME. We also compare these with two feature selection genetic algorithms with weighted fitness functions. Two evaluations are done; one uses all measures to feature select the best inputs and compare accuracy with a baseline, and the other visualises relevant measures on a topology-preserving azimuthal equidistant projection of a skull, to provide better semantic representation of inputs. We find the magnitude measure and LIME best for modelling prediction on reduced inputs, and LIME is most promising for explaining EEG readings, when combined with the skull visualisation technique. Genetic algorithms remain potentially effective feature selectors that maintain accuracy, with a number of caveats. Of note, a genetic algorithm that penalises number of inputs actually performs better than one that doesn't.

Keywords: neural nets, EEG, alcoholism, hyperparameter-tuning, evolutionary algorithms, genetic algorithms, input analysis, feature selection.

1 Introduction

Electroencephalograms (EEGs) are a monitoring tool to measure brain wave activity (Yao et al. 2018, Yao et al. 2020). They have a wide range of applications in disease diagnosis, behaviour monitoring and more recently human-computer interaction (Vernon et al. 2016).

With many electrodes placed at intervals around the skull, they provide a great deal of information about the brain activity. Nevertheless, the information is difficult to interpret and it is hard to separate patterns from environmental interference, measurement error and other noise. Additionally, EEGs tend to have markedly differing signals per subject, making generalisations difficult.

Neural nets are one potential tool to interpret EEG signals for diagnosis and explanatory uses (Yao et al. 2018). However, neural nets themselves are often difficult to interpret and can provide different outcomes depending on the input data and a range of hyperparameters. While current state-of-the art neural nets can be very deep with billions of parameters (Brown et al. 2020), by using simple neural nets, we may be able to better understand how they interpret the input data, and use that information to target more relevant features and improve performance.

Another issue that neural nets have is the choice of suitable hyperparameters. While there are many rules-of-thumb as to which hyperparameter to use (LeCun et al. 1998), understanding how several hyperparameters interact and which ones are significant for a dataset is no trivial task, and can add orders of magnitude on to the training time of a neural network, and result in suboptimal networks.

While a grid search is one approach to hyperparameter optimization, evolutionary algorithms are another (Rothwell 2018). Evolutionary algorithms (EAs) are often used to solve similar problems to neural nets. A key distinction is that whereas neural nets are a natural fit for building predictions based on large datasets, evolutionary algorithms are well-suited to problems where we have a way of evaluating solutions via a fitness function, if not a dataset.

Evolutionary algorithms borrow concepts from the theory of natural evolution and apply it to computing optimization, modelling and simulation problems. A genome becomes the stored representation of a solution, random mutation on the chromosome is implemented to assist in exploring new solutions, and crossover techniques are used to mix chromosomes, similar to sexual reproduction in nature. One of the most powerful aspects of EAs is the ability to create a fitness function customised to a specific goal. Various selection techniques can then be used to ensure the 'fittest' genomes are allowed to crossover and continue to the next generation, and over time some of the results may converge to a solution to a problem.

EAs can't guarantee an optimal solution or even a suitable solution, but their ability to explore the search space based on combining the best of previous generations means that they can perform surprisingly well on complex problems, such as providing suitable solutions for NP complete problems in a reasonable time (Holland 1975).

Genetic algorithms are a subset of EAs that specifically code genomes as bits in a string. Here, mutation and crossover are much more analogous to living DNA with their four protein combinations similar to two binary digits.

Finally, to tackle the challenge of neural net interpretability, we consider techniques that measure individual input features' contribution to output classes. Such measures may be of type 'magnitude' that look at the weights of a trained network, 'functional' that look at how the network responds to data when used for inference, and 'sensitivity' metrics, that perturb input data to see how sensitive the network is to adding error signals (Gedeon 1997). We take this opportunity to compare these with genetic algorithms for feature selection (Sharma & Gedeon 2013, Erguzel et al. 2015).

A further interpretability technique for EEGs was recently proposed by (Bashivan et al. 2015) whereby each electrode response on the head is mapped onto a 2D space in the same way that the Azimuthal equidistant projection (AEP) or polar projection maps the 3D globe to a 2D map. Frequency ranges are mapped to the RGB colour model for a single semantically dense visualisation of all input responses. We combine this with our input analysis techniques for a new way of model interpretation.

2 Methodology

The following diagram outlines the procedure used in this paper to tune and build a neural network, perform input analysis and feature selection, then evaluate, analyse and visualise the results.

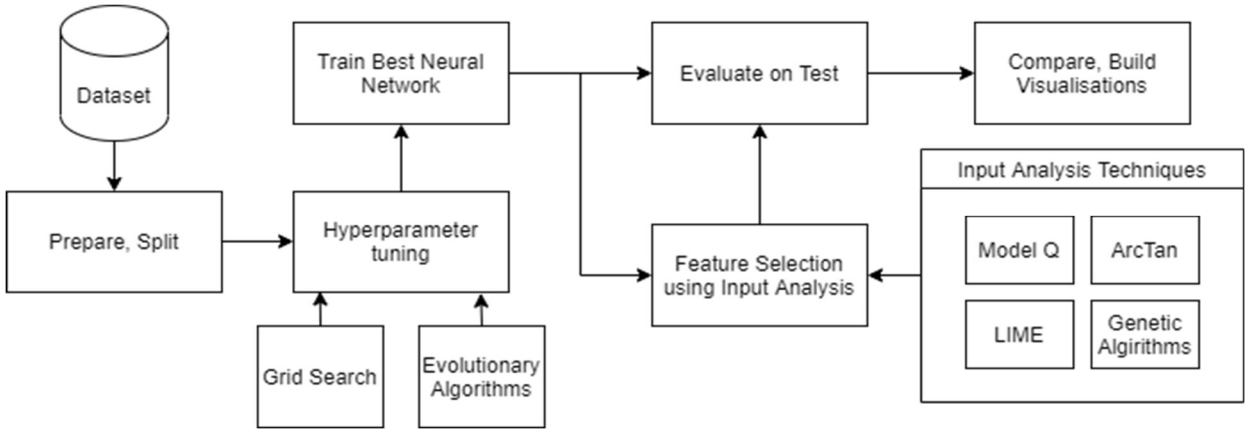


Fig. 1. High-level summary of the procedure used in this paper.

2.1 Data analysis and description

The dataset is based on (Begleiter 1999) and consists of 11,057 records across 122 subjects, 77 labelled alcoholic and 45 control. Each subject was shown one of five stimuli from (Snodgrass & Vanderwart 1980) and their EEG response was recorded via 64 electrodes (channels) on the skull. Each channel's readings were recorded at three different frequencies, theta (4–7 Hz), alpha (8–13 Hz) and beta (13–30 Hz) for 1 second recorded at 256Hz. The provided dataset summarised the responses to an average value over the time, giving 192 numerical features and a stimulus identifier. Each subject undertook 120 such trials, but not all trials are provided in the dataset; the number of trials captured per subject ranged from 30 to 119. All trials for a single subject are consistently labelled as either alcoholic (1) or not (0).

The dataset is unbalanced at the classification level, as well as trials per subject and stimuli tested, per subject and overall. There were no missing values in the dataset. The source dataset does not describe how the alcoholic or control subjects were classified as such.

2.2 Data preparation

A number of features were dropped from the raw dataset. The trial number feature was not used for prediction, to avoid spurious learning correlations. Similarly for the subject ID, although it was used in for cross-subject splitting. The stimulus feature, despite having potential predictive power, was dropped as our focus is on analysis of electrode responses.

To improve training, each of the 192 numerical features were normalised on the whole dataset (LeCun et al. 1998) so that all values are between 0 and 1, according to the formula (sklearn 2020):

$$x := \frac{x - x_{min}}{x_{max} - x_{min}} \times (x_{max} - x_{min}) + x_{min} \quad (1)$$

Here, x is the observation for that feature and x_{min} and x_{max} are each feature's minimum and maximum values.

We do a cross-subject split, roughly 70/15/15 into train/validation/test sets, by ensuring all observations from individual subjects are grouped into one set. This gives us 85/19/18 subjects per set or 7,760/1,620/1,677 records, respectively. The small number of subjects combined with the overrepresentation of alcoholics causes a high degree of variance in the random split. We manage this in three ways:

1. We choose to forgo cross-validation, so maintain a constant validation set during hyperparameter tuning. This reduces variance at the expense of bias. This is acceptable for our purposes.
2. The dataset is unbalanced but without a predictive task, we can't say whether false positives or negatives are a bigger problem. As a result, we experiment with adding the receiver operator characteristic area under curve (AUROC or AUC) to the evolutionary algorithm's fitness function, as it is both scale and classification-threshold invariant.
3. Ideally, we would normalise only from training data, but the small number of subjects and variation between subjects meant that this often gave a test set notably outside the distribution, leading to wide variation in accuracy. We mitigated this by normalising across the entire dataset, and accepting some loss of generalisation.

2.3 Hyperparameter tuning and neural network architecture

We performed hyperparameter tuning by grid search and evolutionary algorithms and compared the results. As this meant that we were training hundreds of neural networks, we choose to use only the summarised EEG data described above, on a network with a single hidden layer. Only the EEG signals were used as inputs, for a total of 192 inputs. The output was a simple binary classifier and loss for backpropagation was measured via Cross Entropy Loss (PyTorch 2019a).

$$\text{Loss}(y, \hat{y}) = -\log\left(\frac{e^{\hat{y}}}{\sum_j e^{y_j}}\right) \quad \textbf{(2)}$$

Here, y_1 and y_2 are the possible classes, and $\hat{y}$ is the predicted classification.

As a baseline, a grid search was performed across eight hyperparameters, for a total of 432 networks to evaluate. Since a grid search is inherently slow, even with a small net, sensible default values were necessary, and were chosen from current best practises (e.g., Nair & Hinton 2010, PyTorch 2019b, Xie et al. 2021) and manual parameter exploration[1]. Only a small number of options was provided per hyperparameter, meaning only a small amount of the search space could be explored. The choice of activation function was sigmoid or ReLU (Nair & Hinton 2010). The choice of optimizer algorithm was *Adam* (PyTorch 2019b) or stochastic gradient descent with momentum (SGD) The hyperparameters explored are shown in Table 1.

Table 1. The selected grid search hyperparameters and their values. All permutations of the below values were tested, except for momentum values which were only tested in conjunction with the SGD optimizer.

Hyperparameter	Set of parameter values tested		
Batch size	Full batch	100	
Number of epochs	100	200	300
Number of hidden neurons	10	20	50
Activation function	sigmoid	ReLU	
Optimizer	Adam	SGD	
Optimizer learning rate	3×10^{-4}	1×10^{-3}	
Optimizer weight decay	0.01	1×10^{-10}	
Momentum (*SGD only*)	0.5	0.9	

For comparison, two evolutionary algorithms were run on the same hyperparameters, differing only by their fitness function. Each used a population size of 6 for 200 generations. A tournament selection algorithm was used with $k = 3$ and a custom elitism algorithm (Pakt 2021) was used on top of simple selection (Fortin et al. 2012), whereby the best performing model was copied to the next generation. This was seen to significantly stabilise the algorithm given the broad search space. Crossover probability was 0.7 and mutation probability was 0.2. As the hyperparameters are all quite different in use, we required a custom value-encoding that stored each parameter separately and used one-point crossover

[1] E.g., it was found that overfitting often occurred with more than around 300 epochs.

in selection. Within this structure, mutation was done per hyperparameter; each mutating chromosome had a chance of mutation of *each* hyperparameter of 0.1, but with the proviso that *at least* one would mutate, to ensure sufficient variation over time.

Mutations had to be customised per hyperparameter as a result. These were randomly sampled by distributions according to Table 2. All individuals were initialised according to the same mutation sampling distributions.

Table 2. Hyperparameter ranges and mutation rates.

Hyperparameter	Mutation sampling distribution	Range or set of values
Batch size	Categorical	100, 200, 1000, or full batch
Number of epochs	Uniform	100 to 300, in multiples of 50
Number of hidden neurons	Uniform	10 to 50
Activation function	Categorical	Sigmoid, ReLU
Optimizer	Categorical	Adam, SGD
Optimizer learning rate	$r \times 10^{-q}$	$r \in [1,9], q \in [-9,-1]$ (both uniform)
Optimizer weight decay	$r \times 10^{-q}$	$r \in [1,9], q \in [-9,-1]$ (both uniform)
Momentum	Beta distribution	range [0,1], $\alpha = 10, \beta = 2$

The goal was to be as agnostic as possible with the search space. We see for example that unlike in the grid search, the learning rate and weight decay can take numbers across 9 orders of magnitude. However, based on trial and error, some constraints were necessary:

- The batch size was limited to a small number of choices, as arbitrary batch sizes performed extremely slowly and usually resulted in poor models, resulting in a very inefficient parameter search.
- Unlike in the grid search, the momentum was free to mutate even when SGD was not the optimization algorithm, so some prior belief was used. Hence the momentum was sampled from a beta distribution with mode of 0.9. The aim here was to mimic a weak beta prior commonly used in Bayesian data analysis (Johnson et al. 1995).

The two algorithms differed only in their fitness functions:

$$f_{EA1} = \text{argmax}_{\text{M}}\{\, \text{accuracy}(M) \,\} \quad \textbf{(3)}$$

$$f_{EA2} = \text{argmax}_{\text{M}}\{\, \text{accuracy}(M) + \text{AUC}(M) \,\} \quad \textbf{(4)}$$

That is, we choose the model M that maximises either the accuracy or sum of accuracy and AUC. As mentioned, the latter is proposed as a stabilisation and regularisation technique.

For each of these three approaches, the hyperparameters for the best validation score was used to train a network with the *combined* training and validation data. This was done 10 times and the average accuracy was scored against the hold-out test set. We chose the best of these as the default hyperparameter combination for the tasks below.

2.4 Input analysis and feature selection

Three techniques were taken from Gedeon's input measures analysis paper (Gedeon 1997), which performed measurements on a network topology GIS dataset and an eye-gaze dataset. This paper used a range of magnitude, functional and sensitivity techniques. Our aim here is to compare results from each and determine if they help summarise or explain the dataset or predictive model. We took the best performing magnitude measure (*model Q* in the paper), and a functional measure relying only on the inputs (model *I* in the paper, named here as *Arctan*). A number of modern input analysis techniques have arisen since this paper, and we use one here for the sensitivity analysis; Local Interpretable Model-agnostic Explanations, or LIME (Ribeiro et al. 2016). We further supplement these with use two evolutionary algorithm approaches for feature selection. All are detailed below.

(Wong, Gedeon & Taggart 1991) proposed the following measure of contribution a unit *i* in one layer has to a unit *j* in the next, for weights *w* and *n* neurons in *i*'s layer:

$$P_{ij} = \frac{|w_{ij}|}{\sum_{r=1}^{n}|w_{rj}|} \quad \textbf{(5)}$$

This can be used to calculate the contribution of any input unit *i* to an output unit *k* (here, for a single hidden layer of *n* units), which we name model *Q*:

$$Q_{ik} = \sum_{r=1}^{n\,h}(P_{ir} \times P_{rk}) \quad \textbf{(6)}$$

We do not measure the contribution of biases here, meaning the contributions to the binary classifier are symmetrical around zero. This allows us to look at contribution to $\hat{y} = 1$ only.

The Arctan measure, described in (Gedeon & Harris 1991) and adapted in (Gedeon 1997) took the combined train and validation set (normalised) and treated each feature as a vector containing all values from its observations. This vector was compared against all other vectors using an inverse tan angle, where a and b are the input vectors for a single feature $c \in C$:

$$\text{angle}_{Arctan}(a, b) = \tan^{-1}\left(\sqrt{\frac{\sum_C a^2 \times \sum_C b^2}{\sum_C (a \times b)^2}} - 1\right) \quad (7)$$

Whereas (Gedeon 1997) used the vectors from both training and test data, we used the combined training/validation set only to give a more generalised measurement on the holdout set, resulting in 192 vectors of dimension 9,390. We then calculated the inputs whose mean of angles to all other inputs were most similar, and ranked accordingly.

LIME aims to treat a classifier as a black box, and identify, only from inputs and outputs, an equivalent interpretable model over the interpretable representation that is faithful to the original classifier. It does this by using a sparse linear model with regularisation as the explanation model. We use the python library at (Ribeiro 2021), and measure all inputs for their explanatory power. Perturbation is done by sampling from a standard normal distribution and inverting our initial normalization on the combined training data. We keep the default sparse linear model exponential kernel width of 0.25, discussed in (Ribeiro 2016). We can then see what inputs contributed to each output for a particular trial. Taking the average of all test trials gives us a summary of sensitivity input significance.

Two genetic algorithms used for feature selection were used by way of comparison. Here, we use a 192-length binary array to determine if an input feature is on or off, and simply evaluate each resulting phenome against the neural network.

GA1 uses a similar fitness function to EA2 above to regularise the model:

$$f_{GA1} = \text{argmax}_{\text{M}}\{\, \text{accuracy}(M) + \text{AUC}(M) \,\} \quad \textbf{(8)}$$

We expect that this network will attempt to use all its features to get the best fitness score, so propose the following novel modification to improve feature selection.

GA2 attempts to mimic penalised feature selection techniques such as Lasso (Tibshirani 1996), by adding a penalty term that discourages models with more input features:

$$f_{GA2} = \text{argmin}_{\text{M}}\{\, \alpha \times \text{error}_{\text{rate}}(M) + \beta \times (1 - \text{AUC}(M)) + \lambda \frac{n(M)}{N} \} \quad \textbf{(9)}$$

Here, we attempt to *minimize* the fitness function, so measure the inverse of accuracy (error rate) and AUC, and add a penalty term. $n(M)$ is the number of inputs in the model and N is the maximum, 192. The coefficients α and β are both set to 0.5 and $\lambda = 0.3$. A larger λ will favour smaller models, while $\lambda = 0$ is equivalent to GA1.

Unlike in the hyperparameter algorithms, we don't need elitism to stabilise the model, so we choose a simple two-point crossover with crossover probability 0.8, mutation probability 0.1, and population size of 10, again across 200 generations. This allows for more mixing of fit models, without the risk of losing evolutionary divergence that elitism has. With the slightly larger population size, we use tournament selection with $k = 4$.

Having obtained a number of magnitude measures, (Gedeon 1997) compared them by removing similar input pairs, against a brute force removal of all combinations of two input neurons time, but his models only had 12 and 16 inputs. With 192 inputs here, this was not feasible. Instead, we feature-selected the baseline model by creating new classifiers using the top 16 and 64 most distinct input neurons according to each of our five measures, and compared accuracy against other classifiers, as well as classifiers built on the *least* distinct 16 and 64 neurons, called the contra-set. The aim here is to determine whether each technique is effective on this dataset at a very coarse level.

Classifiers were built on the same original network architecture as above, with the same hyperparameters of the best full network (baseline), and for each configuration, 10 models were trained and the average accuracy taken. For like-to-like comparison, and to manage complexity, there was no hyperparameter tuning for these networks, and rather than pruning input to hidden weights from the original, we created a new net each time. This means we did not expect the accuracy to compare to the full cross-validated network.

2.5 Visualisation

The above approach provides a quantitative evaluation of each input analysis technique, but with 192 inputs, it is still hard to understand what it means that any input is more significant for prediction than another. It would be better to translate these inputs back to their initial locations as electrodes on the skull, but flattened into a single 2D image. We use

the technique described in (Bashivan 2015) on the above input analysis techniques, by translating each technique's input scores into values corresponding to each of the 192 electrode/frequency measurements, and normalizing to the [0,1] range. We can then map those to an RGB colour palette, with each colour corresponding to each of the three wavelengths at a point, then apply the AEP projection against the actual location of electrodes in 3D space obtained from (Bashivan 2020). Rather than using Bashivan's approach of interpolating spaces between electrodes via the Clough-Toucher scheme, we use a simple linear interpolation, which gives a sharper image, and emphasises the limitation that the measurements are actually only at discrete locations. This is most relevant for the genetic algorithm, which is a feature selector, so it only provides a binary measure for each input.

3 Results and Discussion

3.1 Accuracy on all input features

The grid-based hyperparameter tuning took six hours and had a mean accuracy of 61.07%. EA1 gave a mean accuracy of 60.58% and EA2 was best at 61.11%. Both took around three hours to run 200 generations. The AUC scores were 0.50, 0.52 and 0.53 respectively. EA2, by incorporating AUC in its fitness function proved to be the best at generalising to the holdout test set on both measures. Results are shown in Table 3. The EA2 model's hyperparameters were thus used for the remaining experiments.

It is worth noting that most models tended to have more false positives, which was unsurprising given that the dataset was unbalanced in favour of alcoholics. The test set had a somewhat better balance, so we find the EA2 fitness function a useful applied approach for generalising a model. We also see evidence of hyperparameters that favour generalisations in this choice: fewer epochs and fewer hidden layers.

Furthermore, both EAs had competitive performance and ran faster, despite having a much wider search space. In fact, while the best model for GA2 was at generation 150, there was little improvement beyond generation 30 (Fig. 2), and already hyperparameters such as learning rate and weight decay were approaching common defaults. This can be attributed to the use of elitism.

Table 3. Best hyperparameter tuning parameters and mean results on test set over 10 runs. The hyperparameters are, in order: number of hidden layers, learning rate, batch size, number of epochs, optimizer, momentum, activation function and weight decay parameter.

Approach	Best hyperparameter set	Accuracy (%)	AUC
Grid search	20, 0.0003, full, 100, Adam, 0, sigmoid, 1e-10	61.07	0.50
EA1	45, 0.0001, full, 250, Adam, 0.94, ReLU, 9e-07	60.58	0.52
EA2	14, 0.002, 200, 100, SGD, 0.80, ReLU, 0.002	**61.11**	**0.53**

The lack of differentiation between many of the models, and variation in results suggest that (a) neural nets are effective at finding patterns in a range of situations and (b) there is a limit to how well this simple architecture can be used to predict this data in a generalised way. The scores broadly align with (Li et al. 2017, Yao et al. 2018) which show accuracy of a number of networks on the un-summarised dataset ranging from 60.5% to 75.6% – we would not expect such a simple net to be too high on this scale. Nevertheless, we see that evolutionary algorithms are effective at hyperparameter tuning, provided a suitable fitness function is chosen. Unfortunately, they come with their own set of parameters that need tuning. Despite this, in this case they compare favourably to a grid search.

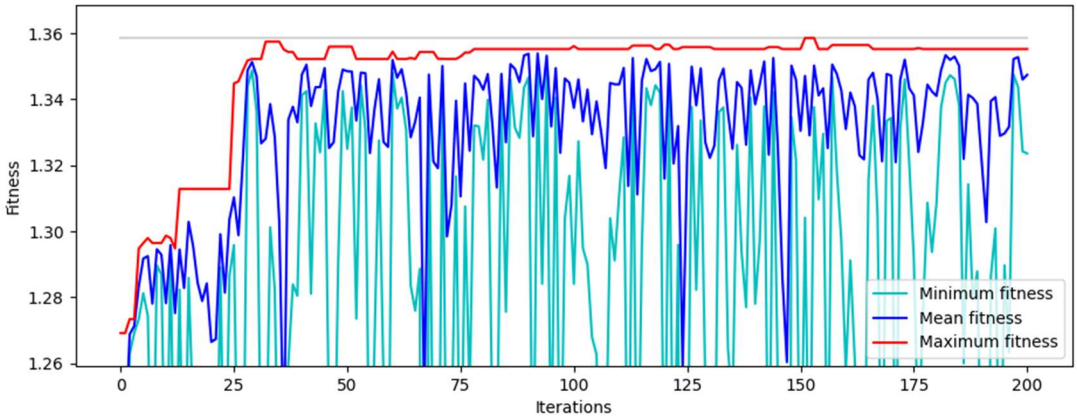


Fig. 2. Fitness of EA2 over 200 generations (cropped). Note that while the best fitness was reached around epoch 150, there was very little change in maximum fitness from epoch 30.

3.2 Input analysis techniques

Each of the input analysis techniques produced markedly different orderings of distinctiveness on this dataset. There were no common features in the top ten most distinct inputs across model *Q*, Arctan and LIME measures.

Evaluating accuracy on reduced models with 16 inputs, both model *Q* and LIME performed the best (Fig. 3). Most feature reduced models performed better than the full model, which is likely indicative that our simple neural network was unable to learn generalised features from the noise in this dataset. However, when comparing AUC, we found all feature selected models were worse than the full model. For the larger 64-input models, the functional model performed better and LIME performed the worst. Model *Q* remained consistent.

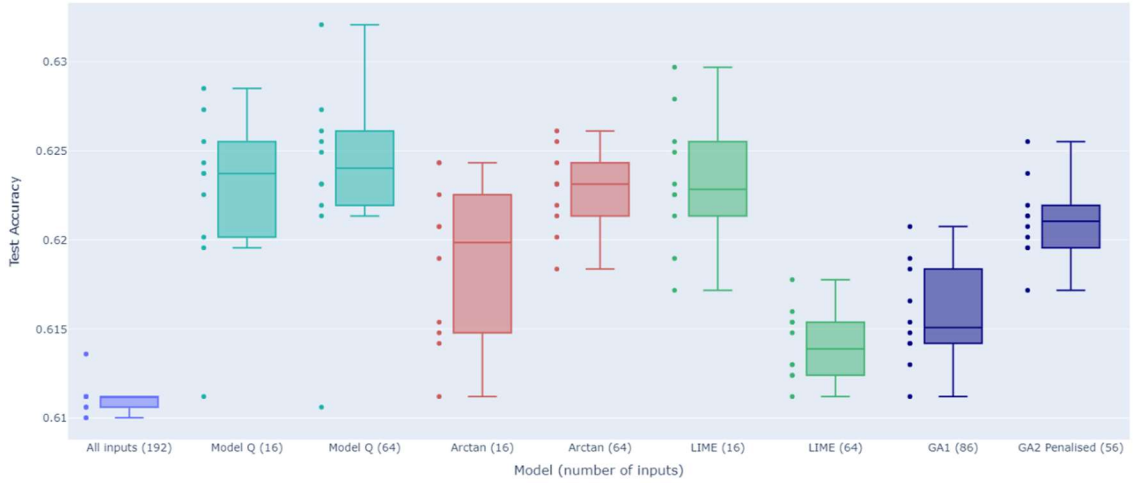


Fig. 3. Box plots comparing accuracy of 16-input feature selected models and both GA feature selected models against the full model.

Unlike (Gedeon 1997), the model *Q* magnitude technique outperformed the functional techniques with this dataset, particularly on more aggressive input pruning (Table 4). This may be related to the much larger number of inputs here than in those datasets (12 and 16). Neither functional technique showed a significant difference in accuracy between most and least distinct features. This may indicate ineffectiveness with this number of features, or with this dataset. Indeed, it may be that for this data, with electrodes measuring brain waves close to each other, patterns can be found both in distinct-input subsets and heavily correlated (similar) subsets. LIME sensitivity analysis showed the best mean accuracy[2] at the most aggressive input selection of 16.

[2] though model *Q* had the highest *median* accuracy.

Table 4. Mean accuracy on test set for neural nets with a range of inputs chosen by each metric, where the most distinct inputs are chosen for each input size. Where relevant, the accuracy is also shown for the same number of *least-distinct* inputs (contra-set accuracy), and the improvement is shown as the difference.

Method	Input size	Accuracy (%)	Contra-set accuracy (%)	Improvement (%)
Full	192	61.11	-	
Q	16	**62.27**	61.96	0.31
	64	**62.36**	61.16	**1.2**
Arctan	16	61.87	62.12	-0.25
	64	62.27	61.29	0.98
LIME	16	**62.34**	61.20	**1.14**
	64	61.41	**62.35**	-0.94
GA1	86	61.57	-	-
GA2	56	**62.15**	-	-

The genetic algorithms showed middling performance despite having more input features available. Against our hypothesis, GA1 only used 86 of the 192 features available. The penalised GA2, with 56 features used, actually performed better than the default, suggesting that it is a suitable fitness function when using genetic algorithms for feature selection. Looking at which features were selected over all generations (Fig. 4) we can see a possible reason why; GA2 is slightly better at exploring the search space, whereas GA1 tends to stick with its parameters for longer. This suggests the penalty term in GA2 also forces exploration even when accuracy is locally optimal. However, they both continued to improve in fitness over the 200 generations, meaning they took a long time and still had suboptimal results, making them less flexible than the other techniques.

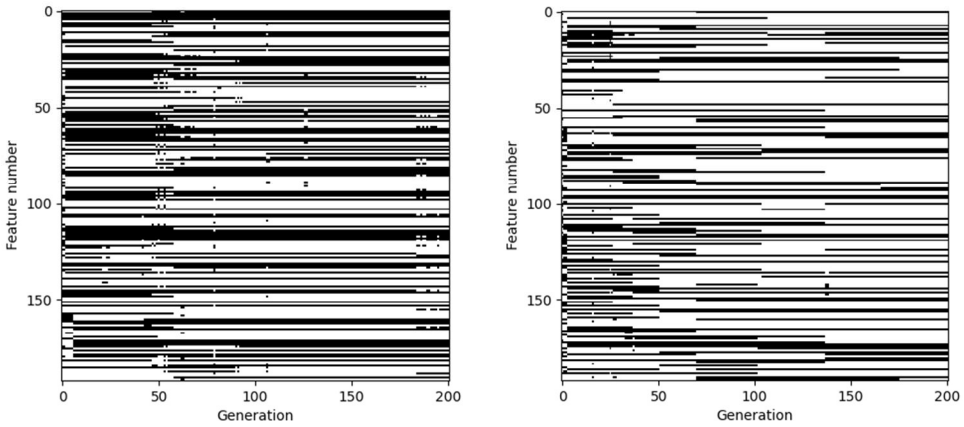


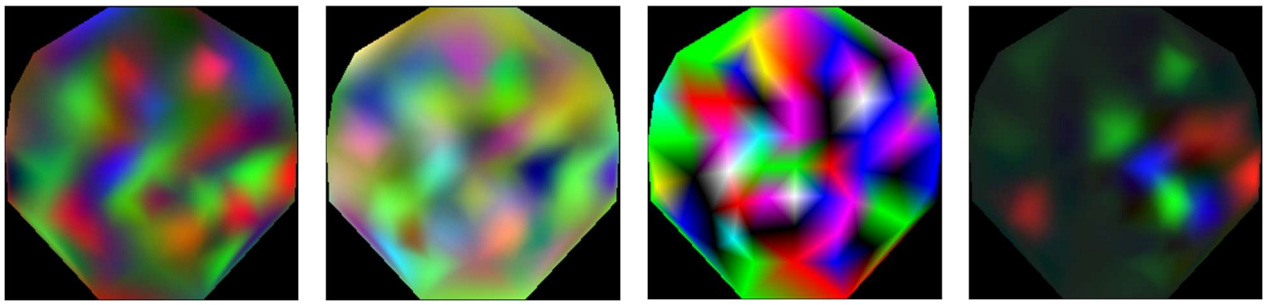
Fig. 4. Input selection over time of GA1 (left) and GA2 (right). Each row of pixels represents a single input feature over 200 generations (left to right). GA2 successfully penalises the number of features within one generation and maintains a smaller input set.

Overall, both Model *Q* and LIME showed the most promise in capturing predictive power on the smallest input subset of 16, though all models performed above the full model baseline, suggesting that for this neural net, the additional inputs add more noise than assistance to classification. Results for the 64-input subset were mixed, suggesting that for larger input numbers, none of these models may be able to effectively summarise the model complexity. A good test of this would be to use the high-dimensional space of image inputs on a convolutional neural net.

We conclude that this dataset cannot easily be summarised by a single subset of features, and that input distinctiveness doesn't necessarily imply prediction power on this data. It appears that there are multiple similar predictive models that the neural net can find, depending on the data available. While genetic algorithms were slower and less effective, we did find that when used, the use of a penalised fitness function is desirable.

3.3 Visualisations

The most promising results were found when converting these models to visual representations for the purposes of model explanation. Model *Q*, GA2 and the mean LIME score across the test set are compared with a random alcoholic trial in Fig. 5.



Random alcoholic trial | Model *Q* | GA2 (penalised) | LIME

Fig. 5. Projections of features onto a 2D space (red: theta waves, green: alpha waves, blue: beta waves). From left to right, (a) a random alcoholic trial, (b) Model *Q* input scores, (c) GA2-penalised inputs, (d) mean of LIME input scores on test set. Note that the GA can only distinguish features as on or off, so limited by being sharper and brighter.

We see the LIME visualisation is particularly effective as it provides a small number of distinct inputs that might help to explain a causal mechanism for differing EEG readings of alcoholics. Interestingly, stronger inputs are distinctly red-green-blue coloured, indicating that by this interpretation, each electrode shows a significant input primarily at only one frequency, whereas the mixed-tone model *Q* visualisation implies significant inputs are shared between frequencies.

3.4 Discussion and future work

Our goal here was to evaluate different neural network hyperparameter tuning and input analysis methods, but with over 4,000 neural networks built on the same architecture, using approximately 15 hours of training time, clearly some of these approaches are not feasible for many use cases.

It is worth noting that the random data splits on the small number of subjects caused high variance in the resulting accuracy. This was successfully mitigated by the regularisation technique used in evolutionary algorithms of incorporating AUC. It also outperformed grid search and converged within 30 generations (180 models built) vs 432 built on grid-search over a much smaller search space. A future optimization could be to use an initial population randomised around best-practise defaults for even further improvement.

We could also compare this performance to other hyperparameter optimization techniques such as random search (Bergstra & Bengio 2012, scikit-learn 2020b) and Bayesian optimization (Snoek et al. 2012).

This paper did not have an applied predictive goal for the dataset, so use of AUC for regularisation was a suitable scale-invariant decision. For specific prediction tasks on an unbalanced dataset, such as detecting early-stage alcoholism from the general population, use of false positive or false negative penalties may also be effective. Additionally, generalisation may be improved by techniques such as pruning of hidden layers by distinctiveness, reusing the same angle measurement techniques described above for input pruning.

While it was not our aim here, better predictive results are of course obtained by using the full EEG signal and encoding using a deep learning approach, such as autoencoders for feature selection and convolutional neural nets (Yao et al. 2018, Li et al. 2017). In any case, migrating the best techniques found here to a deep neural network is a logical next step, for evaluating an even higher number of features.

Unfortunately, we could not determine any specific inputs in this dataset that were important across all of the input analysis techniques tested here. This may suggest that there isn't a simple way to explain how a neural net can use an EEG to predict alcoholism. While all techniques show good accuracy when used for feature selection, more work is needed to validate this in other datasets. In particular, LIME appears to work better with a smaller number of significant features and less well with a larger proportion of the original feature set.

Nevertheless, the sensitivity analysis using LIME is a promising approach for explaining significant inputs when combined with 2D EEG visualisations. The next step would be to use deep learning techniques that operate directly on the 2D models such as those in (Yao et al. 2018, Li et al. 2017) and apply LIME masking directly to the input images. Use of within-subject visualisations as well as cross-subject visualisations here would help to manage high variance of readings between subjects. These visualisations could be further compared with other explainability techniques such as SHAP (Lundberg & Lee 2017) and Explainable Boosting Machines (Nori et al. 2019). The latter is a white-box model, allowing clear interpretations when inputs are semantically meaningful, and detection and editing of input measurement error. We may then be able to use these ideas in conjunction with pharmacological domain expertise to propose causal models of alcoholism's effect on parts of the brain.

References

1. Yao, Y., Plested, J., Gedeon, T.: Deep feature learning and visualization for EEG recording using autoencoders. International Conference on Neural Information Processing. pp. 554--566. Springer (2018)

2. Yao, Y., Plested, J., Gedeon, T.: Information-preserving feature filter for short-term EEG signals. J. Neurocomputing. 408, 91--99. Elsevier (2020)
3. Gedeon T. Data mining of inputs: analysing magnitude and functional measures. Int J Neural Syst. 8(2):209--218. (1997)
4. Lawhern, V., Solon, A., Waytowich, N., Gordon, S., Hung, C. and Lance, B.: EEGNet: A Compact Convolutional Network for EEG-based Brain-Computer Interfaces. CoRR. abs/1611.08024. (2016)
5. Brown, T. et al.: Language Models are Few-Shot Learners. CoRR. abs/2005.14165. (2020)
6. Begleiter, H.: UCI EEG Dataset. https://archive.ics.uci.edu/ml/datasets/EEG+Database (1999)
7. Snodgrass, J. G., & Vanderwart, M.: A standardized set of 260 pictures: Norms for name agreement, image agreement, familiarity, and visual complexity. Journal of Experimental Psychology: Human Learning and Memory, 6(2), 174--215. (1980)
8. Li, Y., Dzirasa, K., Carin, L., Carlson, D.E.: Targeting EEG/LFP synchrony with neural nets. In: Advances in Neural Information Processing Systems, pp. 4623–4633 (2017)
9. Liu, S., Yao, Y.: EEG feature filter and disguising [accessed 15/05/2021] https://github.com/ShiyaLiu/EEG-feature-filter-and-disguising (2021)
10. Bashivan, P.: EEGLearn [accessed 15/05/2021] https://github.com/pbashivan/EEGLearn (2020)
11. Gedeon, T. D., & Harris, D.: Network reduction techniques. In Proceedings International Conference on Neural Networks Methodologies and Applications. vol. 1, pp. 119-126 (1991)
12. Wong, P. M., Gedeon, T. D. and Taggart, I. J.: An Improved Technique in Porosity Prediction: A Neural Network Approach. IEEE Transactions on Geoscience and Remote Sensing, vol. 33, n. 4, pp. 971--980 (1995)
13. Abdou, A., Higashiguchi, S., Horie, K., Kim, M., Hatta, H., Yokooshi, H.: Relaxation and immunity enhancement effects of γ-Aminobutyric acid (GABA) administration in humans. BioFactors, vol. 26 issue 3 pp. 201--208 (2008)
14. Frohlich, J. et al.: Electrophysiological Phenotype in Angelman Syndrome Differs Between Genotypes. Biological Psychiatry, vol. 85, issue 9, pp. 752--759 (2019)
15. Santhakumar V., Wallner M., Otis T. S.: Ethanol acts directly on extrasynaptic subtypes of GABAA receptors to increase tonic inhibition, [published correction appears in Alcohol. 2007 Sep;41(6):461] Alcohol. 2007;41(3) pp 211--221 (2007)
16. PyTorch: cross-entropy loss implementation [accessed 25/03/2021] https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html (2019)
17. Nair, V., Hinton, G.: Rectified Linear Units Improve Restricted Boltzmann Machines (2010)
18. Xie, Z., Yuan, L., Zhu, Z., Sugiyama, M.: Positive-Negative Momentum: Manipulating Stochastic Gradient Noise to Improve Generalization. CoRR, abs/2103.17182 (2021)
19. PyTorch: torch.optim [accessed 7/05/2021] https://pytorch.org/docs/stable/optim.html (2019)
20. Manning, C., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Ch. 6 The vector space model for scoring. Cambridge University Press (2009)
21. scikit-learn User Guide, section 6.8.1: Cosine similarity [accessed 4/05/2021] https://scikit-learn.org/stable/modules/metrics.html#cosine-similarity (2020)
22. Lundberg, S. & Lee, S.: A Unified Approach to Interpreting Model Predictions. In: Advances in Neural Information Processing Systems 30. Pp 4765--4774. Curran Associates, Inc. (2017)
23. Nori, H., Jenkins, S., Koch, P., Caruana, R.: InterpretML: A Unified Framework for Machine Learning Interpretability. CoRR abs/1909.09223 (2019)
24. LeCun, Y., Bottou, L., Orr, G., Müller, K.: Efficient BackProp. In: Neural Networks: Tricks of the Trade, this book is an outgrowth of a 1996 NIPS workshop. pp. 9--50 (1998)
25. Bergstra, J. & Bengio, Y.: Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research. vol. 13 n. 10 281--305 (2012)
26. scikit-learn User Guide, section 3.2.2: Randomized Parameter Optimization [accessed 6/05/2021] https://scikit-learn.org/stable/modules/grid_search.html (2020)
27. Holland, J.H.: EAs, NP Complete. Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor (1975)
28. Bashivan, P., Rish, I., Yeasin, M. & Codella, N.: Learning Representations from EEG with Deep Recurrent-Convolutional Neural Networks. arXiv preprint: arXiv: 1511.06448 (2015)
29. Erguzel, T., Ozekes, S., Tan, O., Gultekin, S.: Feature Selection and Classification of Electroencephalographic Signals: An Artificial Neural Network and Genetic Algorithm Based Approach. In: Clinical EEG and Neuroscience, vol. 46, iss. 4, pp 321--326. Wheaton (2015)
30. Sharma, N., Gedeon, T.: Computational Models of Stress in Reading Using Physiological and Physical Sensor Data. In: Advances in Knowledge Discovery and Data Mining, pp 111--122. Springer Berlin Heidelberg (2013)
31. Fortin, F., De Rainville, F., Gardner, M., Parizeau, M., Gagné, C.: DEAP: Evolutionary Algorithms Made Easy. In: Journal of Machine Learning Research, vol. 13 pp. 2171--2175 (2012)
32. Rothwell, C.: Hyper-parameter Optimisation Using Genetic Algorithms: Classification Task [accessed 15/05/2021] https://www.linkedin.com/pulse/hyper-parameter-optimisation-using-genetic-algorithms-conor-rothwell/ (2018)
33. Pakt Publishing: Hands-On-Genetic-Algorithms-with-Python [accessed 15/05/2021] https://github.com/PacktPublishing/Hands-On-Genetic-Algorithms-with-Python/blob/master/Chapter09/elitism.py (2021)
34. sklearn: MinMaxScaler https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html [accessed 16/05/2021] (2020)
Johnson, N., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions (2nd ed.) vol. 2, ch. 25: Beta Distributions. Wiley (1995)
35. Tibshirani, R.: Regression Shrinkage and Selection Via the Lasso. Journal of the Royal Statistical Society: Series B (Methodological) vol. 58, iss. 1, pp. 267--288 (1996)
36. Snoek, J., Larochelle, H., Adams, R.P.: Practical Bayesian Optimization of Machine Learning Algorithms. arXiv preprint: arXiv:1206.2944 ()
37. Ribeiro, M., Singh, S., Guestrin, C.: Why Should {I} Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, pp. 1135--1144 (2016)
38. Ribeiro, M. et al: LIME python package [accessed 21/05/2021] https://github.com/marcotcr/lime (2021)