Benchmarking Keras and NeuralNetwork Classifying the Shuttle Dataset

Gerard Cegarra Dueñas

March 2019

1 Introduction

Solving classification problems is becoming more usual in the scientific community and industry's every day. The Machine Learning algorithms and techniques used to solve them have had a huge increase of scientific attention in the last years, and so, have reached a new level of efficiency that allow them to solve real world sized problems.

A lot of new frameworks and tools for solving classification and other Machine Learning problems have emerged from this wave of scientific work. While this new set of artifacts opens a new range of options for creating solutions that solve virtually any real world problem, choosing the right framework becomes a problem itself.

In this paper two frameworks are tested facing the problem of data classification of the Shuttle data set [1]. The two tested frameworks are Keras [2] with TensorFlow [3] as the backend and the Neuralnetwork (NN) R's package [4][5].

The code implementing the data exploration, models, experimentation and results visualization can be found at the public repository: https://github.com/gerardcd/nn_comparsion

2 The problem

The problem used to benchmark the Keras and NN Machine Learning frameworks is to build a classifier for the data inside the Shuttle data set, using its first 8 attributes to predict the 9-th one.

2.1 Data exploration

Th shuttle data set consists of 8 numerical attributes and a categorical last one which indicates the class of the individual. According to the description found in the UCI repository, the first attribute is the time and the class one is classified as indicated in the next table:

Numerical Value	Meaning
1	Rad Flow
2	Fpv Close
3	Fpv Open
3	High
5	Bypass
6	Bpv Close
7	Bpv Open

The first noteworthy fact about this data set is that the 80% of observations belong to the class "Rad Flow" (1). This means that a classification accuracy of 80% is easily achievable, and so, we will aim for an accuracy close to 100%.

With this class distribution, it is reasonable to expect new observations in real world to follow a similar trend. Moreover, we not only expect the 80% of the new observations to be class 1, but also the other classes' observations to follow a distribution similar to that one of the train set.

Hence, we want our test set to follow this same class distribution. Otherwise, our model validations won't be realistic. For example, a model that never predicts class 1, and a test set with a uniform class distribution, could give a test accuracy up to 6/7 (if the model predicts perfectly the rest of the classes). However, this model would have a way worst accuracy (20%) when used in practice.

In the UCI repository, the Statlog data set is provided already split in train and test sets. The chi-square test is used to check if the distribution of classes is similar in both train and test data set. The expected number of observations EO_c of each class c in the test data set is calculated as:

$$EO_c = (O_c/|TR|) * |TS|$$

where O_c is the number of observations in the train set of class c, $|TR|$ is the size of the train set and $|TS|$ is the size of the test set.

The comparison between expected (EO_c) and actual counts of observations per class in the test set look similar at first glance (Figure 1). The chi-square test confirms that we can't reject the null hypothesis (distributions are the same) at a confidence level of 95%.

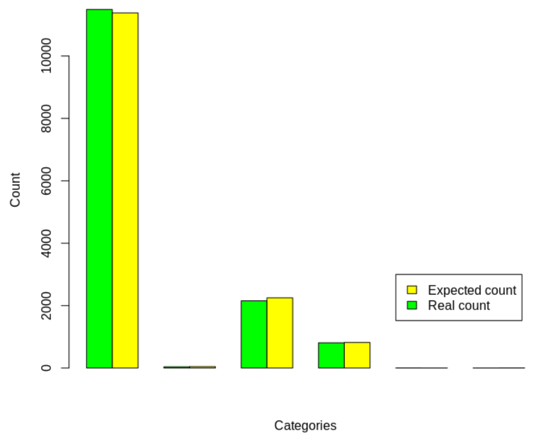

Figure 1: Real (green) and expected (yellow) counts of each class in test data set

As a second data exploration step, PCA is used to visualize the data in 2 and 3 dimensions. More than 50% of the variance is explained by first two dimensions and more than 67% of the variance is explained by 3 first dimensions, which means that the 2D and 3D visualization are fairly good pictures of the set shape.

Looking at variable contributions to PCA's dimensions, the variable 9 doesn't practically contribute to any dimension. That may happen because another variable is already explaining its variance. Looking at the PCA correlation graph we can see a correlation between V8 and V9 (Figure 2). In addition, the Pearson method gives a value of correlation of 0.8595272. However, we don't get rid of neither V8 or V9, as it has been found that removing them lead to worst classification results.

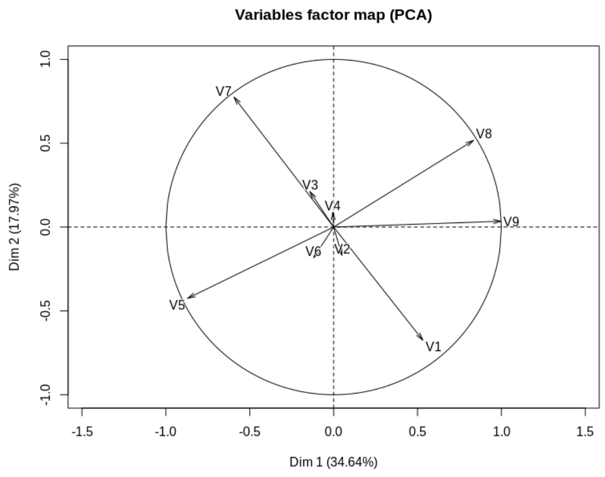

Figure 2: PCA factor map of Shuttle data set

When plotting individuals coloured by their class in PCA's dimensions space, each class seem to conform one or more compact clusters (Figures 3 and 4). Hence, a neural network with one hidden layer and a short number of hidden neurons should be enough to classify this data.

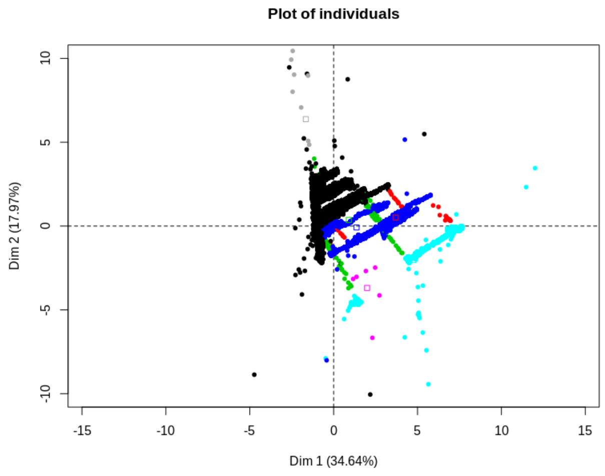

Figure 3: Individuals (painted by class) plotted in first two PCA dimension's space

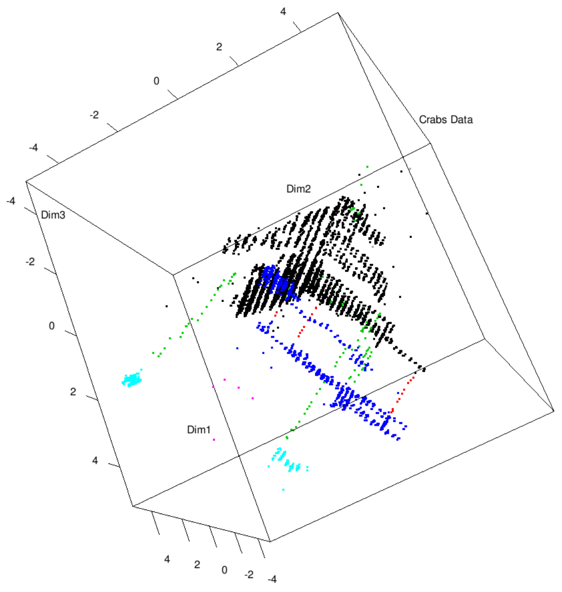

Figure 4: Individuals (painted by class) plotted in first three PCA dimension's space

2.2 Classification models

This section defines the two models used for the classification problem. In both Keras and NN frameworks, a neural network of one hidden layer will be created for latter experimenting. Every configuration parameter that is relevant to a classification problem has been configured identically in both neural networks. The following table shows the values for these parameters:

Parameter	Value
Batch size	128
Hiden layer activation	Relu
Output layer activation	Softmax
Optimizer	RMSprop
Loss function	Categorical cross entropy
Learning rate	0.001
Decay	0.9

The learning rate and the decay values used are the default Keras values for the RMSprop optimizer's learning rate and decay (rho parameter) respectively. The NN's "log" value for the loss function parameter corresponds to the "categorical_crossentropy" Kera's value.

The number of hidden neurons and training epochs will be parametrized with different values during the bench-marking process.

3 Bench-marking

In order to benchmark the two frameworks, Keras and NN, the two models built in the previous section have been trained on the Shuttle data set with different numbers of epochs e and hidden neurons h. Every combination of e and h have been tested for $e, h \in \{2^i \mid 0 \leq i \leq 5\}$. The following sub-sections present the results of these experiments.

The environment used to test the models is composed by a i7 core with 16GB of RAM, running Ubuntu. The R version is 3.5.3 and the Python version is 2.7.15rc1.

3.1 Time

This sub-section presents the training time results for Keras and NN frameworks for each number of epochs and hidden neurons.

Keras time seems not to be affected by the number of hidden neurons (Figure 5). This might happen because the numbers we are using here. Even the bigger combinations with 32 epochs and 32 hidden neurons are too small for Tensor Flow. More experiments are performed latter with Keras and larger numbers (Section 3.3).

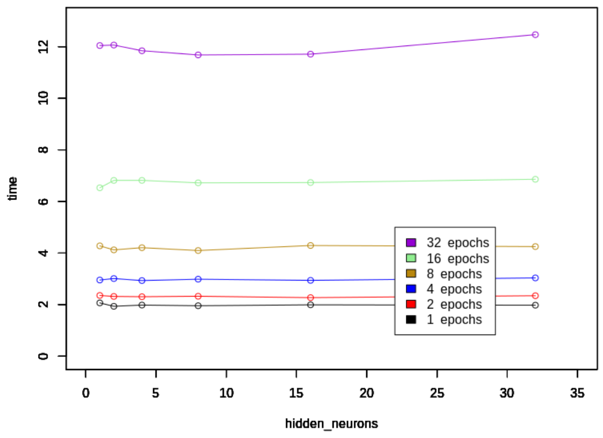

Figure 5: Keras training time by different numbers of hidden neurons and epochs

On the other hand, NN training time is affected by number of hidden neurons, and of course, by the number of epochs. The training time grows linearly with the number of hidden neurons (Figure 6), and this trend is multiplied by a factor also linearly related to the number of epochs.

When fixing the number of epochs fixed to 32, despite both models start with similar training time for one hidden neuron, the training time of the NN model seems to increase linearly with the number of hidden neurons while the training time of the Keras stays constant (Figure 7). Putting this two times in comparison shows a big difference in the frameworks efficiency.

3.2 Acuracy

This sub-section presents the accuracy results for Keras and NN frameworks for each number of epochs and hidden neurons.

Keras accuracy is unstable in the configuration window tested in these experiments (Figure 8). It doesn't follow a clear pattern. Again, this might be due to the small size of the numbers used. The experiments performed latter also include the measurement of the accuracy for larger Keras models (Section 3.3).

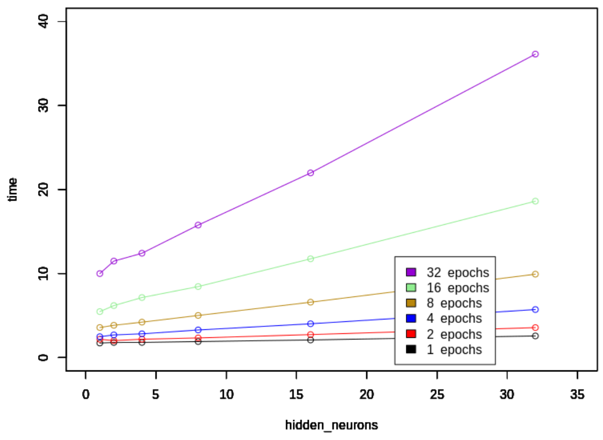

Figure 6: NN training time by different numbers of hidden neurons and epochs

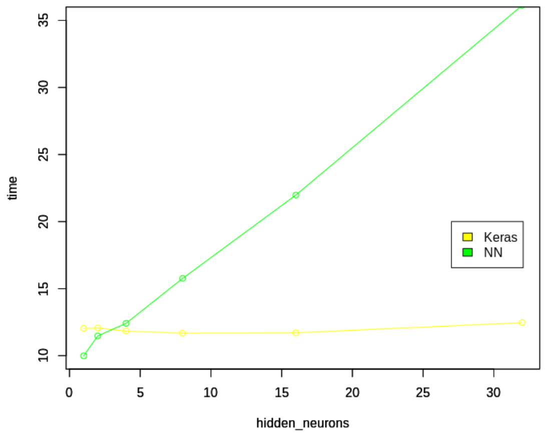

Figure 7: Keras and NN training time by different numbers of hidden neurons and 32 epochs

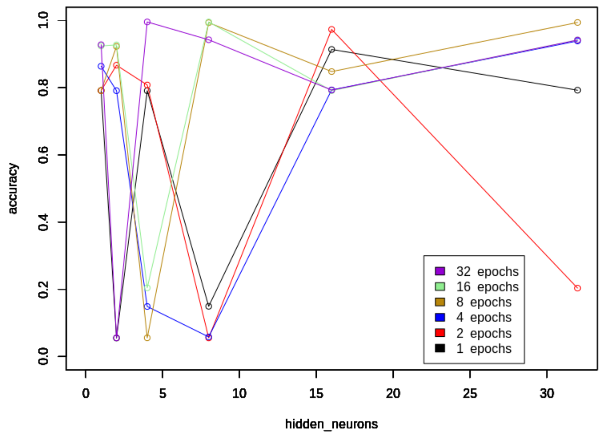

Figure 8: Keras accuracy by different numbers of hidden neurons and epochs

As opposite to Keras, NN's accuracy is more stable and quickly converges to numbers very close to 1 (Figure 9). Even in one training epoch, the NN model shows a clear converging pattern after an eventual drop at 2 hidden neurons.

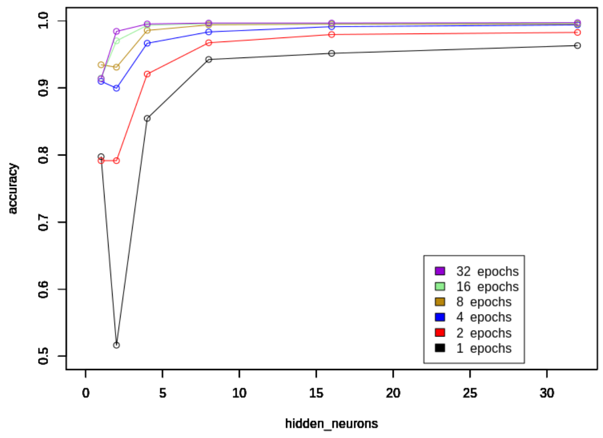

Figure 9: NN accuracy by different numbers of hidden neurons and epochs

Similar to the comparison done before with time, both models' accuracy is compared for 32 epochs and different number of hidden neurons (Figure 10). It's clear that NN is more reliable in terms of accuracy: it is converging to a maximum and never falling down eventually like in the case of Keras.

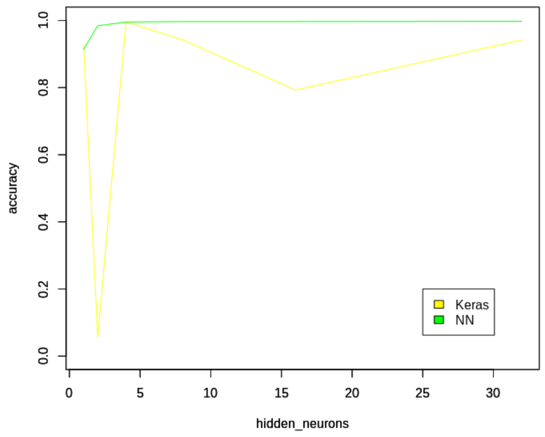

Figure 10: Keras and NN accuracy by different numbers of hidden neurons and 32 epochs

3.3 Keras long run

Finally, this section aims to show time and accuracy results for Keras neural networks when using a wider range of configuration parameters. The following results show how the training process scales in relation to the size of the hidden layer using 32 training epochs.

Scaling up the numbers show that the learning proces time grows actually linnearly with the number of hidden neurons (Figure 11). However, the relation factor between time and hidden neurons is much more smaller than that one of NN. Recall, that NN was already lasting almost 40 seconds to train a neural network of 32 hidden layers in 32 epochs. In slightly more than 40 seconds, Keras is training a neural network with 4096 hidden neurons in 32 epochs.

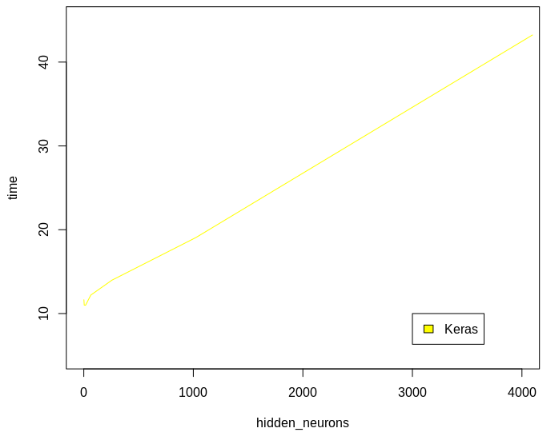

Figure 11: Keras training time by different numbers of hidden neurons and 32 epochs

On the other hand, the accuracy measurements show how the network accuracy starts to converge at 16 neural networks (Figure 12). Then, it seems to reach a maximum at 256 hidden neurons, which is still preserved at 1024 neurons, and then starts to drop down after that. This decreasing trend happens because from 1024 hidden neurons on, the network needs more than 32 epochs to be trained with a high accuracy.

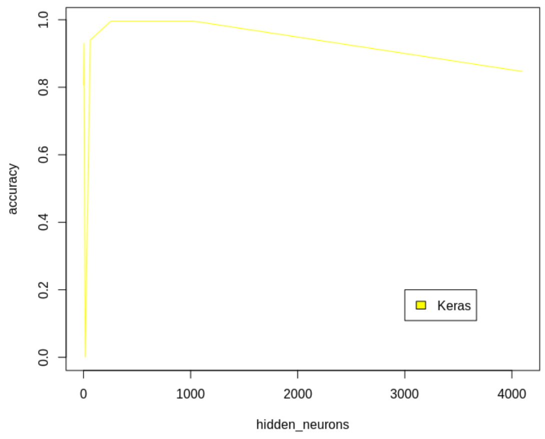

Figure 12: Keras accuracy by different numbers of hidden neurons and 32 epochs

4 Conclusions

After analysing the experiments results, due to the problem complexity and size, NN seems a more suitable option to work with the shuttle data set. Although its training time is slower than the Keras one, it's still reasonable. Regarding the accuracy, NN's one is more stable and converges faster for this particular problem, which makes this framework a more reliable solution.

As seen in the above section 3.3, Keras would be a more appropriate framework for problems in another scale of size, as it has proved to be way more efficient while training neural networks of bigger sizes compared to NN.

References

[1] Dua, D. and Graff, C. (2019). Statlog (Shuttle) Data Set, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[2] Keras Framework [http://keras.io], 2019

[3] Google, TensorFlow Framework [http://keras.io], 2019

[4] Bart Lammers, Neural Network, ANN2 []https://cran.r-project.org/web/packages/ANN2/index.html], 2019

[5] Ross I. and Robert G, R language, 2018