Queen Mary University of London
The Alan Turing Institute

GitHub https://github.com/adrianbevan/TensorFlow-Tutorial

launch binder https://mybinder.org/v2/gh/adrianbevan/TensorFlow-Tutorial/v1.0

ADRIAN BEVAN

TENSORFLOW TUTORIAL

GIVEN IN THE LIV.DAT DATA SCIENCE SCHOOL 12-17OCT 2020

If you want more background material on these ML methods then please see my graduate lectures from the 2020 RAL PPD Summer Lecture Series.

I also have other machine learning lectures available on my teaching webpage.

OVERVIEW

- The code for this tutorial can be found on [github](), and you will be using [Binder]() to work with the code.

- Once the binder session starts click on the `notebooks` directory to navigate to the jupyter notebooks for this tutorial.

- Package requirements for this include:

```
matplotlib==3.2.1
sklearn
tensorflow==2.2.0
numpy==1.18.4
seaborn==0.11.0
```

OVERVIEW

- The following examples are provided to work through:
    - LinearRegression.ipynb
    - NN_parabola.ipynb
    - NN.ipynb
    - CNN.ipynb

- The scripts directory of the github code also includes example scripts for hyper-parameter optimisation that you may wish to explore in your own time.

OVERVIEW

- Training hyper-parameters of interest include:
  - **Batch Size:**
    - The number of examples used in a given iteration of the optimisation algorithm.
  - **Dropout Rate:**
    - The fraction of nodes dropped out in a given layer of a network.
  - **Leaky ReLU alpha:**
    - The coefficient multiplying the negative half of the activation function.
  - **Learning Rate:**
    - Related to the optimisation algorithm step size (usage depends on algorithm).
  - **Nepochs:**
    - Number of times the training data are looped over when learning the model.
  - **Validation Split:**
    - The fraction of training data used for validation when learning the model.
- The model architecture is also configurable and this affects model performance.

LINEAR REGRESSION

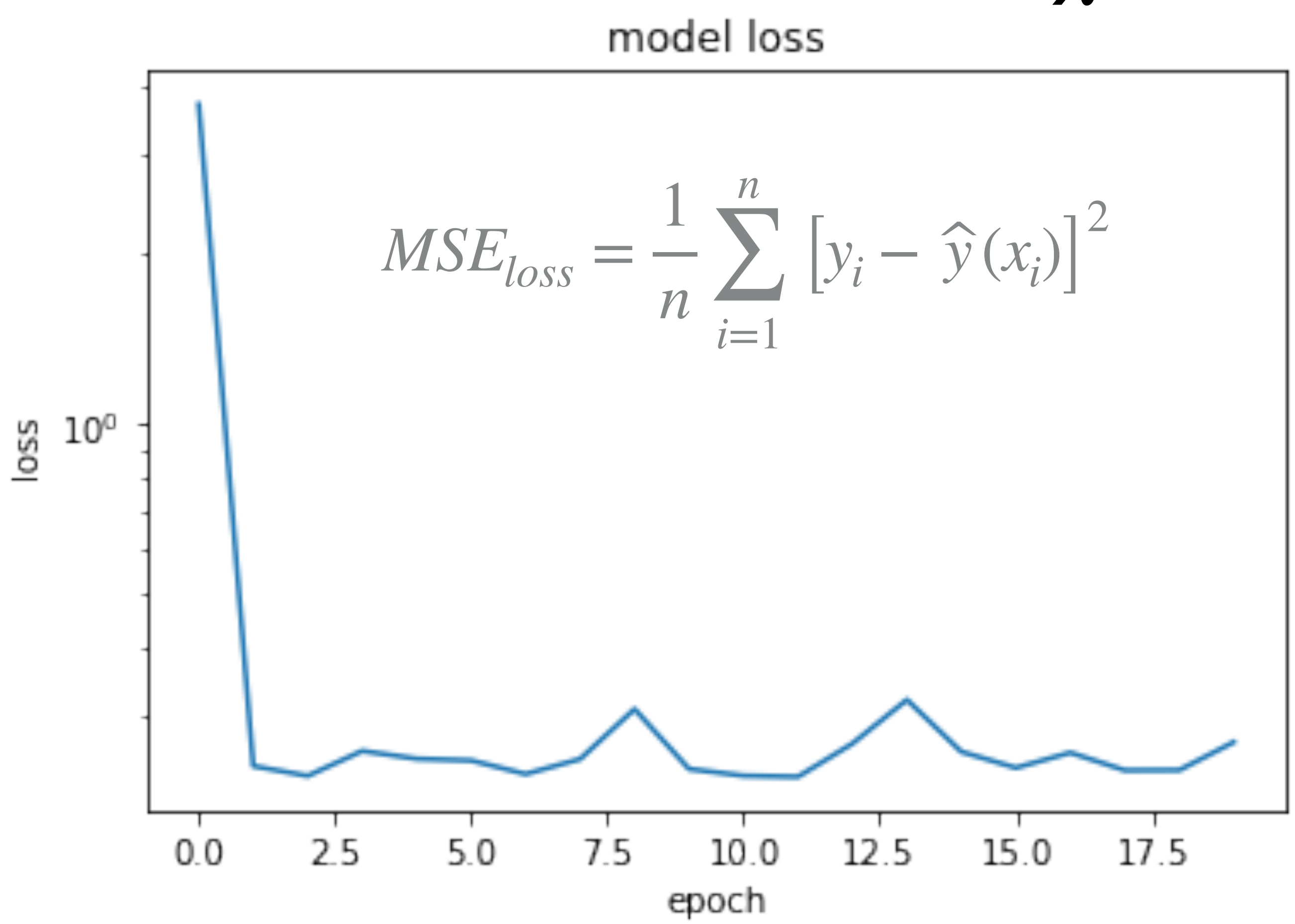
LINEAR REGRESSION:

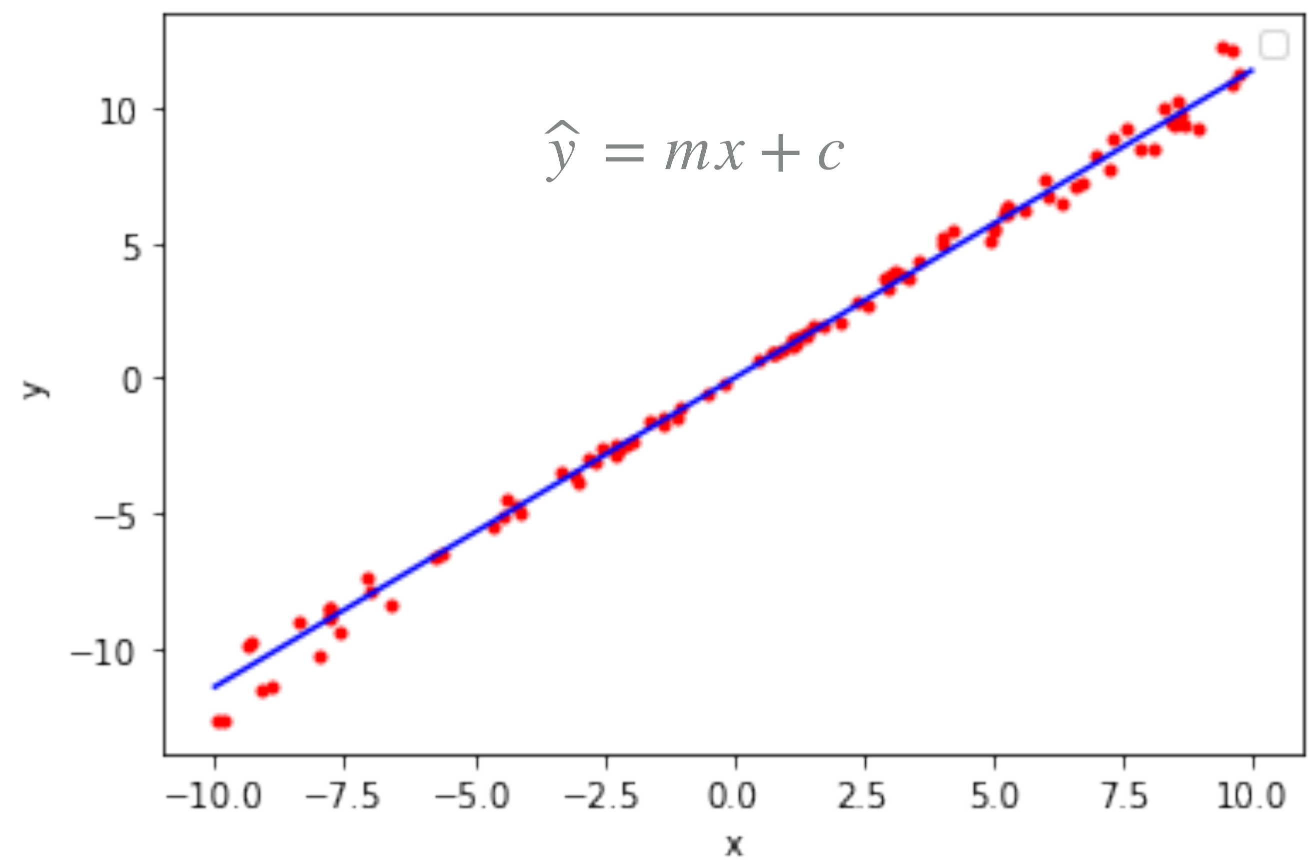
- LinearRegression.ipynb
- Generate noisy data according to $y = mx + c$
- Use a linear activation function to learn m and c
  - How many inputs?
  - How many outputs?
  - How many model hyper-parameters?
- Use the Adam optimiser to learn the function.

LINEAR REGRESSION:

- [LinearRegression.ipynb]

- Able to fit a straight line to extract the parameters.
- Unlike a likelihood or χ^2 fit, we don't get uncertainties





LINEAR REGRESSION: SUGGESTED EXERCISES

- LinearRegression.ipynb
- Change the number of training examples to see how this affects the optimisation performance (increase by a factor of 10 and decrease by a factor of 10).
- Change the value of m and c to extract, Try m=1000, c=-500, to explore how this affects the training. You may also need to change the number of epochs when doing this.
- Change the number of training epochs to see how this affects the optimisation
- Change the noise level to study how this affects the optimisation.
- Change the learning rate to explore how robust the training is with the Adam optmiser.
- You may also wish to explore the use of other optmisers: see https://keras.io/api/optimizers/.

THERE ARE 2 EXAMPLES:

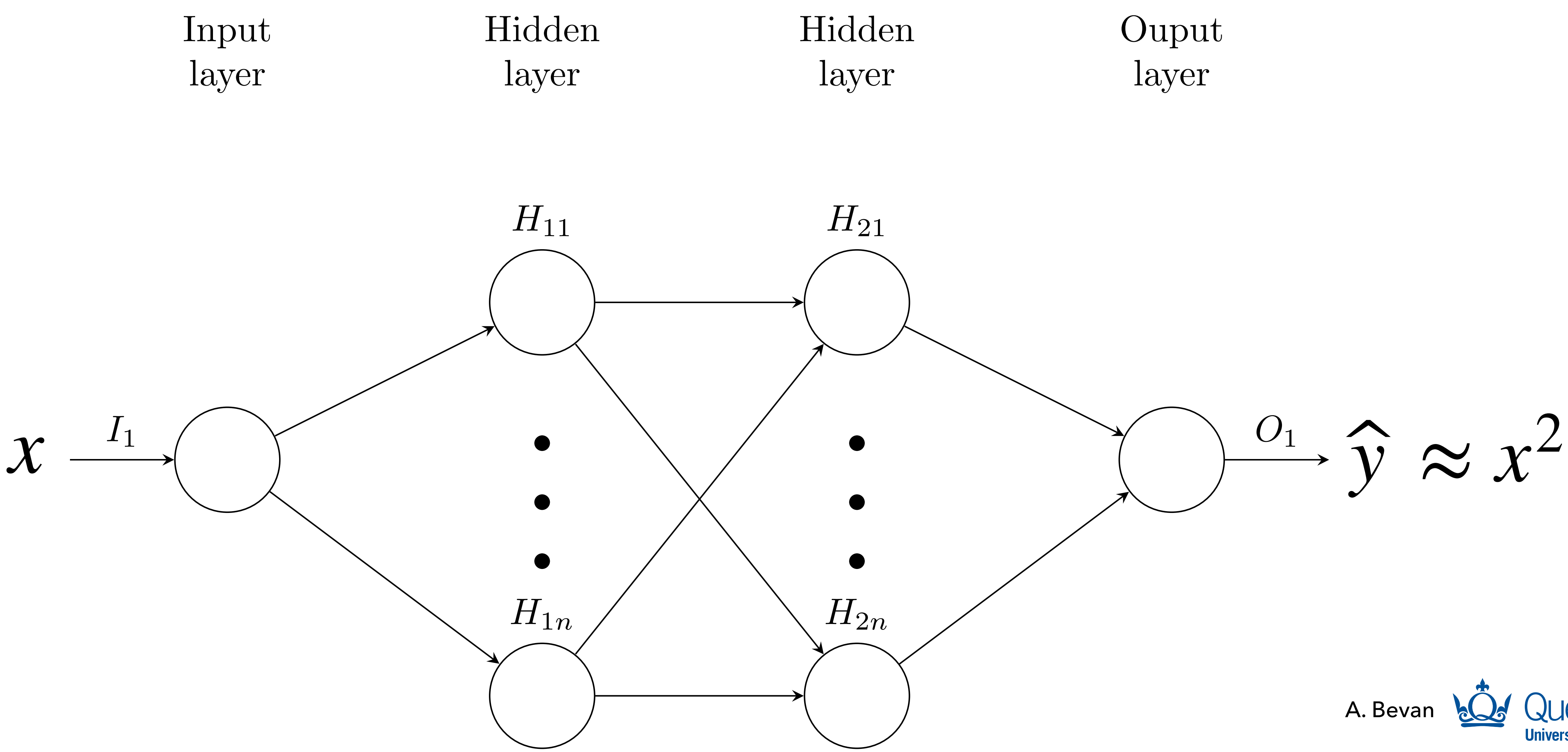
1) PARABOLIC REGRESSION PROBLEM: LEARNING $y = x^2$

2) MNIST CLASSIFICATION PROBLEM: IDENTIFYING HAND WRITTEN NUMBERS

NEURAL NETWORKS

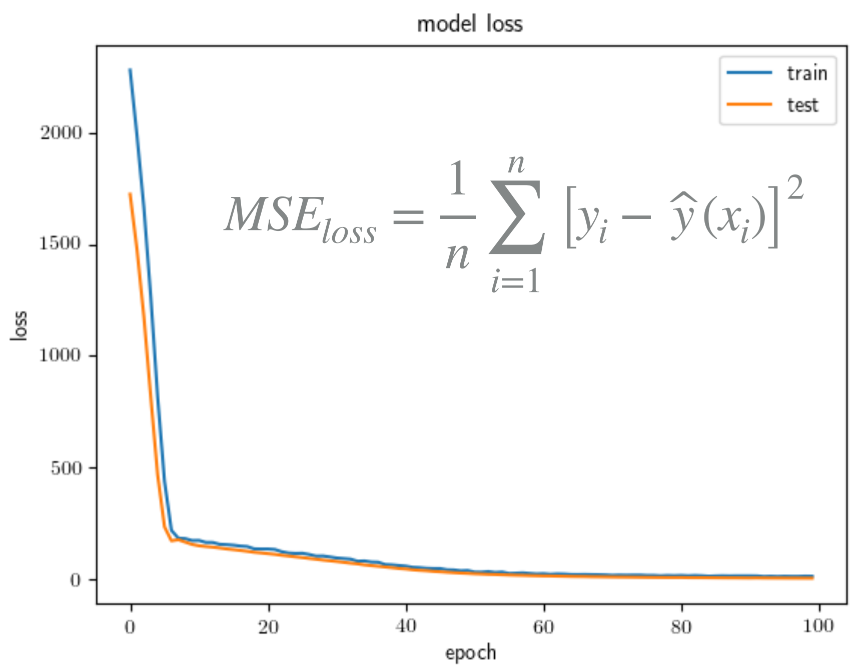
NEURAL NETWORKS

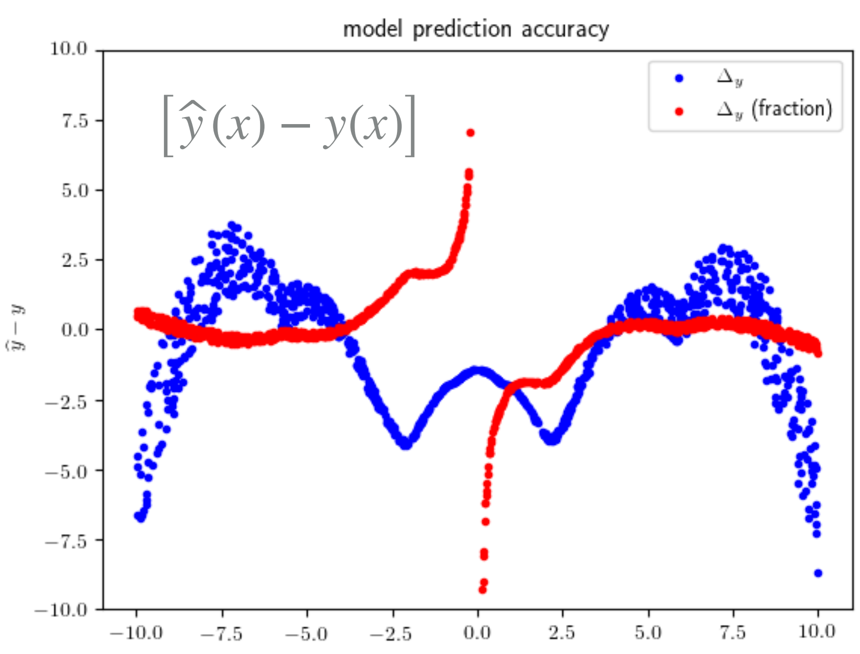
- NN_parabola.ipynb
- Generate noisy data according to $y = x^2$
- Use a multilayer perceptron to learn the function
  - Remember that machine learning is just function approximation (although we may not always think of it in those terms).



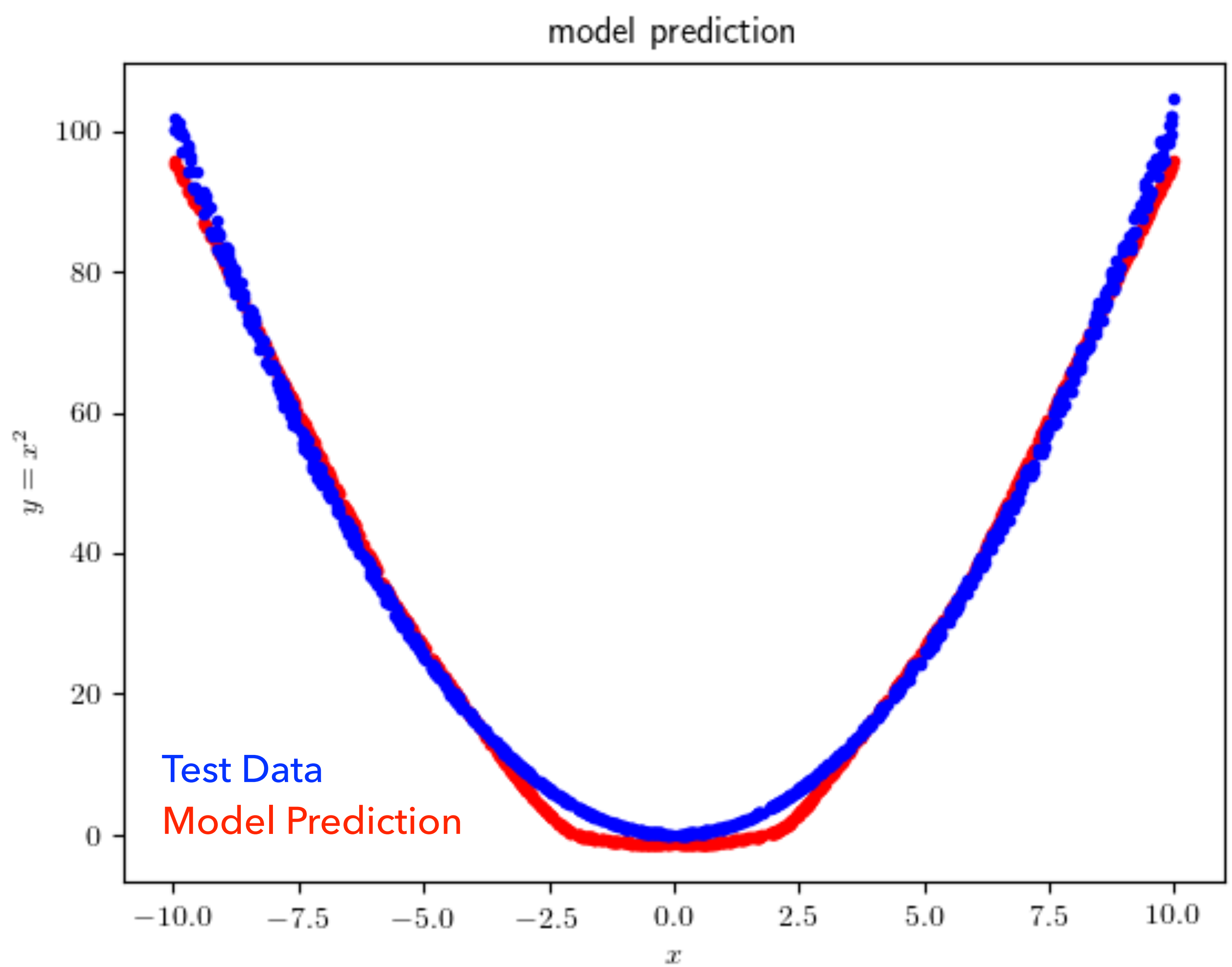
NEURAL NETWORKS

- NN_parabola.ipynb





With a little exploration and tweaking of hyper-parameters you should be able to get a much better model than this.



NEURAL NETWORKS: SUGGESTED EXERCISES

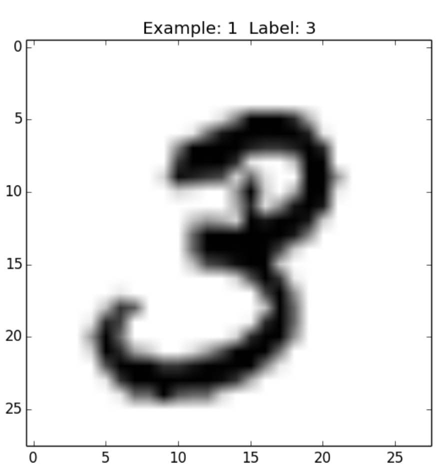
- NN_parabola.ipynb

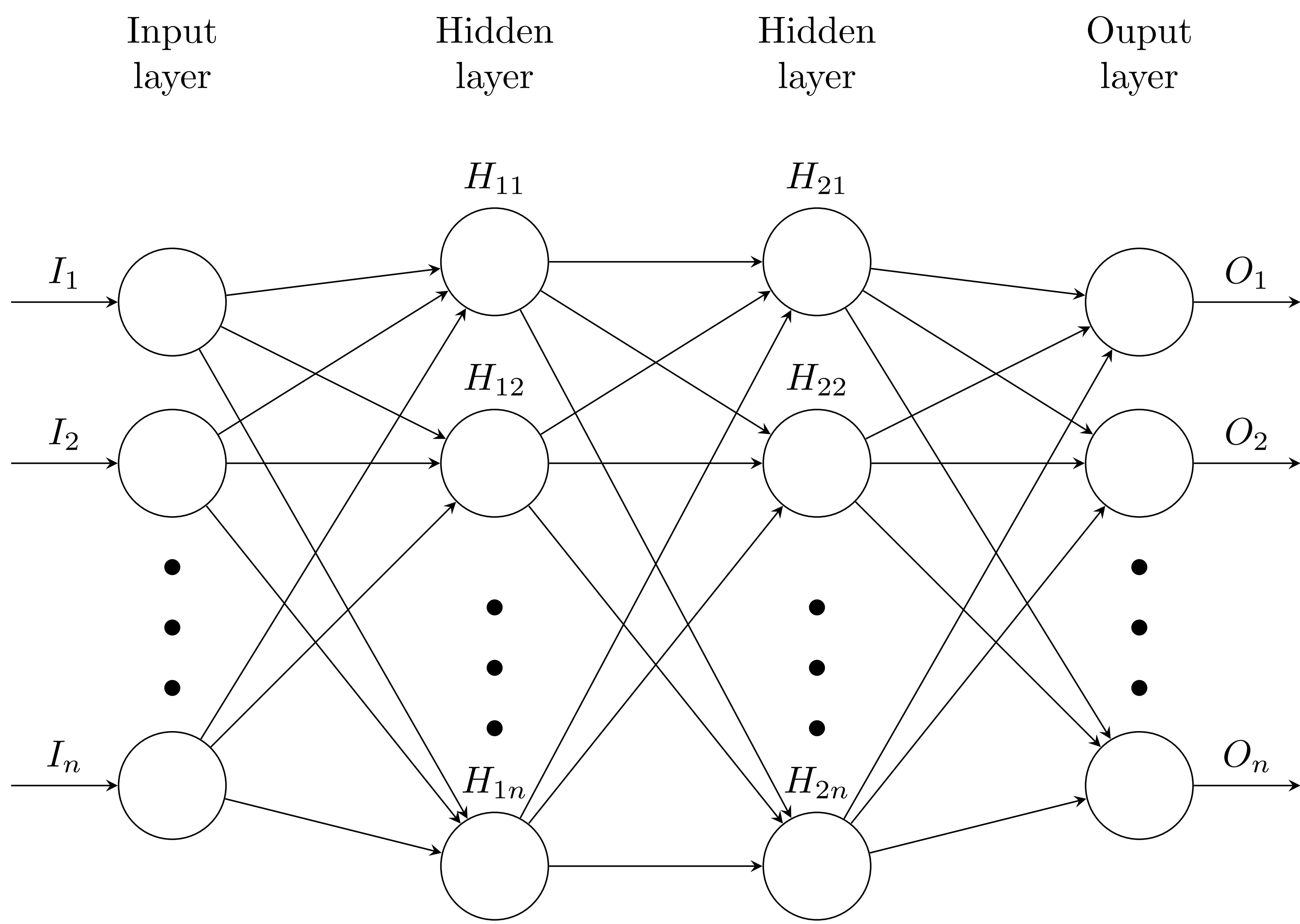
- Explore the effect of DropOut, ValidationSplit, Nepochs, and BatchSize have on the training (try to find a model where the test and train loss function values are similar.
- Explore how the neural network structure affects the training performance (e.g. add double or halve the number of nodes in the hidden layers, the current value is 128 for both)
- Explore the effect of adding a second dropout layer into the network after the first hidden layer.
- Explore what happens when the model is reduced to a single layer perceptron (removing the second hidden layer).
- Explore what happens when the model is changed by adding a third hidden layer to it.

NEURAL NETWORKS

- NN.ipynb
- Use MNIST data
- Use a multilayer perceptron to learn the classification function for the numbers 0, 1, …, 9

784 dimensional input feature space of a flattened image





Output vector

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

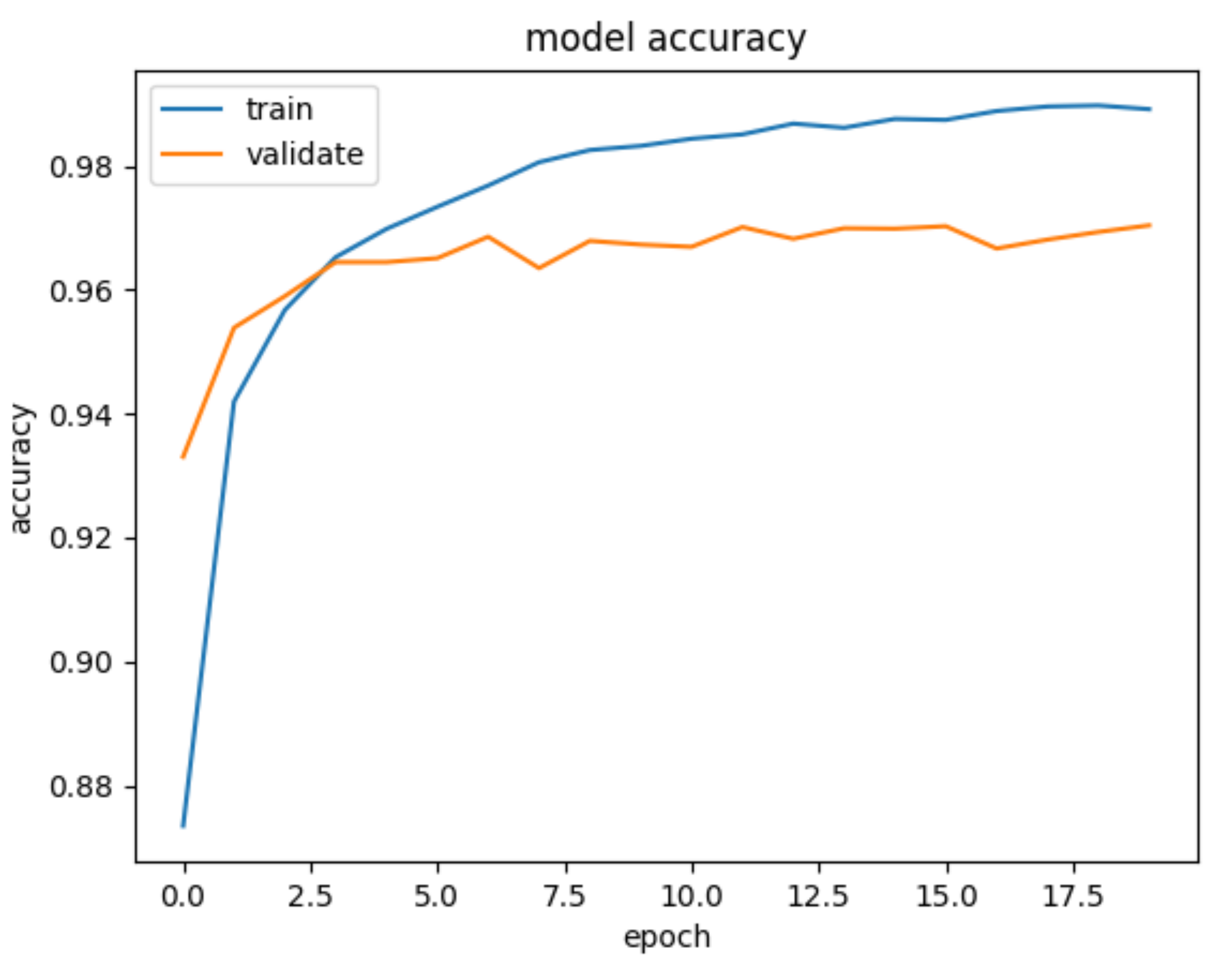
NEURAL NETWORKS

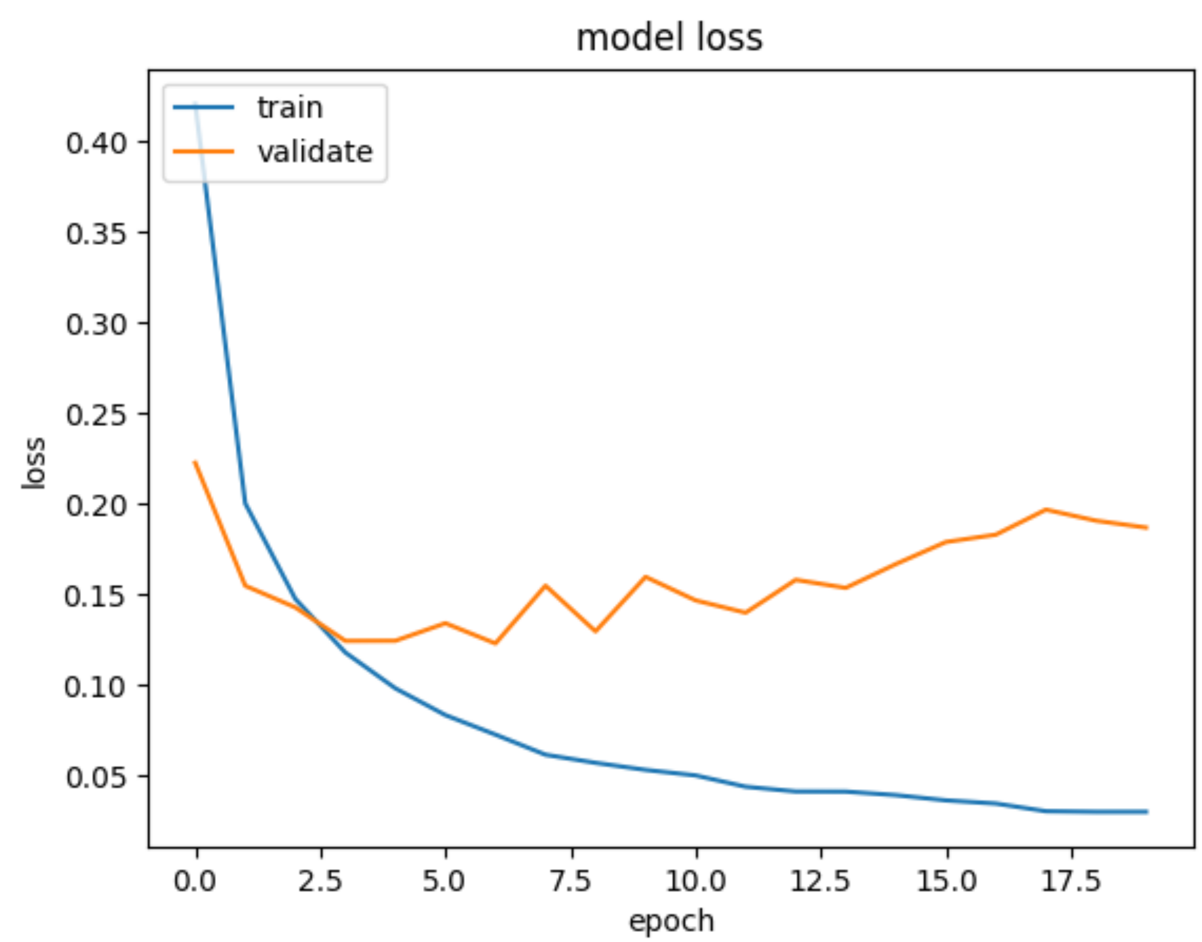
- [NN.ipynb]

Good accuracy, but train and validate sample losses differ - this model overtrains.

Need to vary hyper-parameters to avoid overtraining the model.

BatchSize, DropoutValue and ValidationSplit are hyper-parameters that you might like to vary (along with increasing the number of epochs, Nepochs).





NEURAL NETWORKS: SUGGESTED EXERCISES

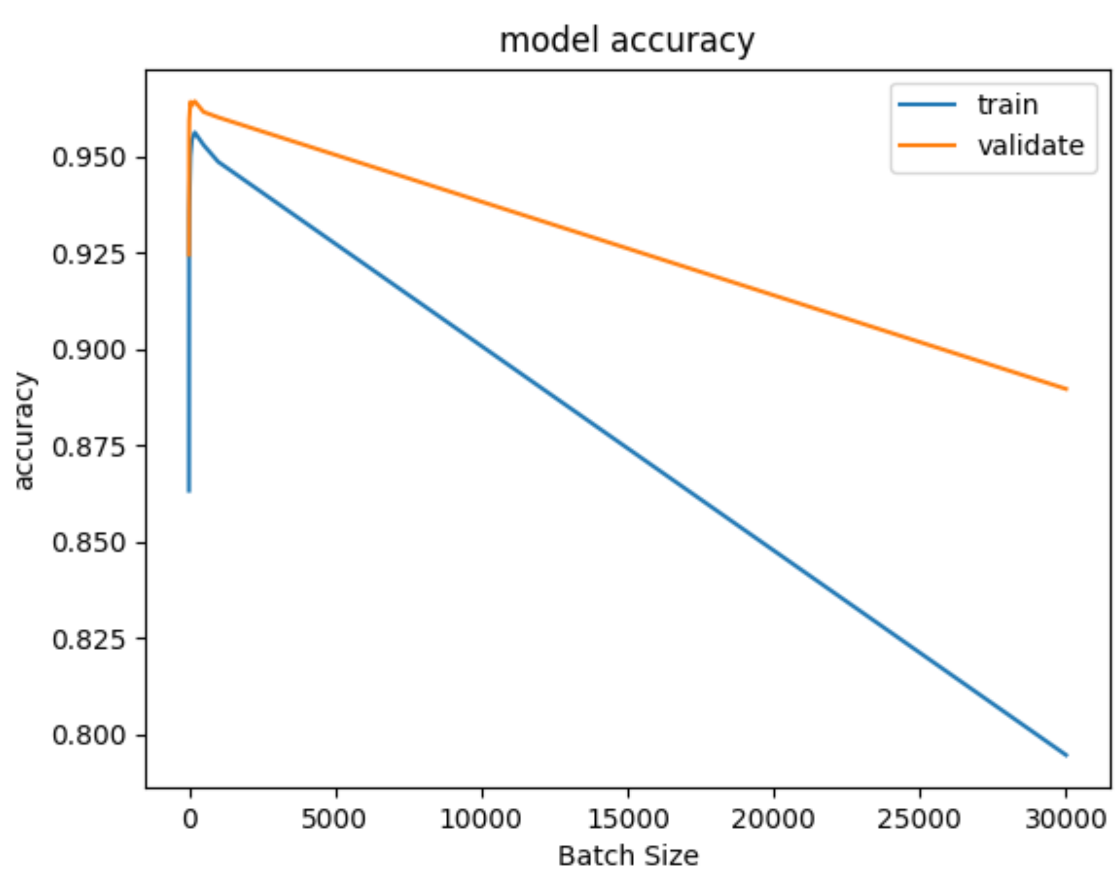
- NN.ipynb

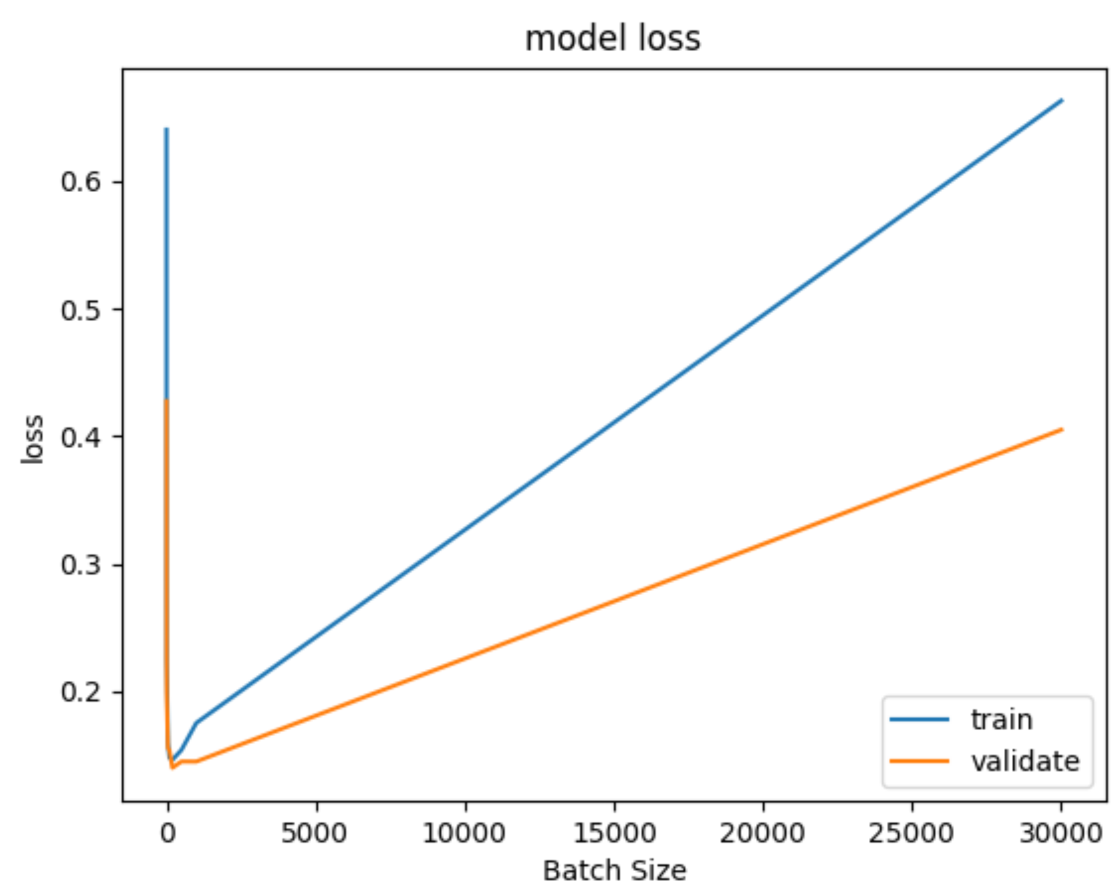
- Explore the effect of DropOut, ValidationSplit, Nepochs, and BatchSize have on the training (try to find a model where the test and train loss function values are similar.
- Explore how the neural network structure affects the training performance (e.g. add double or halve the number of nodes in the hidden layers, the current value is 128 for both)
- Explore the effect of adding a second dropout layer into the network after the first hidden layer.

HYPER-PARAMETER TUNING: BATCH SIZE

- The model hyper-parameters are not just the weights and biases in the network (for NN), the parameters chosen for the training and indeed model configuration affect model performance.

For this NN model, small batch sizes maximise model accuracy & minimise overtraining

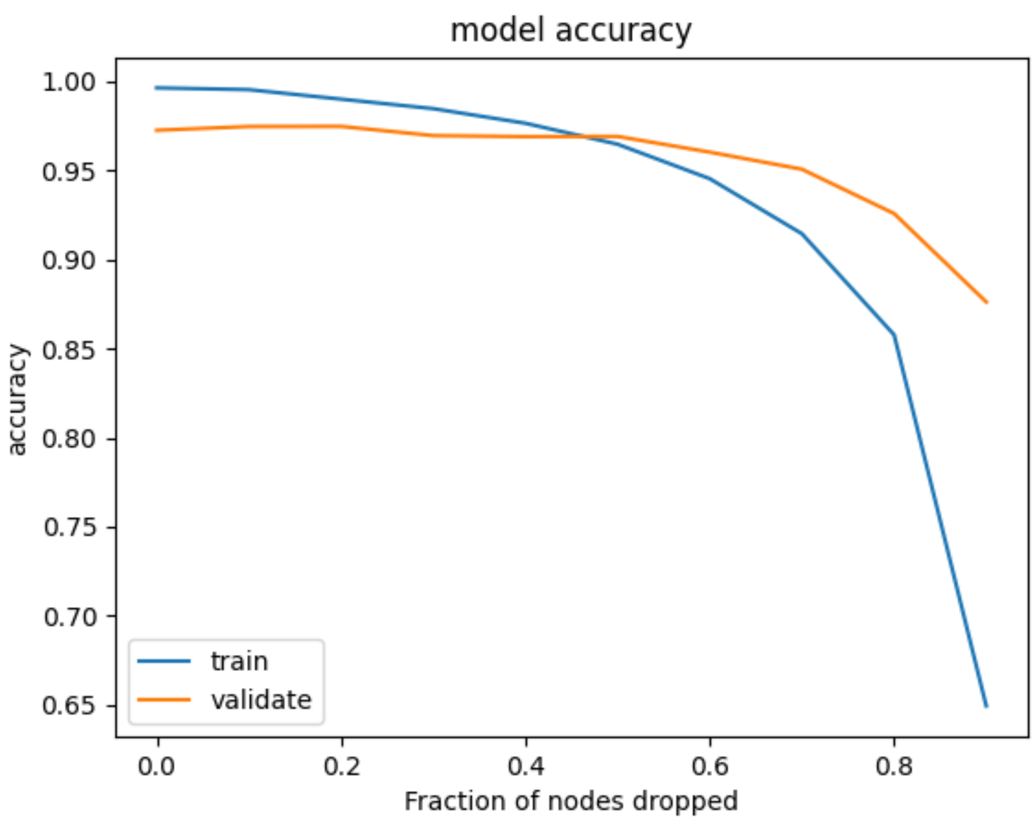


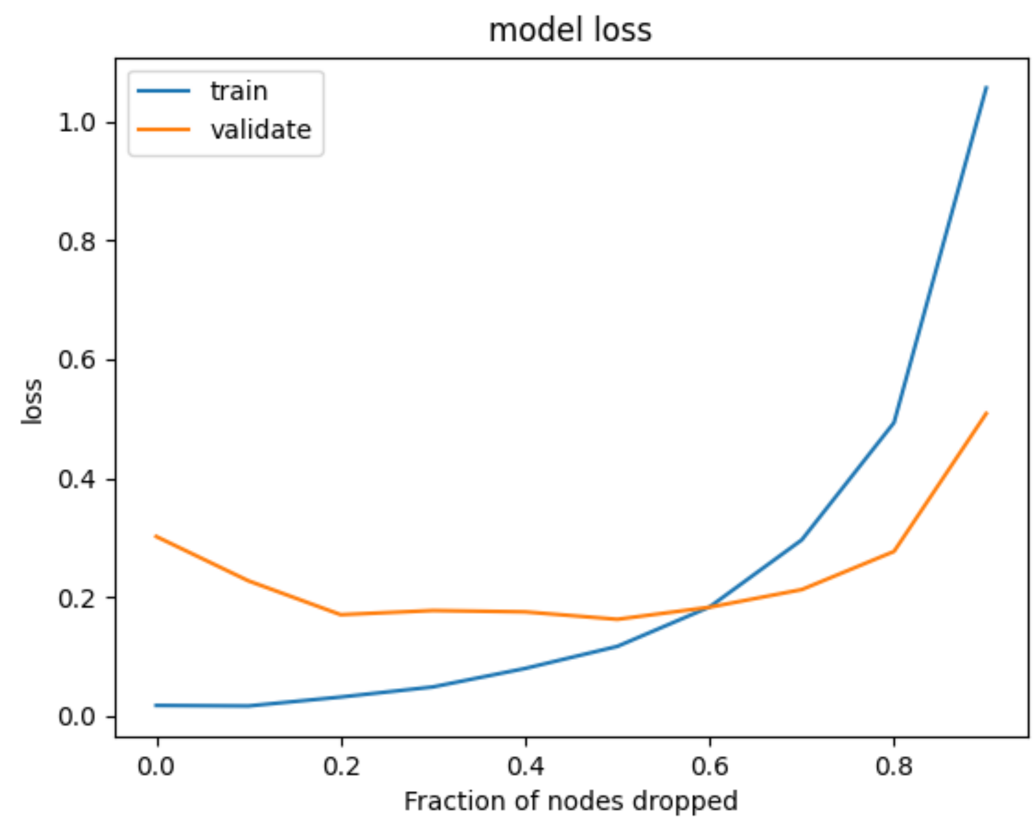


HYPER-PARAMETER TUNING: DROPOUT FRACTION

- The model hyper-parameters are not just the weights and biases in the network (for NN), the parameters chosen for the training and indeed model configuration affect model performance.

For this NN model, a large dropout fraction of ~0.6 gives consistent test and validate losses after Nepochs of training, and the test and validate accuracies are similar. i.e. the model is generalised.

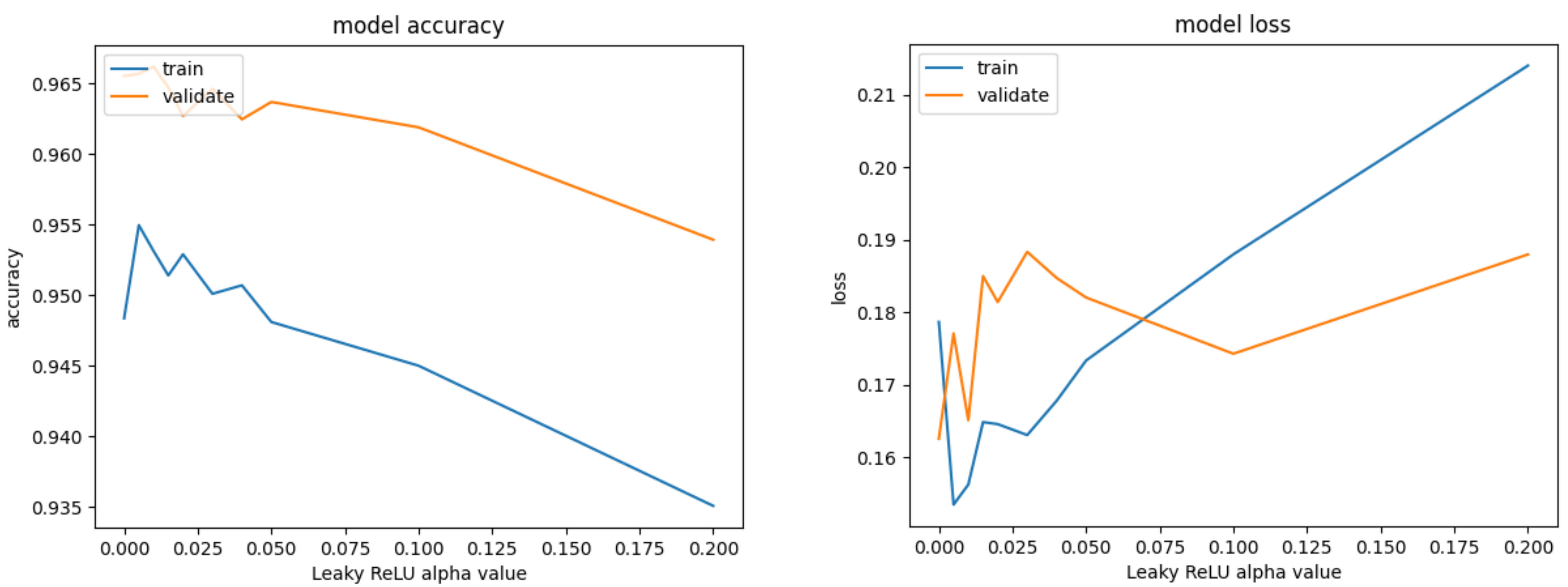




HYPER-PARAMETER TUNING: BATCH SIZE

- The model hyper-parameters are not just the weights and biases in the network (for NN), the parameters chosen for the training and indeed model configuration affect model performance.

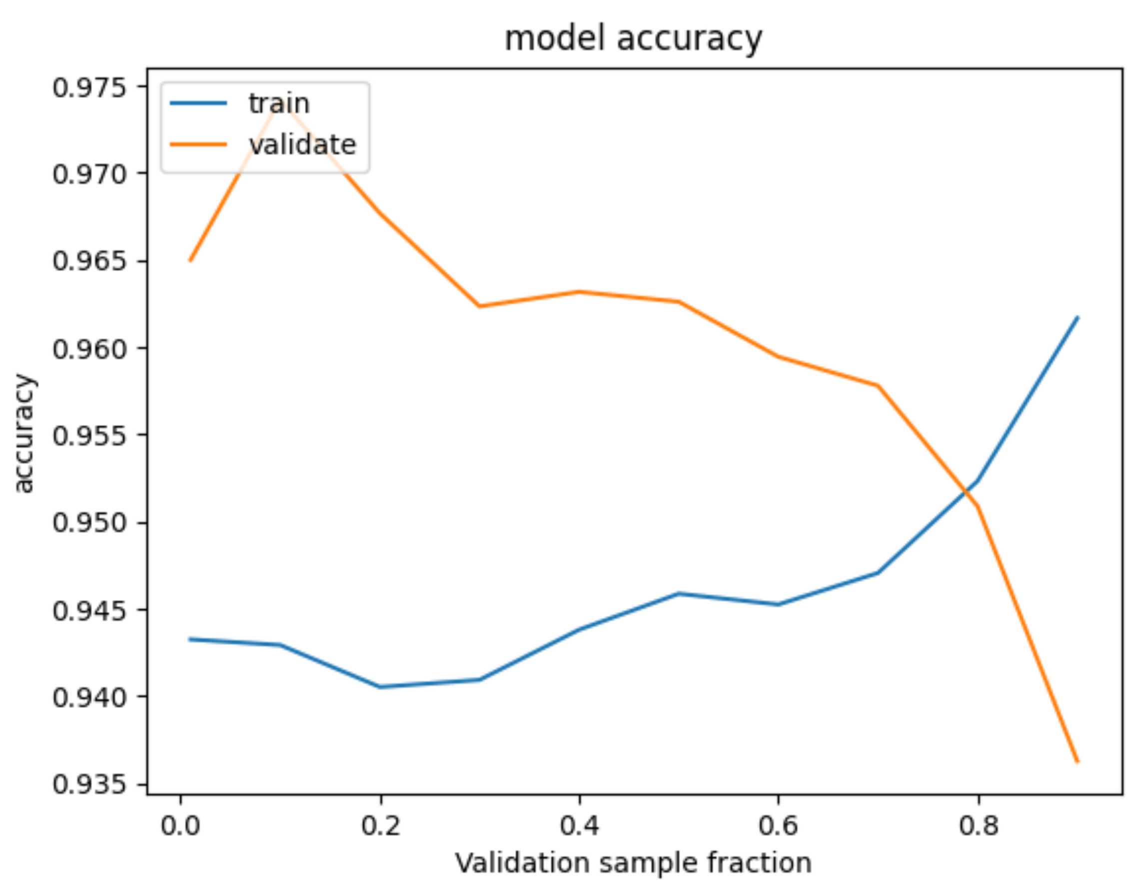
The Leaky ReLU activation alpha parameter affects model optimisation. For this example a value ~0.1 yields similar train and validate performance of the model.

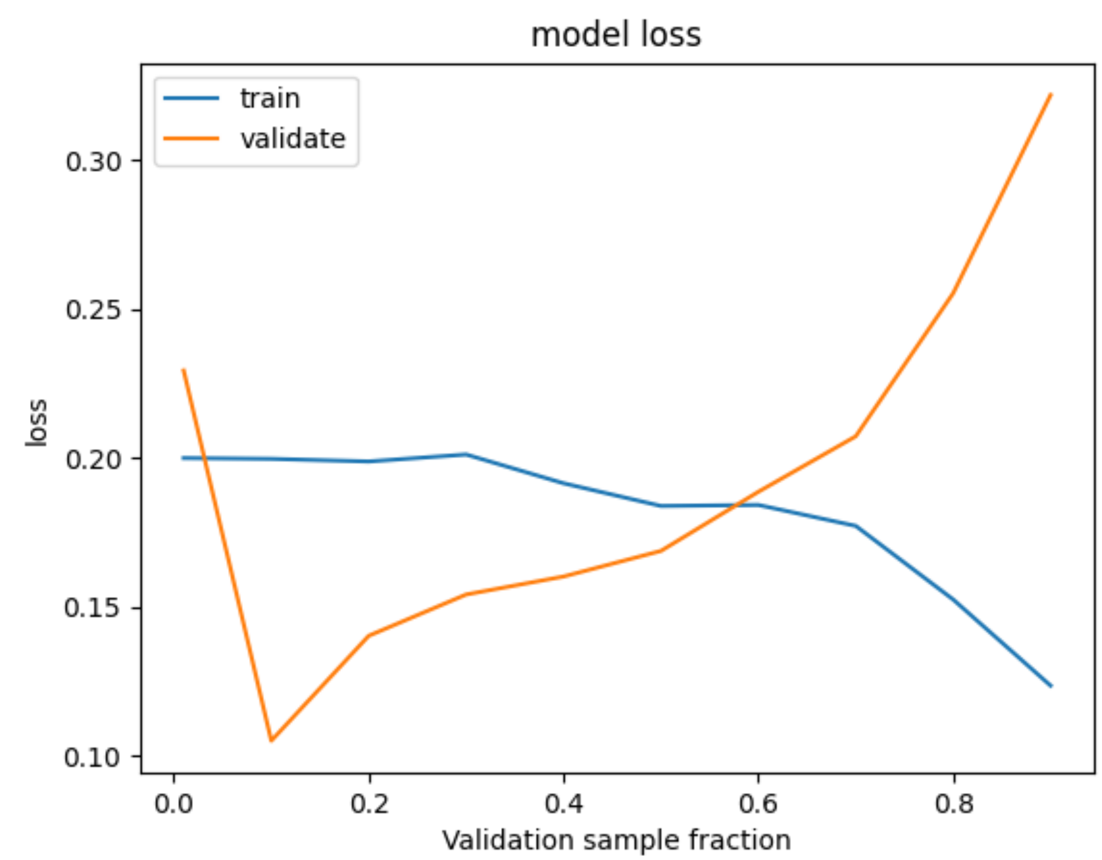


HYPER-PARAMETER TUNING: BATCH SIZE

- The model hyper-parameters are not just the weights and biases in the network (for NN), the parameters chosen for the training and indeed model configuration affect model performance.

Validation sample fraction split of ~0.6 yields similar loss function value for the train and validate samples. This shows tension between model accuracy and generalisability.





THERE ARE 2 EXAMPLES:

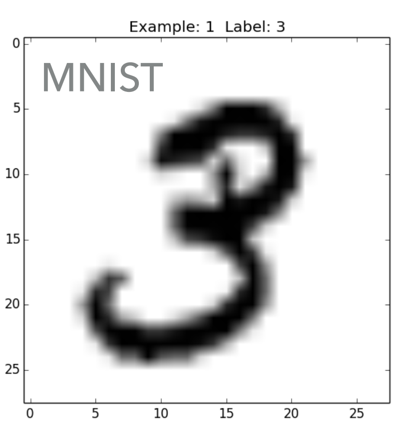
1) MNIST CLASSIFICATION PROBLEM: IDENTIFYING HAND WRITTEN NUMBERS

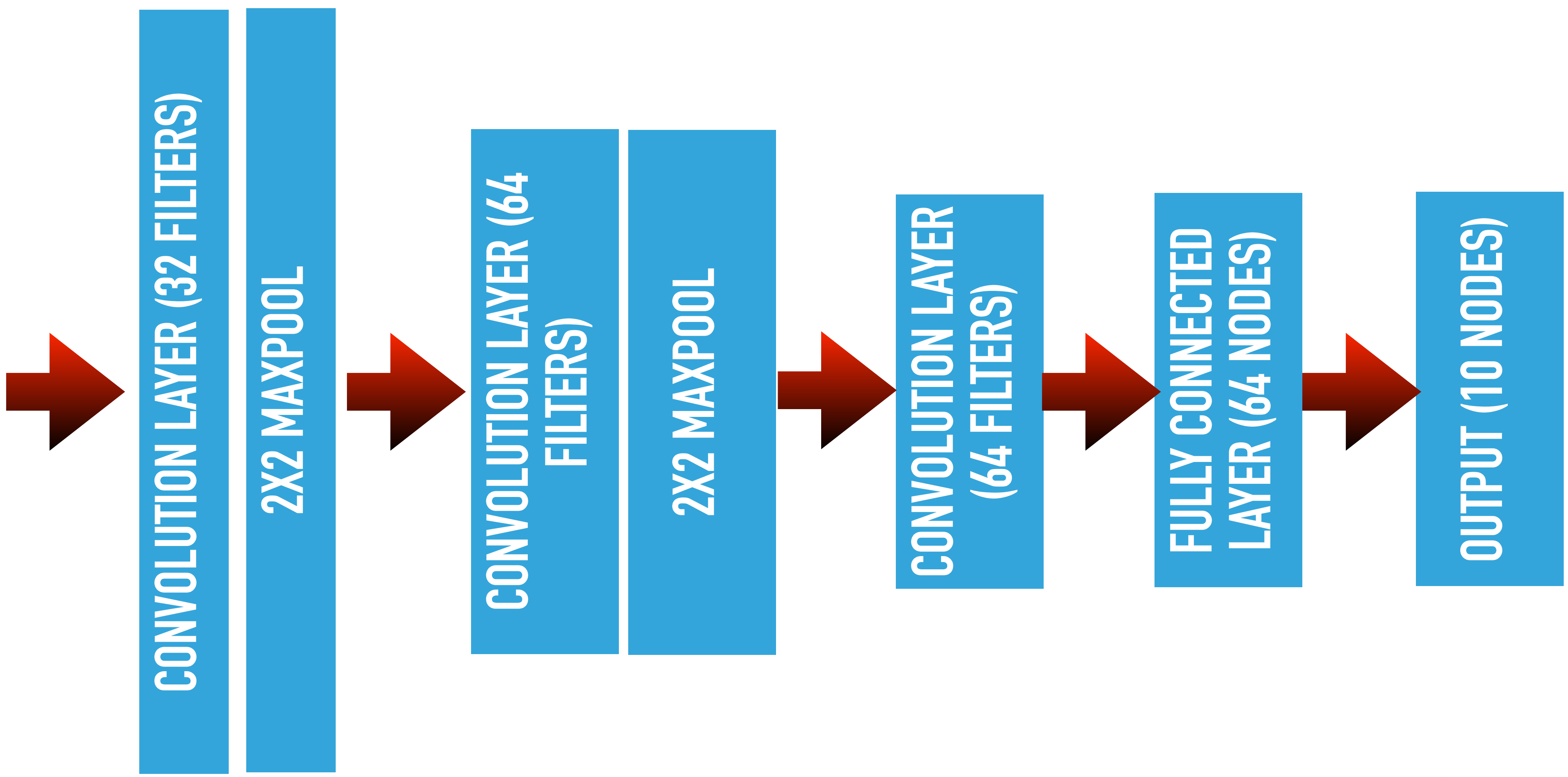
2) CFAR10 CLASSIFICATION PROBLEM: IDENTIFYING 10 DIFFERENT TYPES OF COLOUR IMAGE

CONVOLUTIONAL NEURAL NETWORKS

CNNS

- CNN.ipynb
- Use either the MNIST hand writing data set, or CFAR10 (see appendix)
- Build a CNN model using conv(olution) and maxpool layers, and finishing with a fully connected (Dense) layer.
- Use Dropout.





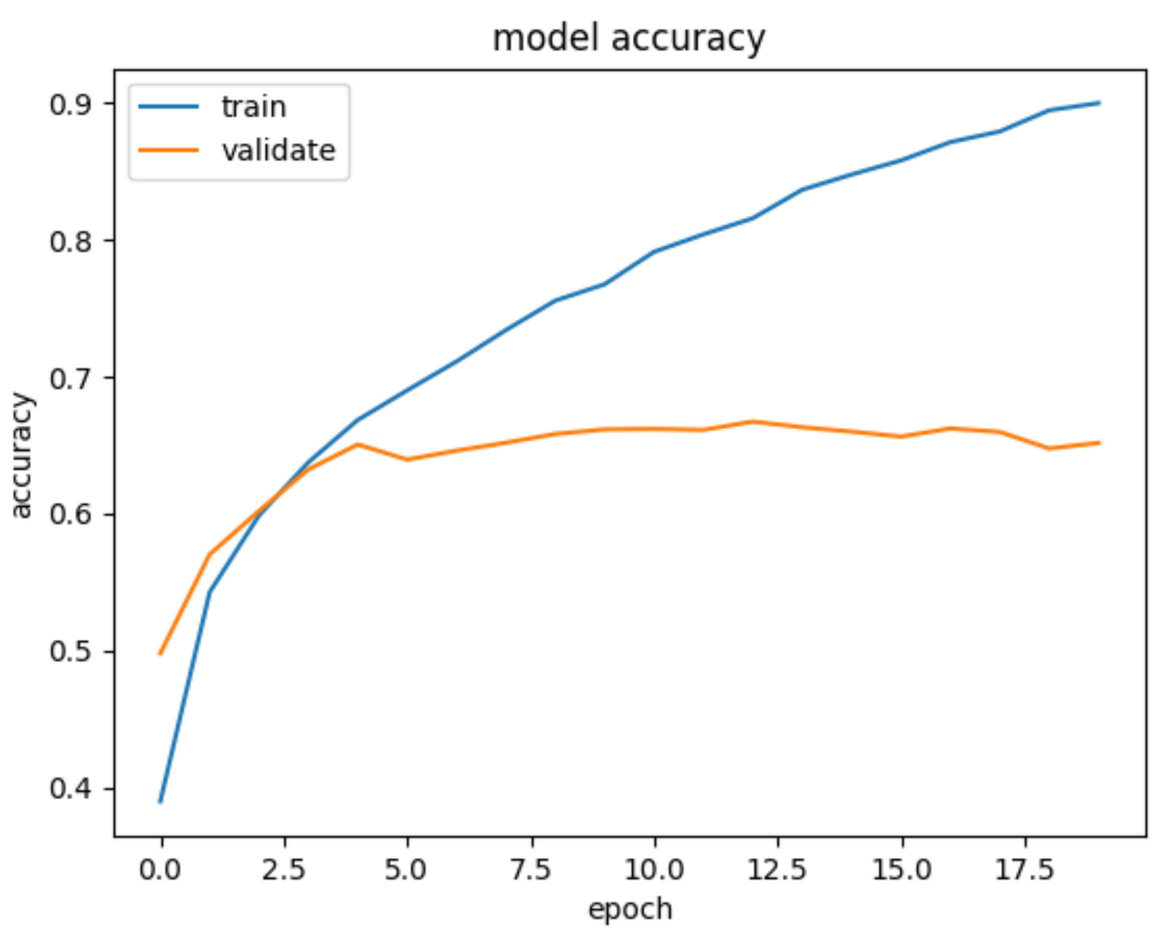
CNNS

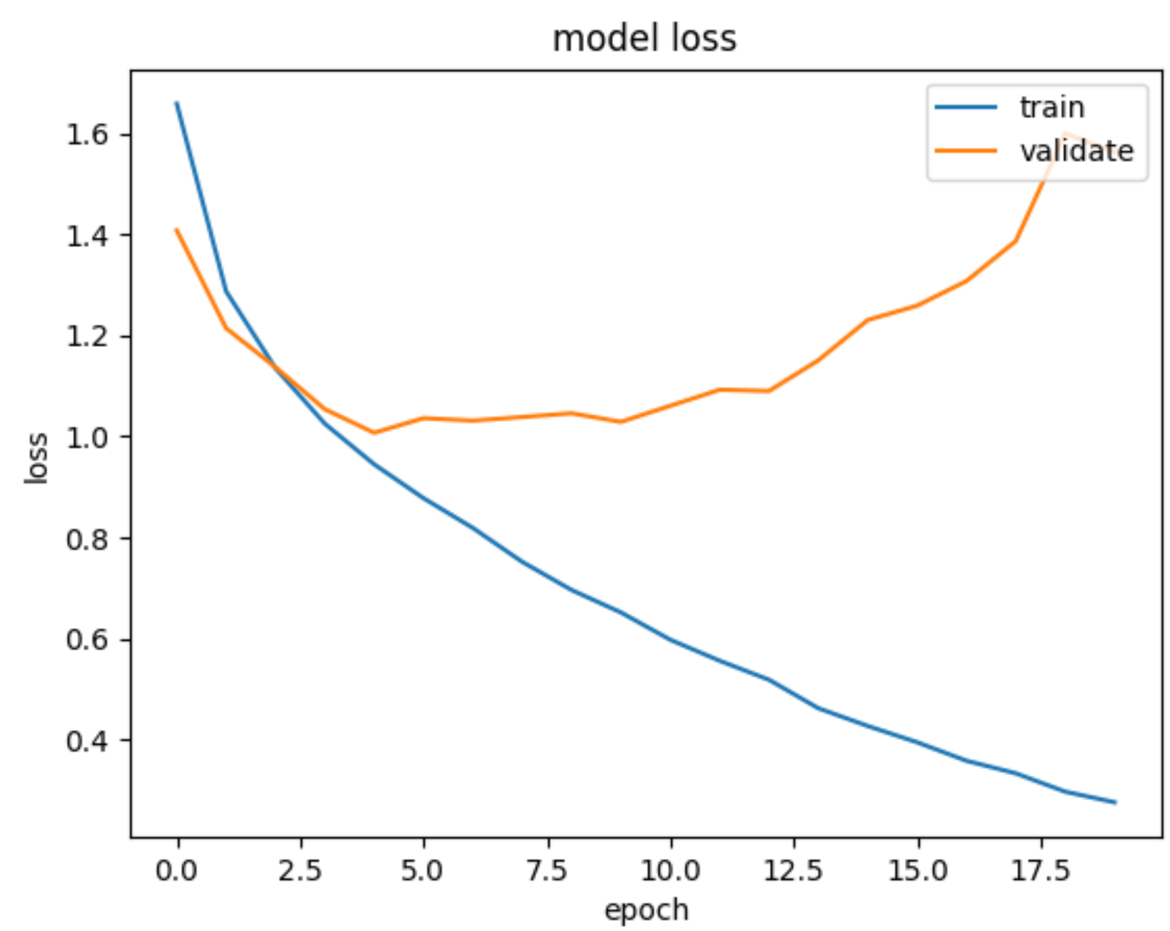
CNN training takes a while, and so you will want to continue to explore this model in your own time.

- CNN.ipynb
- Explore the effect of DropOut, ValidationSplit, Nepochs, and BatchSize have on the training (try to find a model where the test and train loss function values are similar.
- Explore how the CNN affects the training performance e.g.
  - change the number of convolution filters in each layer. The current values of these are 32, 64 and 64.
  - change the number of nodes in the fully connected (Dense) layer. The current value of nodes in this layer is 64.
- Explore the effect of adding a second dropout layers into the network after the conv and Dense layers (see the NN.ipynb example for how to implement a dense layer in a model.

CNNS

- [CNN.ipynb]
- This is an excellent example of an overtrained model.
    - The hyper-parameters set for training with these data allow for the model to be overtrained as seen by the test accuracy significantly exceeding the validate accuracy.
    - The model loss also illustrates this issue well.





If you are unfamiliar with these algorithms please see these lecture notes.

(USING SKLEARN)

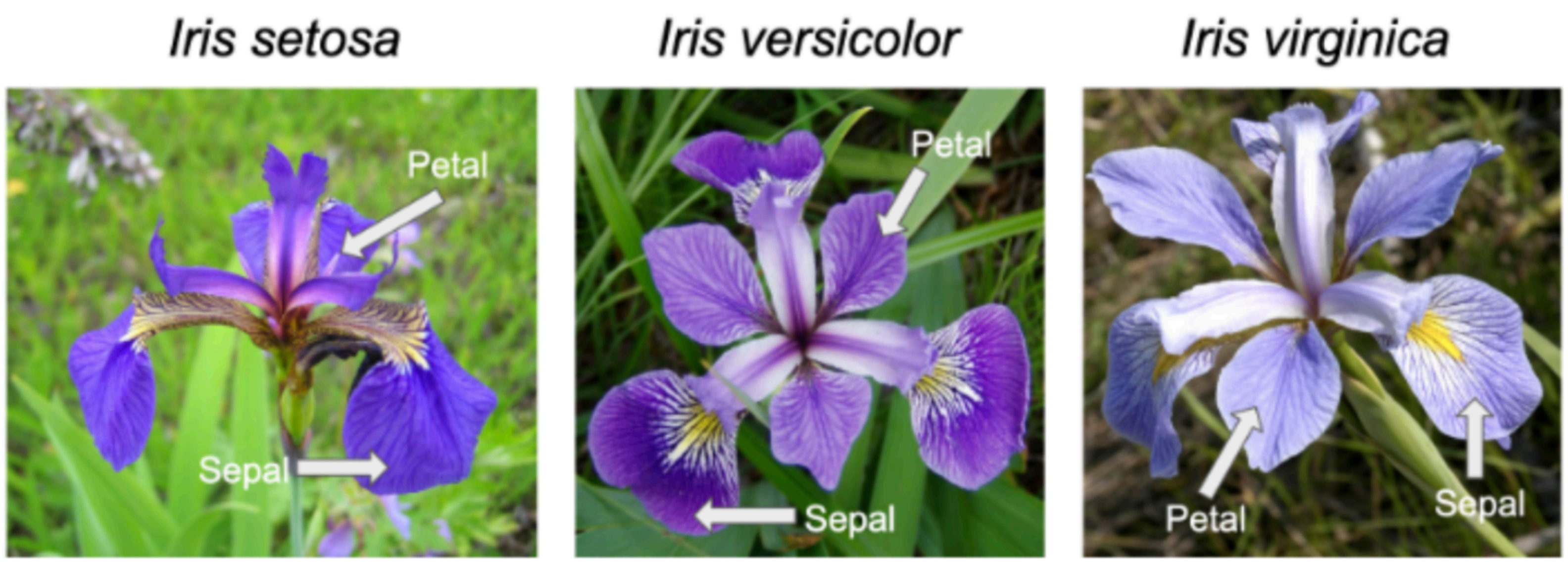
DECISION TREES AND SUPPORT VECTOR MACHINES

SCIKIT LEARN CLASSIFIERS

- SK_DT.ipynb [Decision Tree - single weak learner]
- SK_BDT.ipynb [Boosted Decision Tree - an ensemble of weak learners using the AdaBoost]
- SK_RF.ipynb [Random Forest - an ensemble of weak learners]
- SK_SVM.ipynb [Support Vector Machine]
- These scripts create classifiers to analyse a test sample of the Iris data, and to produce a plot of the confusion matrix.

SCIKIT LEARN CLASSIFIERS

- The data:



Petals & Sepals for Iris setosa, Iris versicolor, and Iris virginica (Sources: 1, 2, 3, Licenses: Public Domain, CC BY-SA 3.0 & CC BY-SA 2.0).

- 50 examples of each type of iris to be classified.
- 4 features: sepal width, sepal length, petal width and petal length.

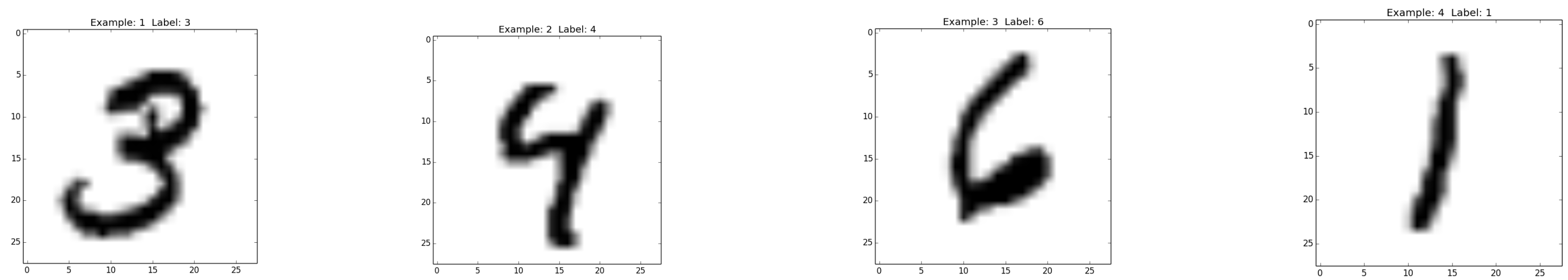
Various datasets can be found in both SciKit Learn and Keras.

DATA:
MNIST
CFAR-10
CFAR-100
KAGGLE
UCI ML DATA REPOSITORY
TIMIT
RCV1-V2

APPENDIX - SOURCES OF DATA

APPENDIX: DATA — MNIST

- MNIST is a standard data set for hand writing pattern recognition. e.g. the numbers 1, 2, 3, … 9, 0

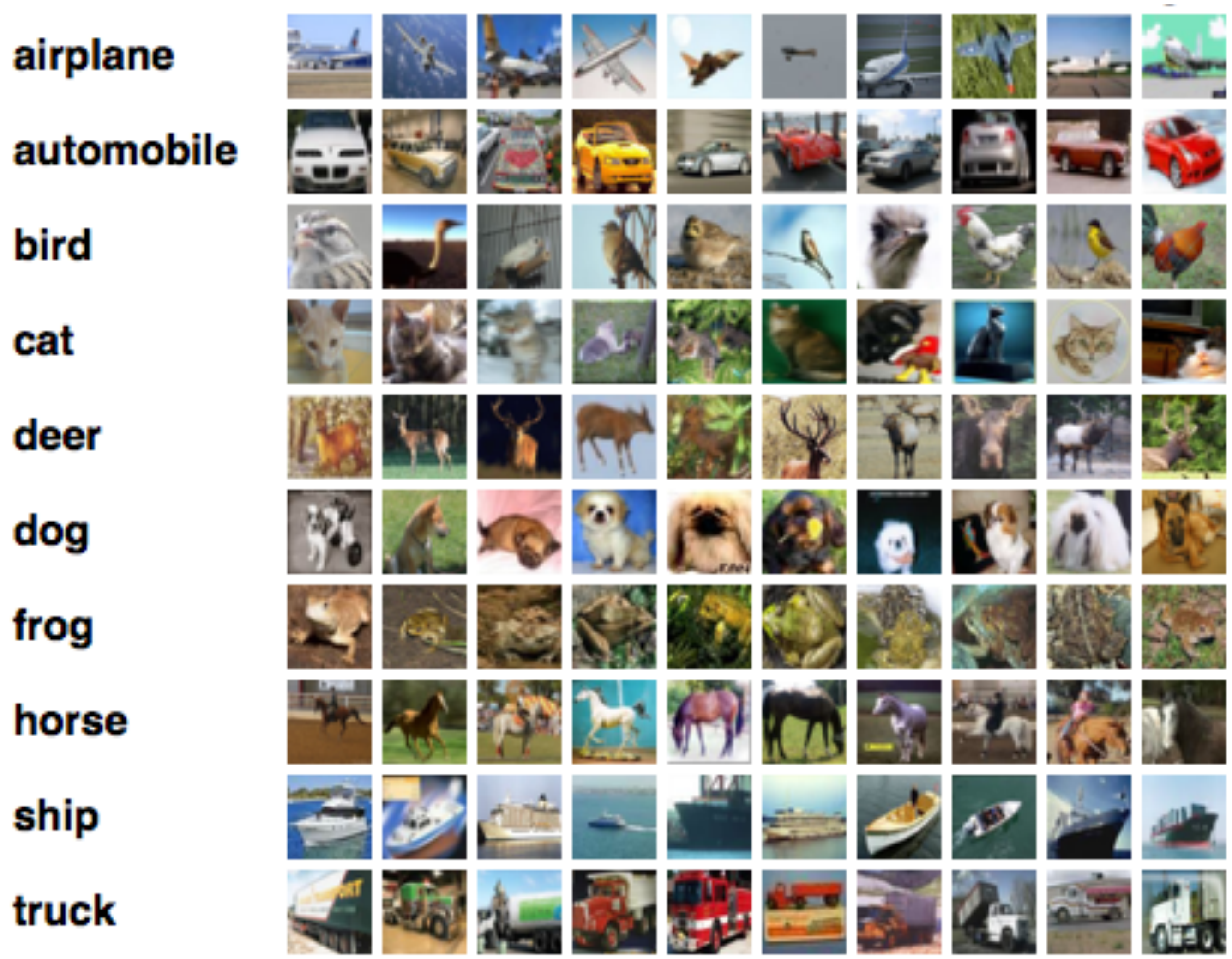


  - 60000 training examples
  - 10000 test examples
  - These are 8 bit greyscale images (one number required to represent each pixel)
  - Renormalise [0, 255] on to [0, 1] for processing.
  - Each image corresponds to a 28x28 pixel array of data.
  - For an MLP this translates to 784 features.

http://yann.lecun.com/exdb/mnist/

APPENDIX: DATA — CFAR-10

- 60k 32x32 colour images (so each image is a tensor of dimension 32x32x3).
- This is a labelled subset of an 80 million image dataset.
- 10 classes:



APPENDIX: DATA — CFAR-100

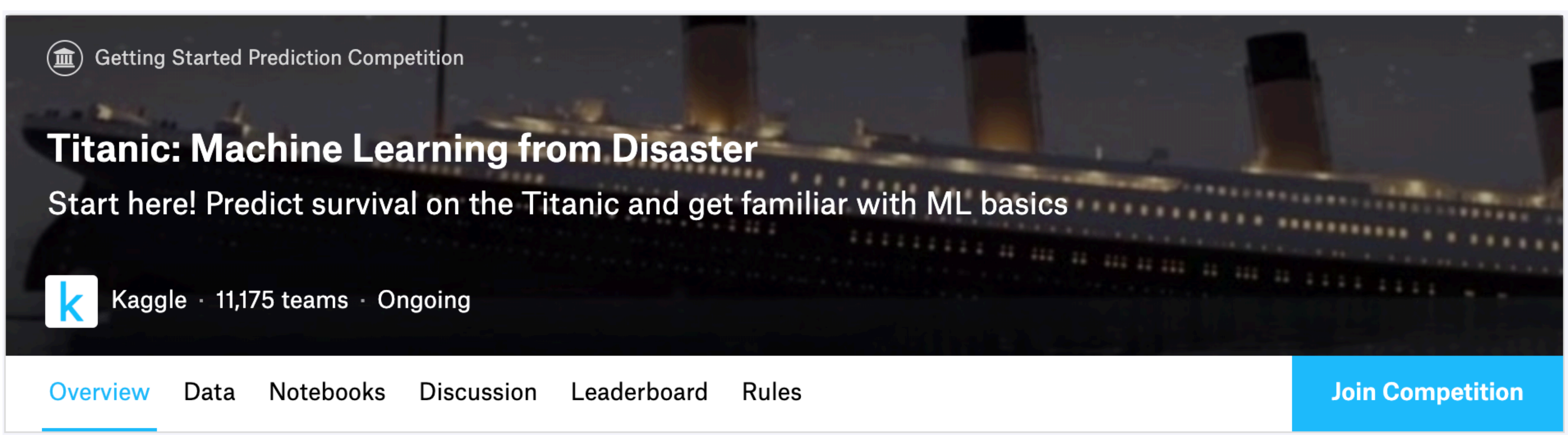
- 100 class variant on the CFAR10 sample:
- 32x32 colour images (so each image is a tensor of dimension 32x32x3).
- 100 classes:

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

https://www.cs.toronto.edu/~kriz/cifar.html

APPENDIX: DATA — KAGGLE

- Well known website for machine learning competitions; lots of problems and lots of different types of data.
- Also includes training material at:
  - https://www.kaggle.com/learn/overview
  - e.g. Intro to machine learning includes a data science problem on predicting titanic survivors from a limited feature space.
    - Since the outcome is known, this is a good sample of real world data to try out your data science skills.



APPENDIX: DATA — UCI ML DATA REPOSITORY

UCI Machine Learning Repository

- Hundreds of data sets covering life sciences, physical sciences, CS / Engineering, Social Sciences, Business, Game and other categories of data.
- Different types of problem: including Classification, regression and clustering samples.
- Different types of data: e.g. Multivariate, univariate, time-series etc.
  - https://archive.ics.uci.edu/ml/datasets.php

APPENDIX: DATA — TIMIT

- A corpus of acoustic-phonetic continuous speech data, provided with extensive documentation.
- Includes audio files and transcripts
- 630 speakers, each with 10 sentences, corresponding to a corpus of 25200 files (4 files per speaker).
- Total size is approximately 600Mb.

https://catalog.ldc.upenn.edu/LDC93S1

APPENDIX: DATA — RCV1-V2

- RCV1: A New Benchmark Collection for Text Categorization Research
- A detailed description of this text categorisation data set can be found in: http://www.jmlr.org/papers/volume5/lewis04a/lewis04a.pdf

http://www.ai.mit.edu/projects/jmlr/papers/volume5/lewis04a/lyrl2004_rcv1v2_README.htm