Application of a Neural Network for Bat Family Classification from Call Sounds

Data Processing

The performance of a neural network with 2 hidden layers and a neural network with 3 hidden layers was compared with respect to their performance at predicting the family a bat call belongs to. Performance was optimised by minimising the expected error evaluated on the training data:

$$\overline{E}(w) = \frac{1}{N}\sum_{i=1}^{N} E_i(w)$$

In this case of a categorical classification problem, the error function is the cross entropy:

$$E(w) = -\sum_{t=1}^{N}\sum_{c=1}^{C} y_{ic} log(o_{ic})$$

Weights are updated based on minimising the expected value of the loss function. The performance metrics used to compare the two models are accuracy and value of the error function at each epoch for the training and test datasets. The evolution of these parameters over the course of the 100 epochs on the training data is shown in the plots below.

The data were recoded to define the bat classes as categorical and one-hot encoding was used to create a series of binary variables based on these categories (see R code below).

Neural Networks

The *keras* package in R was used to fit two and three hidden layer neural networks. L2 regularization was used to apply penalties on the layer parameters that are summed into the loss function and that prevent over fitting of the model. A dropout layer was included where randomly selected neurons are ignored during training so that their contribution to downstream neurons is temporarily removed. This allows better generalization and makes the model less likely to overfit the training data.

Figure 1 Training a 2-hidden layer neural network (left) and a 3-hidden layer neural network (right)

The performance of the models was estimated by assessing their ability to correctly predict the classification of the bats in the test data, which was not involved in the derivation of the models. Performance was also evaluated using a classification table which displays the numbers of bats that were correctly and incorrectly classified by each model. The classification tables for models 2 and 3 are shown below.

```
################################################################################
#  classification tables
################################################################################

# look at classification table for both models
y_raw2 <- max.col(y_test) - 1 # to convert back to categorical
class_hat2 <- model2 %>% predict(x_test) %>% max.col() # predicted classes
tab2 <- table(y_raw2, class_hat2)

y_raw3 <- max.col(y_test) - 1 # to convert back to categorical
class_hat3 <- model3 %>% predict(x_test) %>% max.col() # predicted classes
tab3 <- table(y_raw3, class_hat3)

# compute accuracy model2 and 3
acc_model2<-sum( diag(tab2) ) / sum(tab2)
acc_model3<-sum( diag(tab3) ) / sum(tab3)

colnames(tab2) = c('Vespertilionidae','Molossidae','Mormoopidae','Phyllostomidae','Emballonur
idae')
rownames(tab2)= c('Vespertilionidae','Molossidae','Mormoopidae','Phyllostomidae','Emballonuri
dae')
x<-knitr::kable(tab2, caption = " Table 1 Classification table of bat species from Model 2 (2
hidden layers)", format = "html", table.attr = "style='width:100%;'")
column_spec(x, 1, bold = TRUE)
```

Table 1 Classification table of bat species from Model 2 (2 hidden layers)

	Vespertilionidae	Molossidae	Mormoopidae	Phyllostomidae	Emballonuridae
Vespertilionidae	3249	109	0	1	107

	Vespertilionidae	Molossidae	Mormoopidae	Phyllostomidae	Emballonuridae
Molossidae	225	433	0	62	0
Mormoopidae	2	0	510	26	6
Phyllostomidae	3	17	34	626	0
Emballonuridae	195	0	7	2	830

```
##############################################################################
#  classification tables
##############################################################################
colnames(tab3) = c('Vespertilionidae','Molossidae','Mormoopidae','Phyllostomidae','Emballonur
idae')
rownames(tab3)= c('Vespertilionidae','Molossidae','Mormoopidae','Phyllostomidae','Emballonuri
dae')
y<-knitr::kable(tab3, caption = " Table 2 Classification table of bat species from Model 2 (3
hidden layers)", format = "html", table.attr = "style='width:100%;'")
column_spec(y, 1, bold = TRUE)
```

Table 2 Classification table of bat species from Model 2 (3 hidden layers)

	Vespertilionidae	Molossidae	Mormoopidae	Phyllostomidae	Emballonuridae
Vespertilionidae	3290	58	0	7	111
Molossidae	340	329	0	51	0
Mormoopidae	1	0	519	18	6
Phyllostomidae	2	24	105	549	0
Emballonuridae	183	0	9	3	839

The overall accuracy of Model 2 was 0.8427 while Model 3 had a slightly higher accuracy of 0.8696. The classification tables for both models above breaks down misclassification by bat species. Both models perform poorly in classifying Molossidae, (37% and 57% correctly classified for model 2 and 3, respectively. Model 2 and 3 approach their highest value achieved during the 100 epochs after around 40 epochs, with the rate of performance improvement continuing at a slower rate thereafter. Nevertheless, the rate of improvement of performance for both models is still increasing at a very slow rate at the end of the 100 epochs.

Because the training set was used to learn the parameters that reduce the estimated training error, these estimating performance metrics are optimistic estimates, but in this case the test error seems to be greater than or equal to the estimated training error. This could be because the data used for training and testing the model are not homogenous, that there is some difference between the two datasets that affects our ability to compare the performance of the models. A more likely reason for higher test compared to training error in this case is the dropout regularization layer which is deactivated when evaluating the test data and can give an improved accuracy on test datasets. When dropout is applied during model training, a percentage of the network nodes are switched off, leading to deliberate underfitting of the data. In contrast, during the testing stage all the network nodes are employed, giving increased fitting power, robustness and higher test accuracy. The higher accuracy and lower loss function values seen in the test data fit here for model 2 and 3 suggest the dropout regularisation has had this effect.

The loss and accuracy predicted by the models are plotted in Fig 2 below. The gap between test and training data is slightly larger for model 3 compared to 2, but again, this is a very small difference that probably doesn’t support a function improvement in performance for model 3.

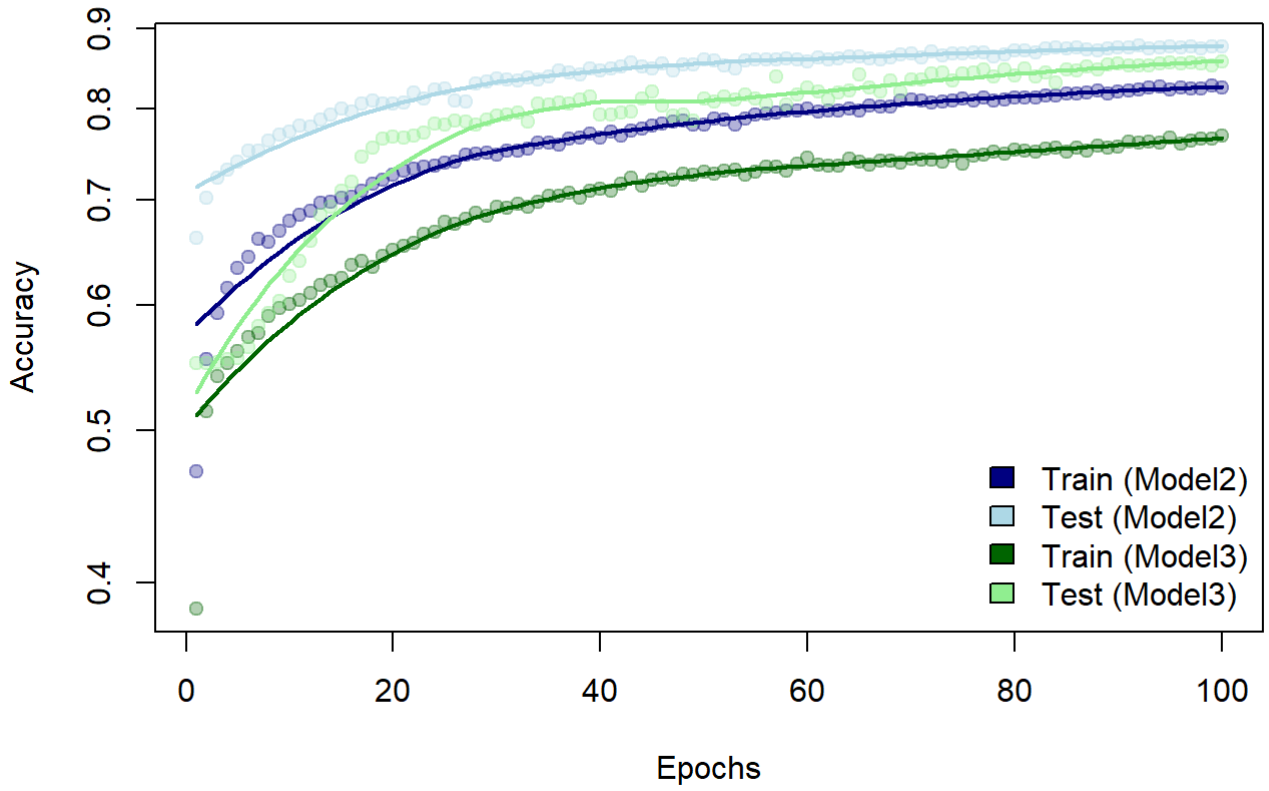
```
# to add a smooth line to points
smooth_line <- function(y) {
  x <- 1:length(y)
  out <- predict( loess(y ~ x) )
  return(out)
}
# some colors will be used later
cols <- c("navy", "lightblue", "darkgreen", "lightgreen")

#compare accuracy with 2 and 3 hidden layers
out <- cbind(fit2$metrics$accuracy,
             fit2$metrics$val_accuracy,
             fit3$metrics$accuracy,
             fit3$metrics$val_accuracy)
str(fit2$metrics)
```

```
## List of 4
##  $ loss        : num [1:100] 2.21 1.89 1.73 1.64 1.57 ...
##  $ accuracy    : num [1:100] 0.471 0.555 0.593 0.615 0.634 ...
##  $ val_loss    : num [1:100] 1.74 1.58 1.47 1.38 1.32 ...
##  $ val_accuracy: num [1:100] 0.662 0.702 0.723 0.732 0.74 ...
```

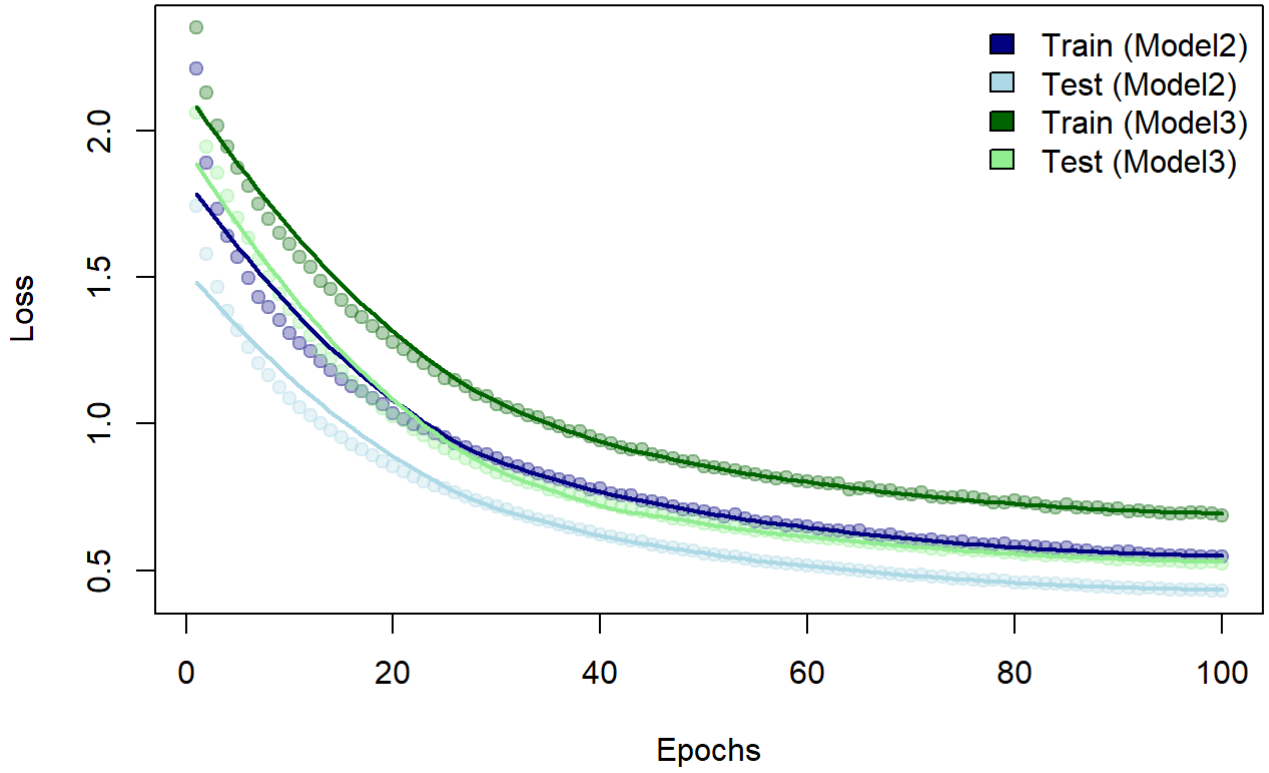
```
# compare performance with 2 and 3 hidden layers
matplot(out, pch = 19, ylab = "Accuracy", xlab = "Epochs",main="Figure 2  Comparing accuracy
 of 2- and 3-hidden layer neural networks",
        cex.main = 1,font.main=1, col = adjustcolor(cols, 0.3), log = "y")
matlines(apply(out, 2, smooth_line), lty = 1, col = cols, lwd = 2)
legend("bottomright", legend = c("Train (Model2)", "Test (Model2)", "Train (Model3)", "Test
 (Model3)"),
       fill = cols, bty = "n")
```

Figure 2 Comparing accuracy of 2- and 3-hidden layer neural networks




```
# compare loss with 2 and 3 hidden layers
loss <- cbind(fit2$metrics$loss,
              fit2$metrics$val_loss,
              fit3$metrics$loss,
              fit3$metrics$val_loss)
matplot(loss, pch = 19, ylab = "Loss", xlab = "Epochs",main = "Figure 3  Comparing loss of 2-
and 3-hidden layer neural networks",
        cex.main = 1,font.main=1,col = adjustcolor(cols, 0.3))
matlines(apply(loss, 2, smooth_line), lty = 1, col = cols, lwd = 2)
legend("topright", legend = c("Train (Model2)", "Test (Model2)", "Train (Model3)", "Test (Mod
el3)"),
       fill = cols, bty = "n")
```

Figure 3 Comparing loss of 2- and 3-hidden layer neural networks



The performance metrics for models 2 and 3 outlined above, suggest that there was very little difference between the models and do not support addition of an extra hidden layer. The overall accuracy of Model 2 was 0.8427 while Model 3 had a slightly higher accuracy of 0.8696, but this difference is small and probably doesn't represent a functionally important difference in model performance between the two models. Rather, this small difference indicates that the addition of the regularisation step does not impact the performance of the model sufficiently to justify the increased complexity it brings to the model.

The accuracy increased to over 80% in the test data on both models (Figure 2), and it is possible that longer epochs might have improved the performance of both models further since accuracy was still increasing after 100 epochs. There is no evidence of overfitting, since the test accuracy was greater than the training accuracy for both models.