Neural Network Architecture Design for Parkinson Speech Classification

Alex Gould
Masters of Data Science
University of New South Wales
Sydney, Australia
alextgould@gmail.com

***Abstract*—Artificial neural networks are a powerful and flexible machine learning tool, but come with the requirement of selecting an architecture and hyperparameters which are suited to the task at hand. This study takes a systematic approach to the task of training a neural network to classify people between those with Parkinson's Disease and healthy controls. The data used proves to be too small to effectively utilise a neural network approach, with all models significantly overfitting the training data. This paper highlights the need to match the power and flexibility of neural networks with an appropriately sized dataset, before spending time trying to optimise the network.**

***Index Terms*—neural network architecture, hyperparameter tuning, minimum sample size, Parkinson's disease**

I. Introduction

Artificial neural networks are a family of machine learning techniques that have advanced the state of the art across multiple domains in recent years [1]. While these techniques can be both powerful and flexible, this comes with the potential cost of having to make decisions about network architecture, and the associated tuning of hyperparameters, in order to optimise the network's performance [2].

In this study, a systematic approach was taken to tune a neural network to classify people into People with Parkinson's Disease (PWP) or Healthy Controls (HC), based on vocal features extracted from voice samples. Parkinson's Disease (PD) is a common neurodegenerative disease, and one which might be identified through applying machine learning techniques to vocal data rather than requiring the in-person presence of the patient and human expertise [5]. This task replicates some of the work done in a previous study ("original study") [3], for which the data is available on the UCI Machine Learning Repository, but uses neutral networks in place of k-NN and SVM methods. [4].

A range of sequential neural networks were considered, with varying parameters such as the number of hidden layers and neurons per layer, learning rate, optimizer, activation function, initialisation strategy and regularisation strategies. Experiments were evaluated using a range of metrics, including binary cross entropy loss, prediction accuracy, Mathew's correlation coefficient, run time, AUC, ROC curves and confusion matrices. Each experiment was repeated 30 times, with mean and 95% confidence intervals being used to compare experiments.

II. Problem Definition and Algorithms Used

A. Task Definition

The training data belongs to 40 participants, of whom 20 are PWP and 20 are HC. Each subject has 26 voice samples, belonging to one of a few types of tasks, ranging from making single sustained vowel sounds, to saying numbers, words and short sentences. For each voice sample, 26 features are extracted. The task is to predict whether a voice sample was taken from a PWP or HC subject; that is, this is a binary classification task.

Fig. 1 shows the correlations within the training data, highlighting the related nature of some of the features. The first five features all relate to jitter, the next relate to shimmer and so on. For some models, such correlation would be concerning and it would be necessary to apply techniques such as principle component analysis or drop some of the correlated features. However, neural networks are generally able to handle such multicollinearity due to their redundant architecture [6].

Fig. 2 shows summary plots for the first feature, jitter (local). The first subplot highlights a positive relationship between this feature and PWP subject proportion. The second panel shows the distribution of numbers of participants, while the final panel provides a boxplot which again indicates PWP tend to have higher levels of this feature. Such plots were produced for all features (see Appendix A). PWP tend to have higher levels of jitter and shimmer (across the various related features), lower pitch, higher period and fewer unvoiced frames and voice breaks. These findings are consistent with the literature (e.g. [7]).

In addition to the features provided, an additional feature was added which captured the type of voice sample, being an "a vowel", "o vowel", "u vowel", "numbers", "sentences" or "words". These categories were one hot encoded.

Finally, a standard scaler was applied to all features, so that they had zero mean and unit variance. Gradient descent optimization in particular can be quite sensitive to differences

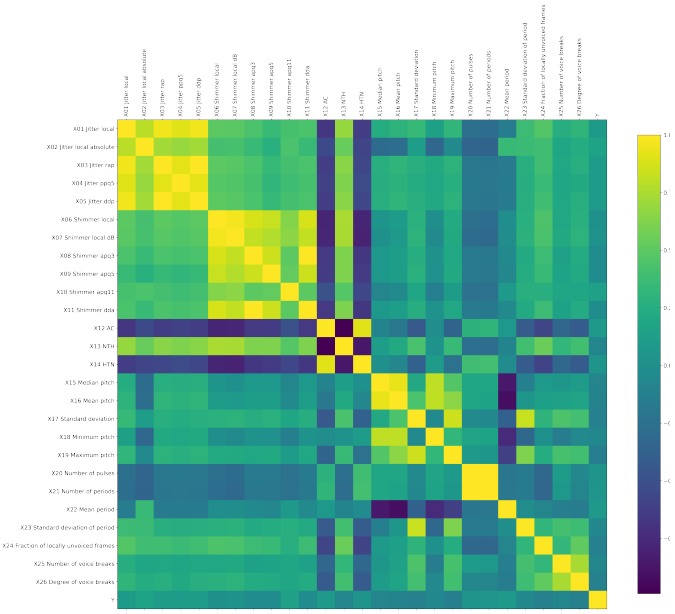


Fig. 1. Training data correlation matrix

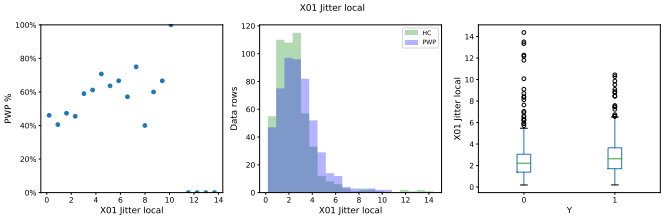


Fig. 2. Plots of PWP by Jitter local

in feature scale, and failing to standardize features may lead to slower convergence [8].

B. Algorithm Definition

A range of sequential neural networks were considered, with varying hyperparameters. The networks were implemented in tf-keras, primarily using regular densely-connected layers.

Two optimizers were considered: Adam and stochastic gradient descent (sgd). As an adaptive optimizer, which keeps track of an exponentially decaying average of past gradients and squared gradients, it was anticipated that Adam would perform better at different learning rates and generally run faster, where sgd would require a more precise learning rate and be slower, particularly when momentum was not used. For sgd, different levels of momentum were considered. It was expected that the use of momentum would result in faster convergence.

Two activation functions were considered: relu and selu. Given that the networks used consist entirely of densely-connected layers, the use of selu allows the network to self-normalize, which avoids vanishing/exploding gradients [2]. For this reason, selu was expected to perform better for deeper networks. The initializers used for these activation functions were He and Le Cun Normal respectively. These are recommended in [2] to avoid vanishing/exploding gradients, speed up training and ensure the self-normalising guarantee of the selu activation.

Three regularisation strategies were considered:

- One of the best regularisation techniques is early stopping [2]. A series of experiments were run with with early stopping, up to 1,000 epochs. Based on the number of epochs used in these experiments, experiments were run without early stopping, with epochs varying between 5 and 50.
- l2 regularisation restricts the size of the weights in the network, which makes it harder for the network to overfit as only parameters which significantly reduce the objective function are preserved relatively intact [9]. Models were run with l2 regularisation factors between 1e-3 and 5e-1 (as well as with no l2 regularisation).
- Alpha dropout layers were added before the hidden Dense layers, with up to 40% of the hidden neurons being dropped, to encourage the network to avoid relying too heavily on individual neurons.[1]

III. Experimental Evaluation

A. Methodology

The data for the 40 participants was split into three sets. The first set ("train") used 40% of the data (16 participants), and was used by the neural network to fit weight parameters and to fit the standard scaler. The second set ("valid") used 20% of the data (8 participants), and was used by the network to determine when to stop training when using early stopping. The third set ("test") used 40% of the data (16 participants) and was used as a hold out set to evaluate the model's expected performance on unseen data. A group shuffle split approach was used to ensure all records for a given participant fall into one of these three sets. This step prevents the network from "cheating" by finding ways to identify specific participants using features which are common to the participant's vocal samples but which are not actually relevant to PD.

26 experiments were conducted, with each having a primary hyperparameter of interest which was was varied while others were held constant. The default hyperparameters were 1 hidden layer, 5 neurons per layer, a learning rate of 3e-3, Adam optimizer, early stopping (up to 1000 epochs), relu activation and He initialization. For several experiments, the parameter of interest was varied both for a "shallow" network, which follows the default parameters above, as well as a "deep" network, which used 3 hidden layers and 15 neurons per layer, but was otherwise identical to the "shallow network". This

[1]In hindsight, it would have been preferable to use regular dropout with the relu activation and only use Alpha dropout with the selu activation.

approach aims to capture some of the interaction between different hyperparameters, particularly the size of the network. It was confirmed that the default learning rate was generally appropriate in both cases (and hence for other experiments as a baseline).

Each experiment was repeated 30 times, with mean and 95% confidence intervals being used to compare experiments. This was particularly important given the relatively small size of the dataset; results for individual models within a given experiment varied significantly, based on the randomness in the training procedure. Random seeds were set such that the experiments overall were reproducible, but with seeds varying for each iteration of each experiment.

A range of metrics were considered. The models were optimised using binary cross entropy loss, and this was compared across levels of the varying hyperparameter of interest. Prediction accuracy and Mathew's correlation coefficient were also key metrics, with the former being a popular metric while the latter is considered a more reliable metric as it requires the model to do relatively well throughout the confusion matrix categories [6]. Additional metrics were run time, ROC curves and AUC, confusion matrices and learning curves.

B. Results

More detailed results are shown in this section for two of the experiments, based on their difference in approach and relatively good performance on validation and test sets. In particular, experiment 1 (shallow network with default parameters) fared relatively well on the valid set, while experiment 26 (2 layer network, no early stopping, dropout) fared relatively well on the test set (by a small margin).

Fig. 3 shows the key metrics plots for experiments 1 and 26. These plots show the mean and 95% confidence intervals for cross entropy loss, accuracy and MCC for the train, valid and test sets, as well as run times. Such plots were produced for all experiments (see Appendix B).

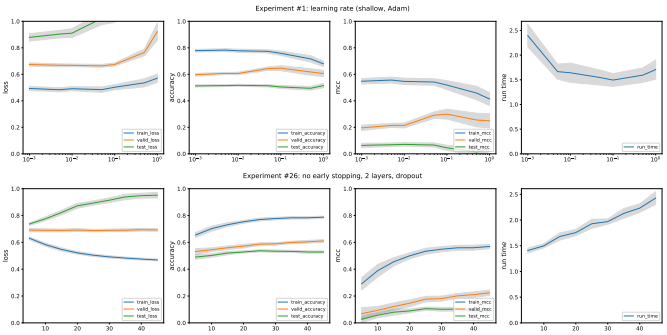


Fig. 3. Results plots for selected experiments

Table I summarises experimental observations for all 26 experiments.

TABLE I. Experimental Observations

Experiments	**Observations**
Exp 1-4 examined learning rates for shallow and deep networks, and compared the Adam and sgd optimizers.	Smaller learning rates take longer but converged to similar performance levels. The default learning rate of 3e-3 was appropriate for both the shallow and deep networks. Adam was much faster than sgd at lower learning rates.
Exp 5-8 examined learning rates for shallow and deep networks without using early stopping, at 10 and 20 epochs.	While the training loss appears to be higher than when using early stopping, the test MCC was generally higher (better), particularly with learning rates between 1e-2 and 1e-1. Deep networks with high learning rates (1+) performed particularly poorly. Run time was relatively constant, independent of the learning rate used.
Exp 9 and 10 increased the number of hidden layers and neurons per hidden layer.	Results were relatively stable regardless of changes to these parameters, with some evidence of lower (worse) MCC as more layers were added. This may indicate that even the shallow base model is already over-fitting the data.
Exp 11 and 12 used a selu activation instead of relu.	Train and valid results seemed to increase with selu, however test results decreased. This may indicate selu was associated with increased over-fitting. Run time was similar between selu and relu.
Exp 13 and 14 added l2 regularization.	Adding some l2 regularization (1e-3 to 1e-2) led to an increase in accuracy and MCC, particularly for the deep network which was likely over-fitting to a greater extent. Higher levels (1e-1 and above) led to a significant drop in results. Adding l2 regularization increased run times, with higher levels of regularization associated with greater increases in run times.

Experiments	Observations
Exp 15 and 16 added Alpha Dropout.	Accuracy and MCC generally reduced as greater dropout was applied.
Exp 17 and 18 added momentum to the sgd optimizer.	Momentum had very little impact, aside from a significant reduction in run time at large levels (1e-1 to 1e-0)
Exp 19-22 considered even smaller networks, with 1-5 neurons per layer for 1-2 layers with or without dropout.	In each case, having 5 neurons resulted in better accuracy and MCC compared to having 1-4 neurons.
Exp 23-26 looked at networks with no early stopping at a range of epochs (5-50)	Test set accuracy and MCC generally plateaued between 20-30 epochs, for networks with 1-2 layers with or without dropout. Run time increased linearly with epochs.

C. Discussion

Overall, most experiments followed a similar trend, with high performance on the training set, moderate performance on the validation set and low performance on the train set. Fig. 4 compares the AUC across all experiments and supports this observation. This suggests that regardless of the hyperparameters selected, the neural network methodology is inclined to overfit the training data, and to a lesser extent the validation data used for early stopping.

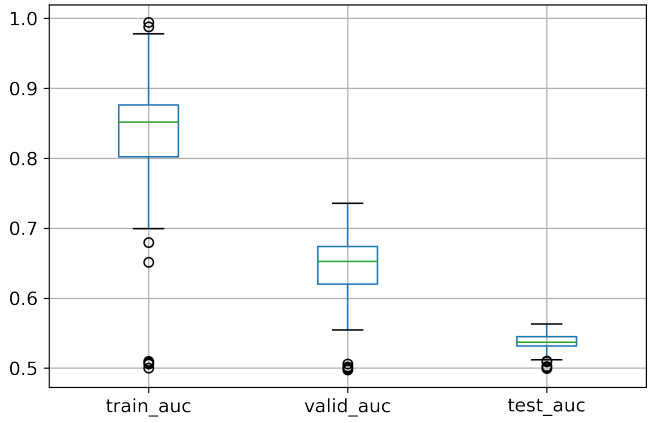


Fig. 4. Train, valid and test AUC scores across all experiments

Fig. 5 and 6 show learning curves for an individual level of experiments 1 (learning rate 0.05) and 26 (30 epochs), with mean and 95% confidence intervals based on 30 iterations. The valid set loss is stable or increasing from a very low number of epochs, which confirms that the models are overfitting.

Fig. 7 and 8 show the corresponding confusion matrices for these two experiment levels, averaged across 30 iterations. As

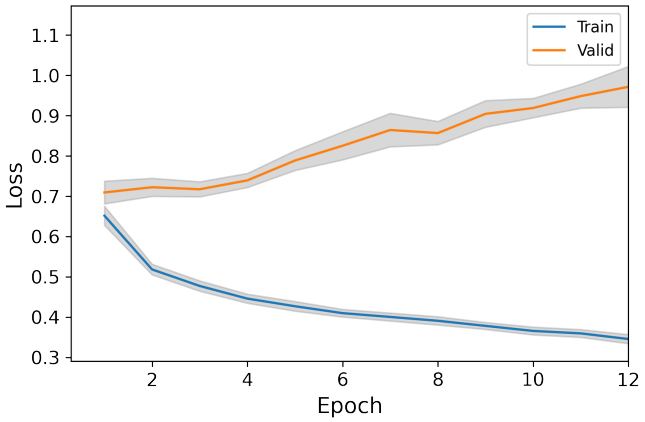


Fig. 5. ROC curves for experiment 1

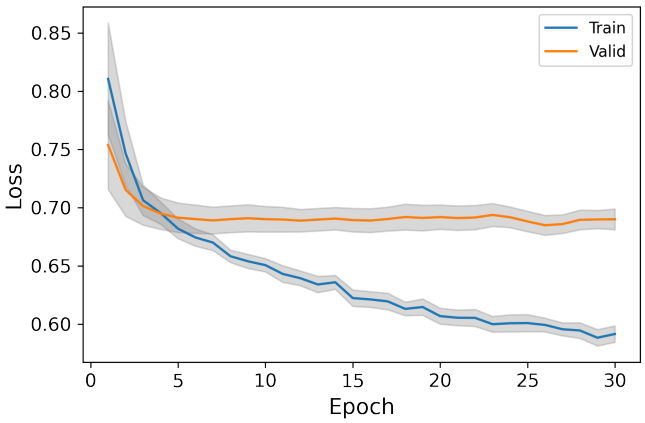


Fig. 6. ROC curves for experiment 26

with the AUC scores, there is a noticeable drop in performance in both cases, from train to valid to test, particularly for accuracy and recall. MCC was used with preference to confusion matrices in this study as it more readily allows a large number of similar models to be compared, and captures much of the information embedded within the confusion matrix.

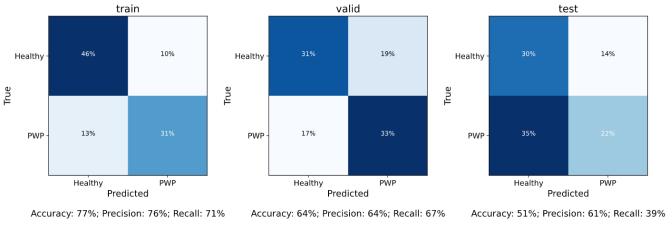


Fig. 7. Confusion matrices for experiment 1

Fig. 9 and 10 show the corresponding ROC curves for these two experiment levels, with 95% confidence intervals shown around the mean. AUC was used with preference to ROC curves as as it more readily allows a large number of similar models to be compared, and captures much of the information

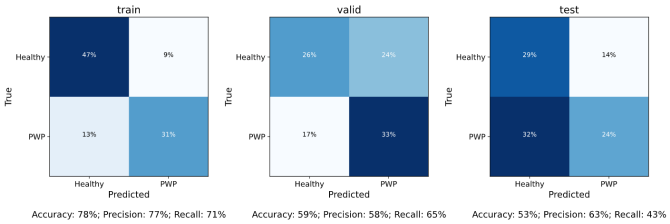


Fig. 8. Confusion matrices for experiment 26

embedded within the ROC curve. In general, ROC curves are more relevant if we have a particular preference for sensitivity or specificity and are considering where to set the threshold along the curve [11].

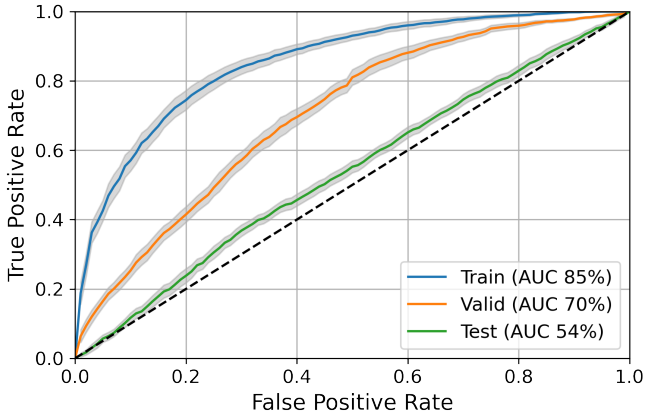


Fig. 9. ROC curves for experiment 1

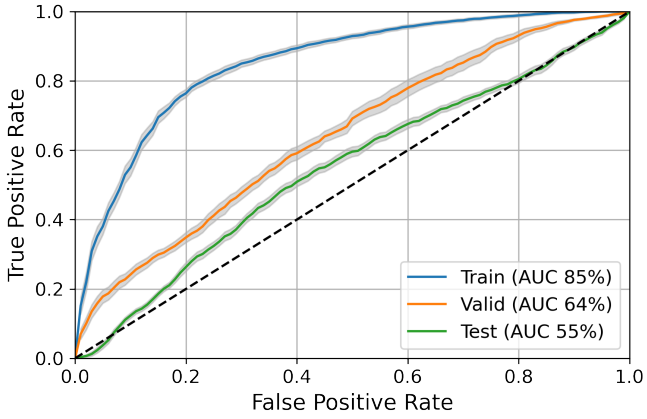


Fig. 10. ROC curves for experiment 26

Fig. 11 and 12 show the ROC curves again, having reshuffled the data. This gives some indication of the extent to which the random allocation between train, valid and test sets is impacting results. The impact on experiments 1 and 26 is similar, with both experiments showing lower train AUC and higher test AUC. That both experiments responded in a similar manner suggests some participants are harder to classify than others, and the random split of the data between train, valid and test sets is contributing to model performance evaluations. Hence, despite using repeated experiments to draw more robust conclusions, this has not addressed all sources of randomness in the results.

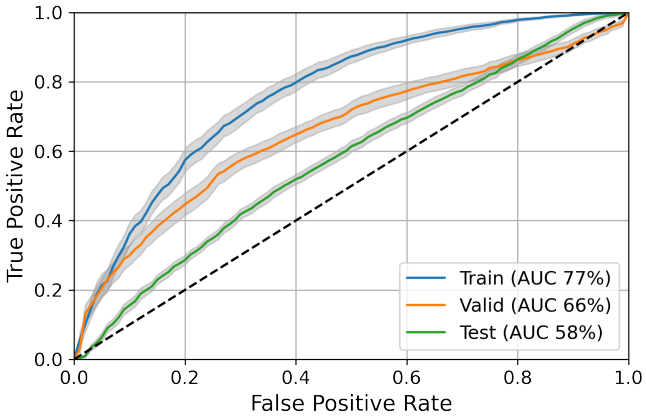


Fig. 11. ROC curves for experiment 1 with reshuffled data

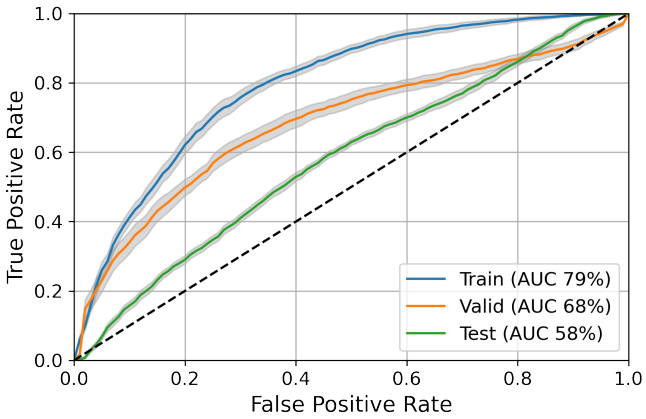


Fig. 12. ROC curves for experiment 26 with reshuffled data

In an attempt to addressing the overfitting evident in the results, PCA was applied to the features to reduce the number of features that could be used to identify individual rows of data in the train set. The amount of data required to fit a neural network is, in part, proportional to the number of weights in the model [12]; a smaller input layer will lead to lower weights and hence may reduce the propensity of the network to overfit. Fig. 13 and 14 show the corresponding ROC curves. While there is less evidence of overfitting when applying PCA, performance has dropped across the train, valid and test sets.

While additional data could not be sourced, it is possible to use more of the data for training. Fig. 15 shows the ROC curve for experiment 26 where 80% of the data is used for training and 20% of the data is used as a hold out test set.

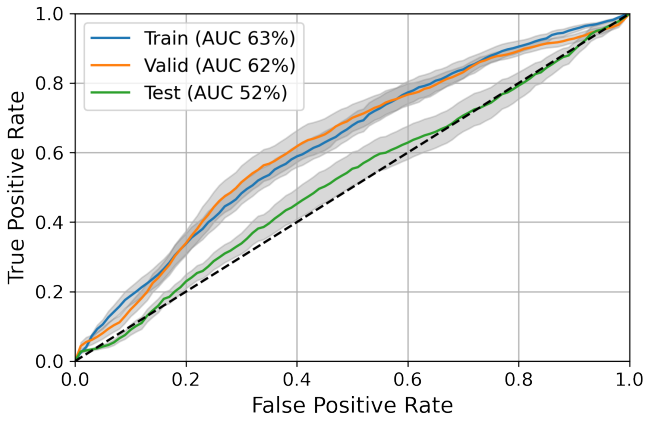


Fig. 13. ROC curves for experiment 1 using PCA

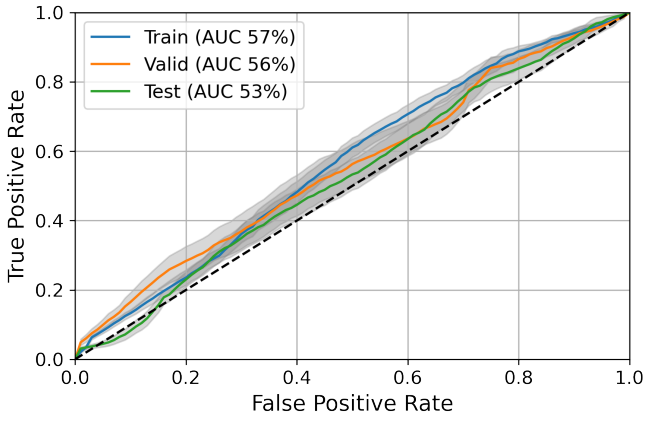


Fig. 14. ROC curves for experiment 26 using PCA

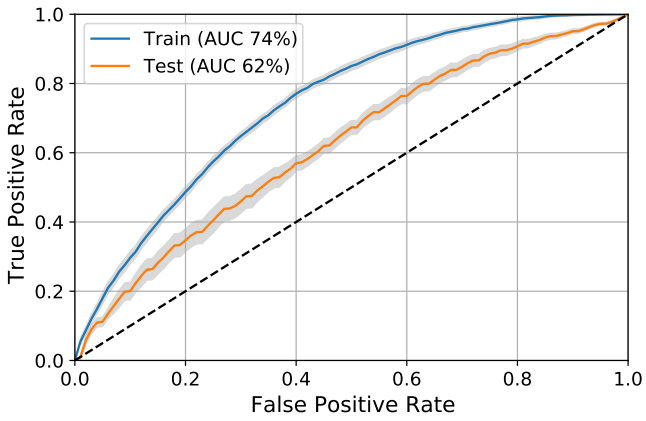


Fig. 15. ROC curves for experiment 26 using 80% of data for training

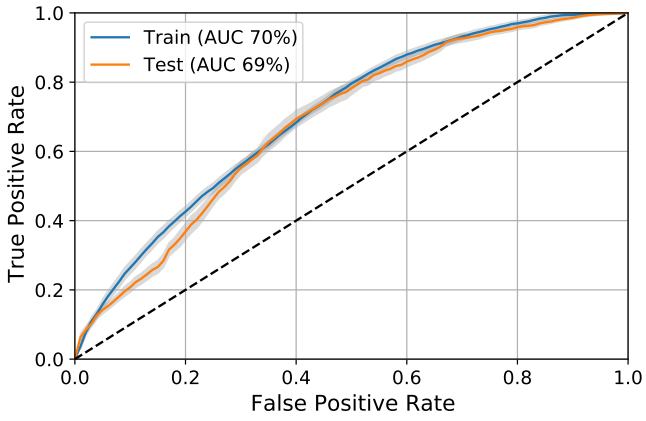


Fig. 16. ROC curves for experiment 26 using a naive train-test split instead of a group split

This experiment was used as no validation set is needed when early stopping is not being used. There is a clear increase in the AUC to a level not seen in the experiments conducted using just 40% of the data for training.

As a final remark, Fig. 16 shows the ROC curve for experiment 26 where the observations for individual participants are allowed to spread between the train and test sets. The difference is remarkable, with test set performance on par with the training set. This result seems to confirm the commentary in section III-A that, given the chance, the network will "cheat" by trying to identify the participants rather than figuring out whether they actually have PD.

IV. Conclusion

Neural networks are a powerful and flexible machine learning tool, but come with the requirement of selecting an architecture and hyperparameters which are suited to the task at hand. This study considered such choices in the context of the task of classifying voice samples between PWP and HC. The neural network significantly overfit the training data, with far worse performance on the hold out test set. This occurred regardless of the choice of architecture and hyperparameters. Attempts to address this using smaller networks, various types of regularisation and dimensionality reduction applied to features, were largely unsuccessful.

This is most likely driven by the relatively small volume of training data used, and the potential for the neural network to "memorise" the training data examples. [12] notes a widely used rule-of-thumb for minimum sample size is ten times the number of weights in the ANN (although their modelling actually suggests fifty times is more appropriate). The number of weights varies based on the network architecture, but the networks used in this study have around 450 weight parameters[2], implying implying a required sample size of ~4,500. In contrast, for this study there were 1,040 observations, and only 40% was used for training the network (~400 observations). Even with a PCA model reducing the number of inputs from

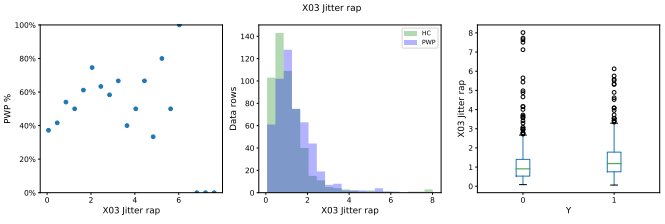
[2]A network with 32 input features, two hidden layers with 10 neurons per layer and a single output has (32+1)*10 + (10+1)*10 + (10+1)*1 = 451 weights.

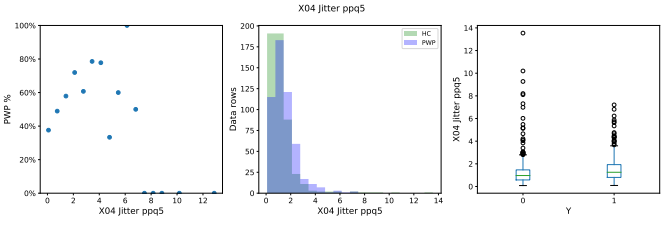
32 to 2, the recommended observations would be ~1,500; several times greater than the 400 used for training. Consistent with this analysis, there was an observed improvement in results when more of the data was used for training rather than validation or hold out testing.

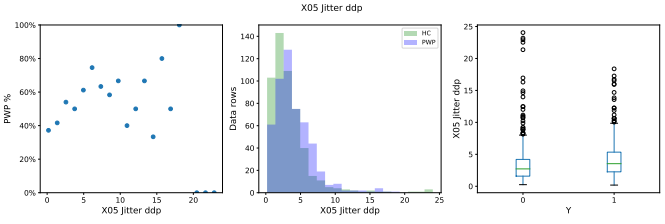
The expected outcomes of experiments were somewhat in line with expectations (e.g. faster run time and less reliance on a specific learning rate for Adam over sgd). However, in addition to the overfitting issue, the task did not necessitate a particularly deep network where exploding/vanishing gradients or training time would be a significant concern, and hence in many cases the impact of any hyperparameter choice was less evident.

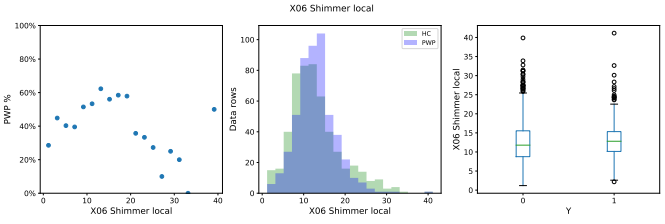
Repeated experiments were successful in providing more robust estimates of classifier performance, despite the relatively small dataset size, to allow the impact of hyperparameter selection to be assessed more reliably. In particular, while the results of individual experiments varied significantly, by repeating each point 30 times, the confidence intervals around the mean were relatively narrow. On the other hand, the impact of randomness in the initial split of the data between train, valid and test sets was not addressed, and did seem to have some impact on results. More importantly, the significant overfitting makes it hard to place much weight in the findings of the individual experiments. Perhaps the main outcome of this study is to highlight the importance of matching the power and flexibility of neural networks with an appropriately sized dataset, before spending time trying to optimise the network.

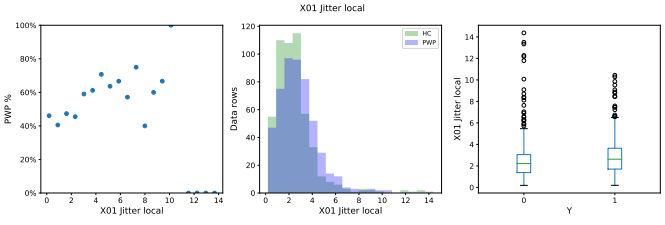
V. Appendix A

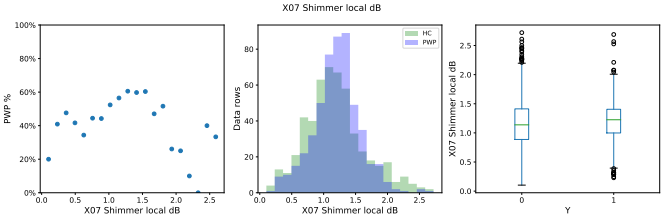


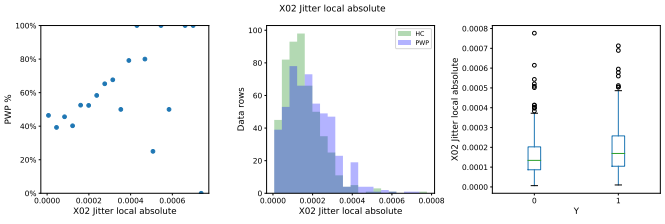


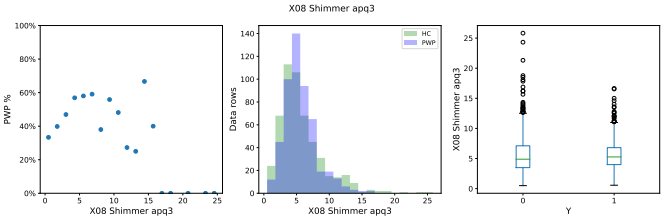


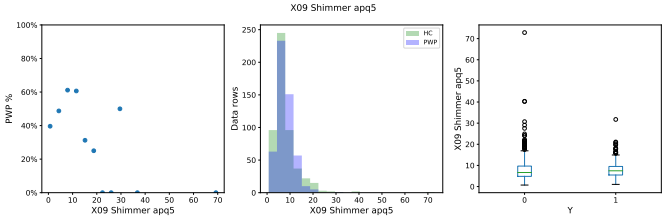


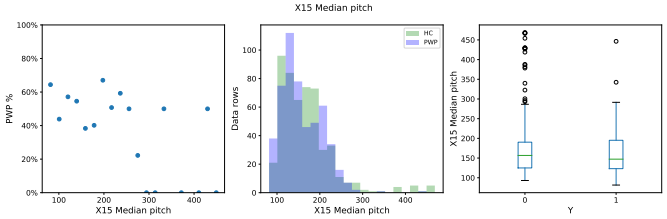


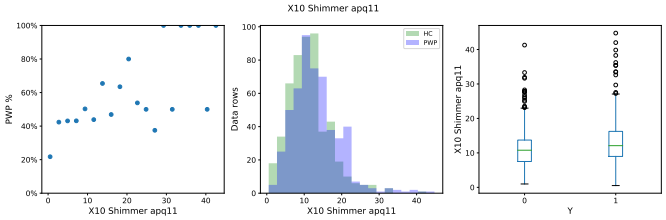


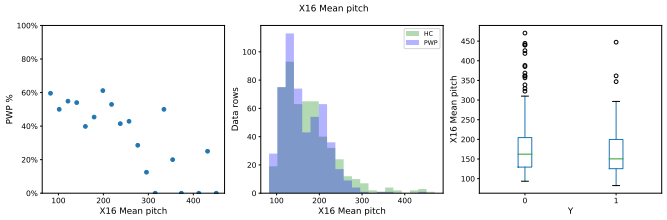


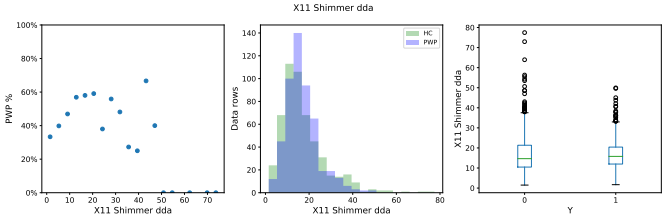


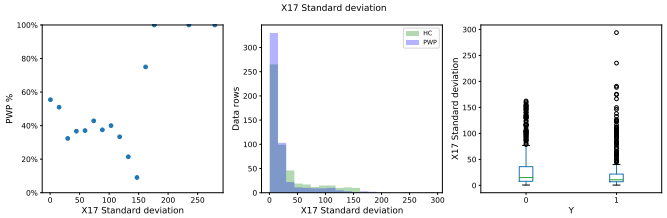

X09 Shimmer apq5
PWP %
X09 Shimmer apq5
Data rows
HC
PWP
X09 Shimmer apq5
Y

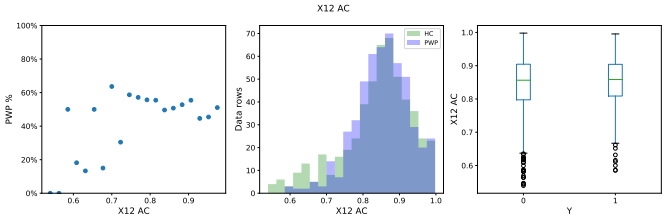

X15 Median pitch
PWP %
X15 Median pitch
Data rows
HC
PWP
X15 Median pitch
Y

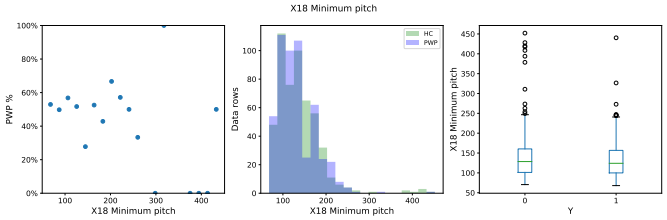

X10 Shimmer apq11
PWP %
X10 Shimmer apq11
Data rows
HC
PWP
X10 Shimmer apq11
Y

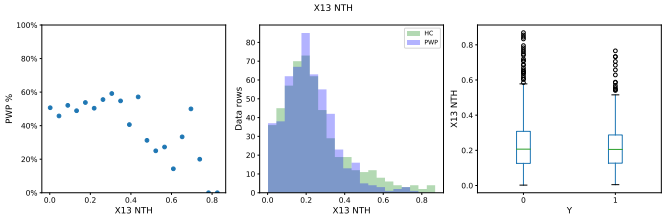

X16 Mean pitch
PWP %
X16 Mean pitch
Data rows
HC
PWP
X16 Mean pitch
Y

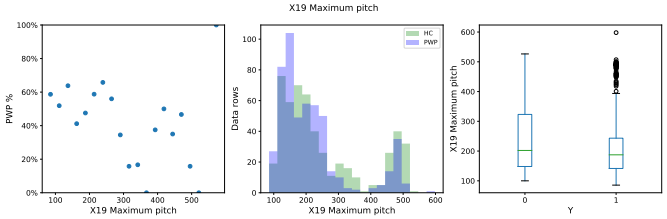

X11 Shimmer dda
PWP %
X11 Shimmer dda
Data rows
HC
PWP
X11 Shimmer dda
Y

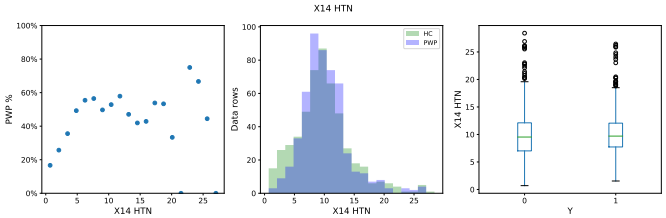

X17 Standard deviation
PWP %
X17 Standard deviation
Data rows
HC
PWP
X17 Standard deviation
Y

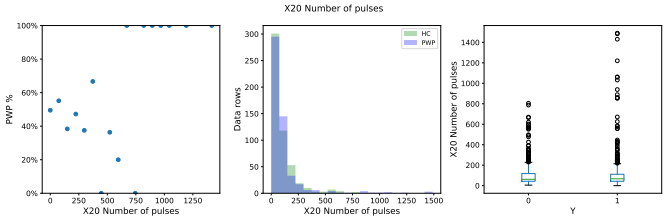

X12 AC
PWP %
X12 AC
Data rows
HC
PWP
X12 AC
Y


X18 Minimum pitch
PWP %
X18 Minimum pitch
Data rows
HC
PWP
X18 Minimum pitch
Y

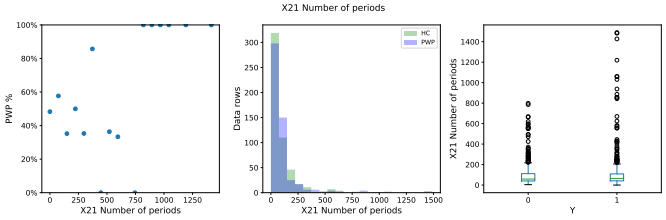

X13 NTH
PWP %
X13 NTH
Data rows
HC
PWP
X13 NTH
Y

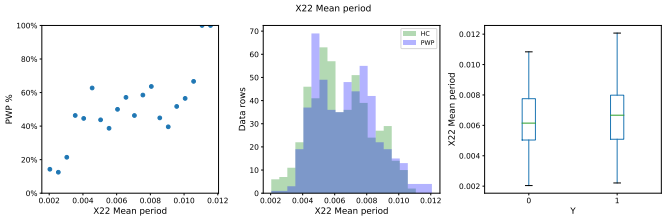

X19 Maximum pitch
PWP %
X19 Maximum pitch
Data rows
HC
PWP
X19 Maximum pitch
Y

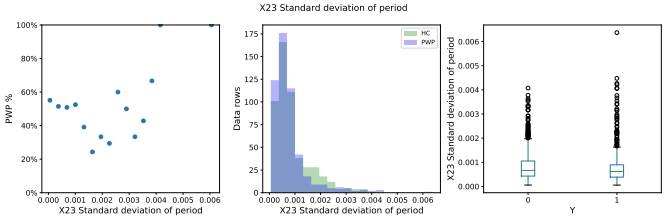

X14 HTN
PWP %
X14 HTN
Data rows
HC
PWP
X14 HTN
Y

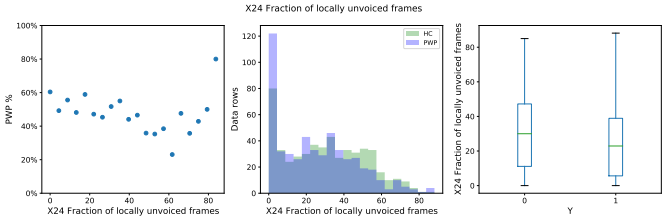

X20 Number of pulses
PWP %
X20 Number of pulses
Data rows
HC
PWP
X20 Number of pulses
Y

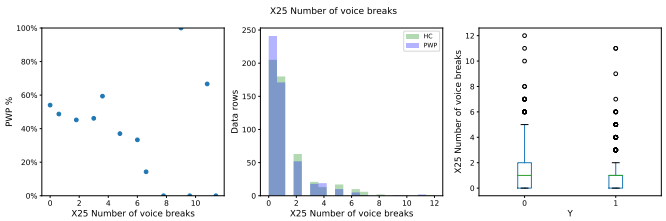
VI. Appendix B

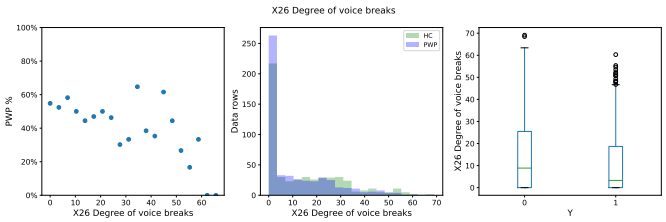


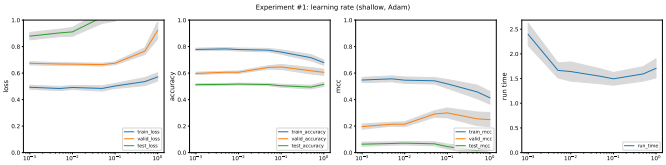


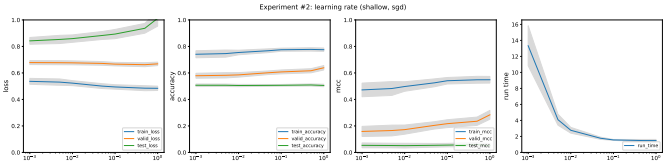


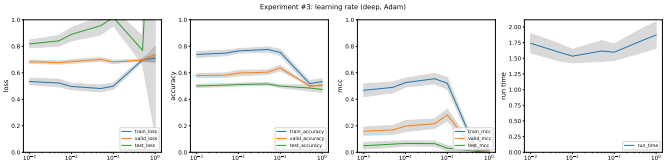


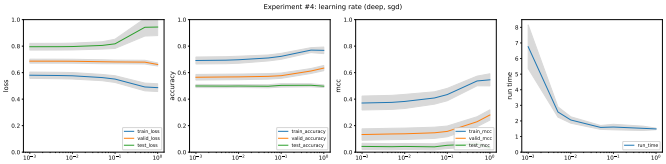


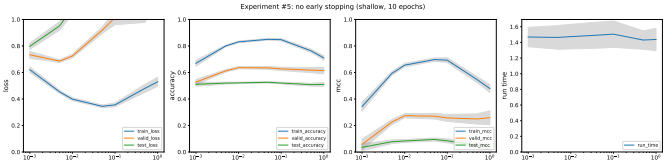


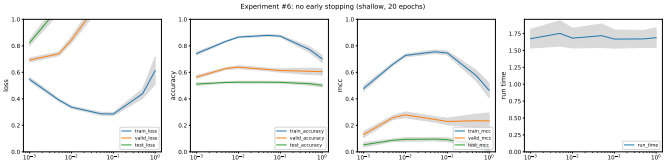


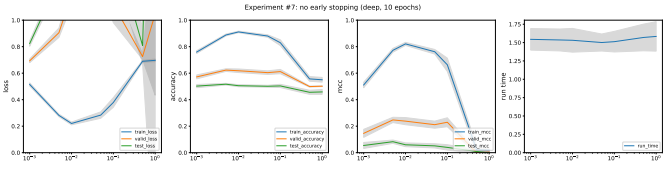


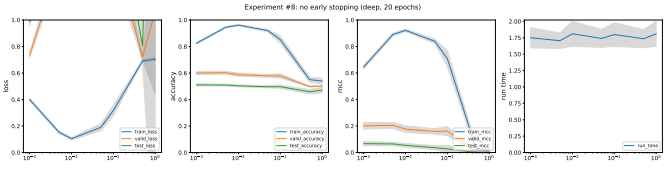


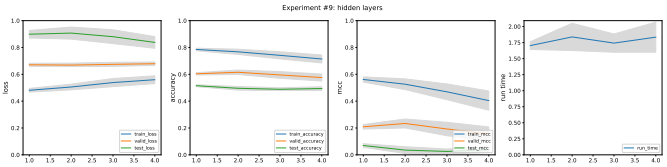


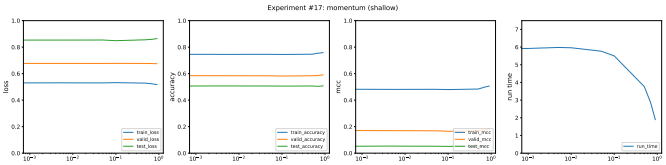


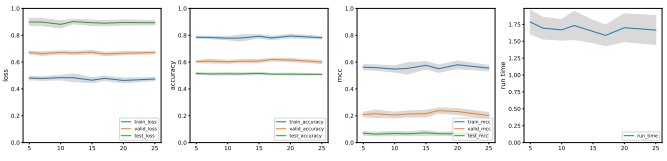


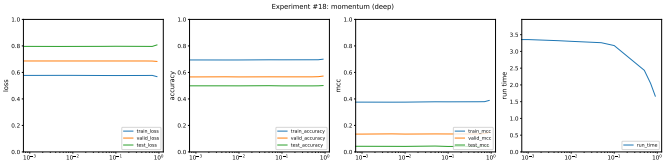


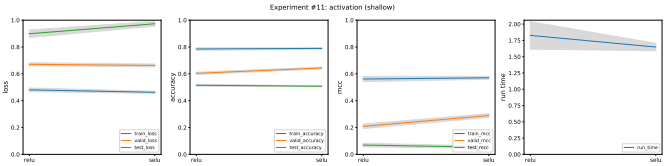


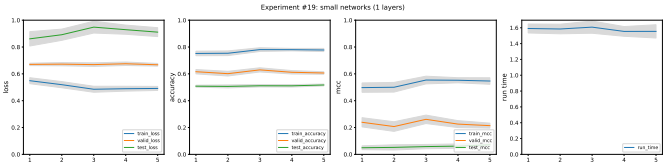

Experiment #9: hidden layers
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

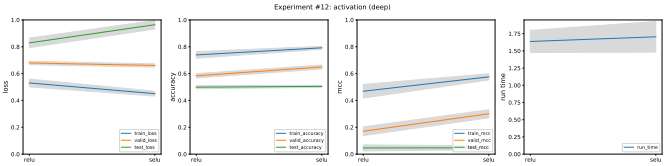

Experiment #17: momentum (shallow)
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

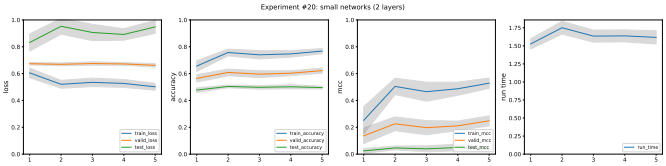

Experiment #10: hidden neurons
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

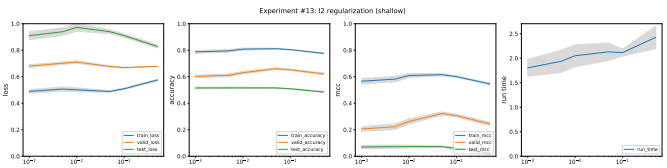

Experiment #18: momentum (deep)
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

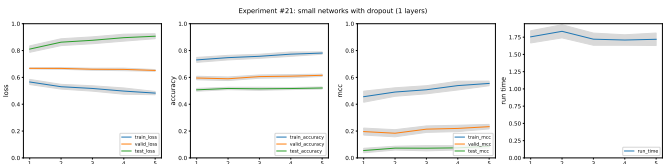

Experiment #11: activation (shallow)
loss
accuracy
mcc
run time
relu
selu
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

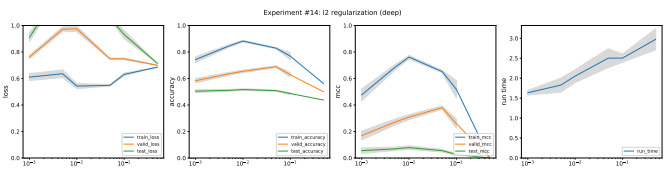

Experiment #19: small networks (1 layers)
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

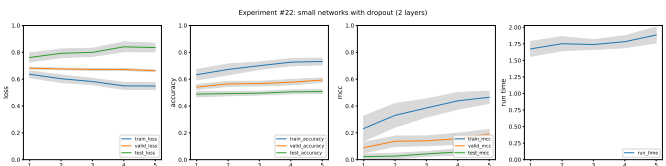

Experiment #12: activation (deep)
loss
accuracy
mcc
run time
relu
selu
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

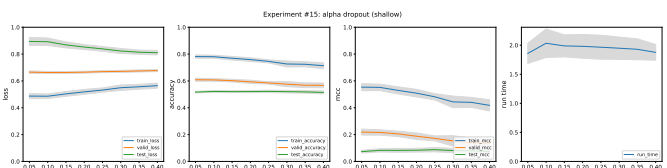

Experiment #20: small networks (2 layers)
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

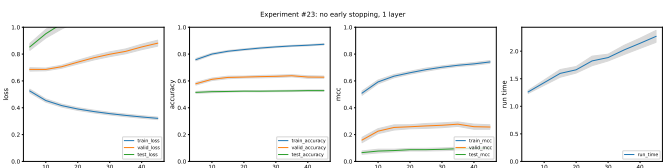

Experiment #13: l2 regularization (shallow)
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

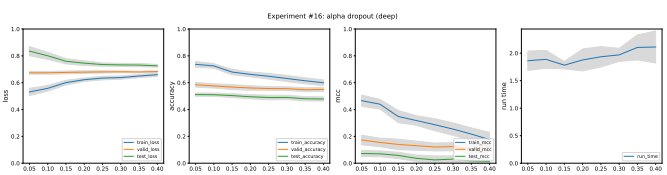

Experiment #21: small networks with dropout (1 layers)
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

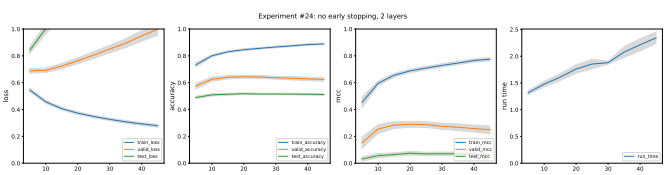

Experiment #14: l2 regularization (deep)
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time

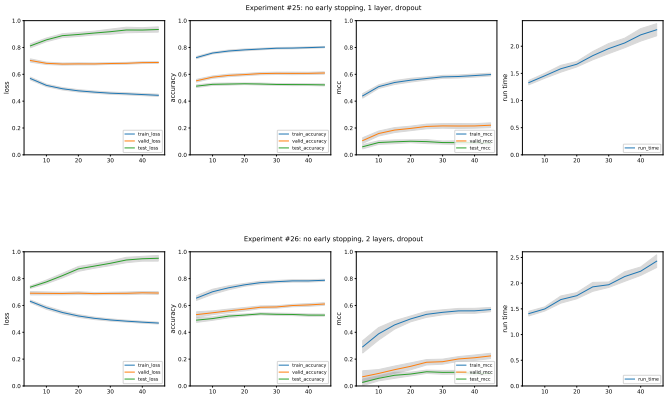

Experiment #22: small networks with dropout (2 layers)
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time


Experiment #15: alpha dropout (shallow)
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time


Experiment #23: no early stopping, 1 layer
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time


Experiment #16: alpha dropout (deep)
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time


Experiment #24: no early stopping, 2 layers
loss
accuracy
mcc
run time
train_loss
valid_loss
test_loss
train_accuracy
valid_accuracy
test_accuracy
train_mcc
valid_mcc
test_mcc
run_time



REFERENCES

[1] "Artificial neural network", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neural_network. [Accessed: 27- Sep- 2020].

[2] Géron, Aurélien. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media, 2019.

[3] Sakar, B., Isenkul, M., Sakar, C., Sertbas, A., Gurgen, F., Delil, S., Apaydin, H. and Kursun, O., 2013. Collection and Analysis of a Parkinson Speech Dataset With Multiple Types of Sound Recordings. IEEE Journal of Biomedical and Health Informatics, 17(4), pp.828-834.

[4] "UCI Machine Learning Repository: Parkinson Speech Dataset with Multiple Types of Sound Recordings Data Set", Archive.ics.uci.edu, 2020. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Parkinson+Speech+Dataset+with++Multiple+Types+of+Sound+Recordings. [Accessed: 27- Sep- 2020].

[5] S. Arora, L. Baghai-Ravary and A. Tsanas, "Developing a large scale population screening tool for the assessment of Parkinson's disease using telephone-quality voice", The Journal of the Acoustical Society of America, vol. 145, no. 5, pp. 2871-2884, 2019. Available: 10.1121/1.5100272.

[6] R. De Veaux and L. Ungar, "Multicollinearity: A tale of two nonparametric regressions", Selecting Models from Data, pp. 393-402, 1994. Available: 10.1007/978-1-4612-2660-4_40 [Accessed 2 October 2020].

[7] J. Holmes, Rhonda, Jennifer M. Oates, Debbie J. Phyland, and Andrew J. Hughes. "Voice characteristics in the progression of Parkinson's disease." International Journal of Language & Communication Disorders 35, no. 3 (2000): 407-418.

[8] "comp.ai.neural-nets FAQ, Part 2 of 7: Learning", Faqs.org, 2002. [Online]. Available: http://www.faqs.org/faqs/ai-faq/neural-nets/part2/. [Accessed: 02- Oct- 2020].

[9] I. Goodfellow, Y. Bengio and A. Courville, "Deep Learning", MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org. [Accessed: 1 October 2020]

[10] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation", BMC Genomics, vol. 21, no. 1, 2020. Available: 10.1186/s12864-019-6413-7.

[11] L. Oakden-Rayner, "The philosophical argument for using ROC curves", Luke Oakden-Rayner Blog, 2020. [Online]. Available: https://lukeoakdenrayner.wordpress.com/2018/01/07/the-philosophical-argument-for-using-roc-curves/. [Accessed: 28- Sep- 2020].

[12] A. Alwosheel, S. van Cranenburgh and C. Chorus, "Is your dataset big enough? Sample size requirements when using artificial neural networks for discrete choice analysis", Journal of Choice Modelling, vol. 28, pp. 167-182, 2018. Available: 10.1016/j.jocm.2018.07.002 [Accessed 1 October 2020].