IWSM MENSURA
HAARLEM 7-9 OCTOBER 2019

SIZING SOFTWARE IN A MACHINE LEARNING CONTEXT: A COSMIC CASE STUDY

Arlan Lesterhuis
&
Alain Abran

LIST OF TOPICS

1. Motivation & Challenge
2. Machine Learning Case Study & Functional Requirements
3. COSMIC measurement of functional processes
4. Summary & next steps

MOTIVATION

In Machine Learning (ML) :

- A large body of literature on:
  - the mathematical aspects of the variety of machine learning (ML) algorithms &
  - analysis of their performance with various datasets, once implemented in software applications.

- But very little about the software itself that needs to be developed to implement the ML algorithms in specific industry contexts.

CHALLENGE & MOTIVATION

The literature describes the 'system viewpoint' of ML, bundling together all tasks carried on by ML researchers (including the design, coding, operation & data analysis.)

It would be valuable to segregate:

- the ML specific tasks (ML expertise is very specialized & requires considerable expertise, and is in very high demand & in short supply)

from

- those generic software development tasks of generic software (for which expertise is more widely available & less specialized).
  - E.g., the coding tasks could be delegated to staff with programming expertise.
    - ✓ freeing up the ML data analysts with more freedom to address additional ML challenges.

RESEARCH STRATEGY

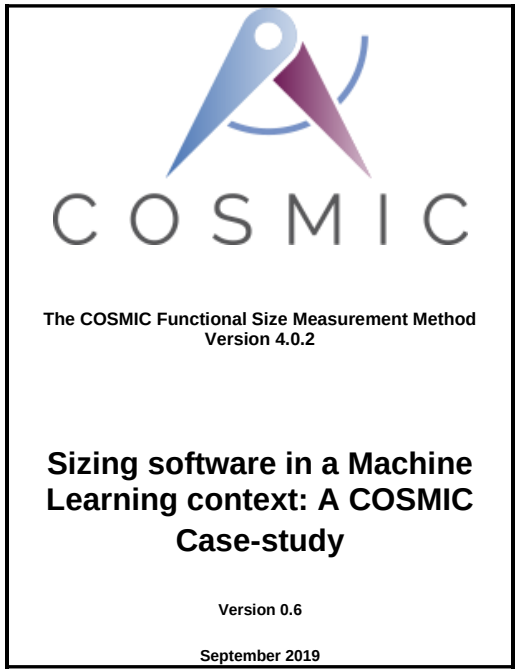
Prerequisite for such delegation:

- Untangle generic software tasks from ML specific analytical tasks:
  e.g. generic software functions must be segregated from ML specific functionalities and be documented & assigned to software developers.
- Describe their functional requirements, for delegating them to software developers.

- Success in this endeavor will allow parallelism of tasks in ML projects:
  - ✓ including shortening the development cycle.

WORK IN PROGRESS

A Machine Learning image classifier of manuscript digits



CASE STUDY – HIGH LEVEL REQUIREMENTS

1. A feedforward neural network.
2. Classification accuracy: to be equal or above 98%.
3. The network can be tuned by varying the parameters of its architecture.
4. The Image Classifier software receives the hyper-parameters (special training session parameters) from the data analyst: learning rate η, number of epochs & mini-batch size.
5. The Image Classifier software must informs the data analyst when an error has or has not occurred (but does provide details, in this case study the requirements for finding the errors are not specified).

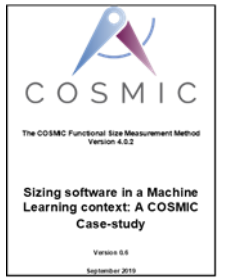
CASE STUDY: SYSTEM VIEW

System view: ML image classifier functions (1 of 3)

- A given file of images of separate individual handwritten digits each image must be classified (e.g., assigned its correct digit).
  - A file of randomly selected training images is available, each image showing the handwritten digit & the corresponding digit it represents.
    - An image of a digit consists of 28x28 pixels in greyscale, with a value of 0.0 representing white, a value of 1.0 representing black, and in between values representing shades of grey.
- Re-using a pre-programmed feedforward neural network algorithm to make it learn to classify the images of the file.
  - To initialize the network: the data analyst must specify the numbers of units per layer.
  - To initialize the values of the weights & biases the data analyst must specify the mean & standard deviation.

CASE STUDY: SYSTEM VIEW

System view: ML image classifier functions (2 of 3)

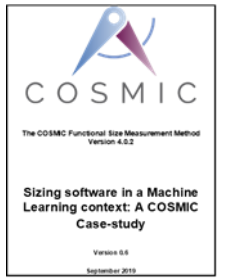


- Learning uses 3 sub-sets of the training images:
  1. Training set is used for training,
  2. Test set for testing the result of training.
  3. Validation set to determine the values of the hyper parameters:
     a) learning rate (indicated by η,
     b) the size of the mini batches to be used, and
     c) the number of epochs of training.
        - The size of the mini batches to be applied must be determined.

  All images are stored with their sub-set name ('training', 'test', or 'validation') for re-use.

CASE STUDY

System view: ML image classifier functions (3 of 3)



- The learning performance is monitored by printing & displaying:
  - the classification accuracy & displaying the cost (error) per epoch of training.
- The training set is enlarged by adding one elastically distorted copy of each image in the training set.
- For acceptance by the client the classification accuracy is required to be not less than 98%, verified on the basis of the test set.
- It must be possible to tune the network,
  - i.e. investigate the accuracy & speed by varying the main parameters of the network structure (number of its layers, number of units per layer and hyper-parameters to be applied).

1st Measurement Step

Identification of the Functional users of the generic software to be measured:
(e.g., the functional view of the system requirements)

1. The data analysts of the neural network:
   - those who tune the network so as to meet the accuracy requirement.

2. The reused neural network algorithm (e.g., feedforward algorithm in Fig. 1):
   - Therefore, it does not have to be measured in the measurement of the Case Study.

1ST MEASUREMENT STEP & CONTEXT DIAGRAM

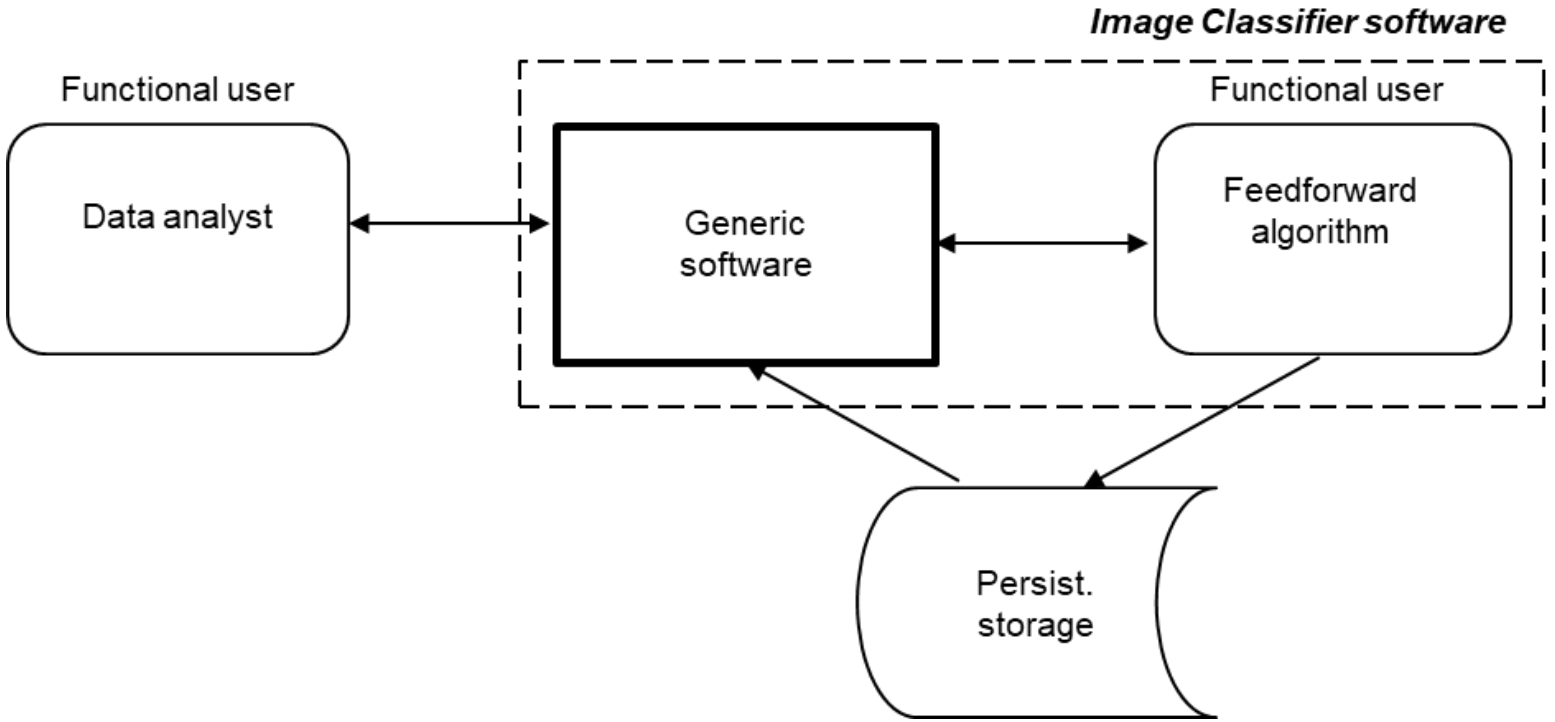
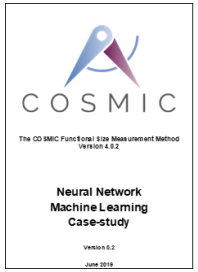


Figure 1: Context diagram of the generic software



CASE STUDY CONTEXT

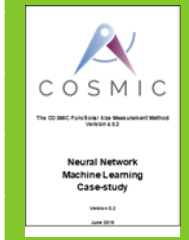
Functional reuse of a pre-programmed feedforward neural network algorithm (including its cost function).

- Consequently, to create the feedforward neural network the data analyst needs only specify a sequence of numbers in which:
  - its length indicates the number of layers,
  - each number indicates the number of units in its layer, and
  - each unit in the layers (except the input layer) has one weight per input & one bias.

- The feedforward algorithm software receives the training parameters, including the hyper parameters.

LIST OF FUNCTIONAL PROCESSES

ID	Functional Process Name
FP1	Initialize the feedforward network architecture
FP2	Expand the images
FP3	Divide images into 3 sub-sets
FP4	Display cost per epoch (Fig. 2)
FP5	Display classification accuracy per epoch (Fig. 3)
FP6	Determine mini-batch size (Fig. 4)
FP7	Train the network
Total Size	



FUNCTIONAL PROCESS 1: INITIALIZE NETWORK ARCHITECTURE

FP1: The network architecture is initialized by the parameters of the neural network architecture specified by the data analyst.

FP 1: Initialize a feedforward network architecture. Size - 4 CFP

DM (data movement)	Data group/ data attributes
Entry	Number of units from data analyst
Exit	Number of units to feedforward algorithm
Entry	Result of initialization from feedforward algorithm
Exit	Error/confirmation message to data analyst

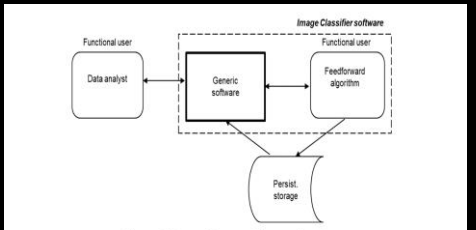


Figure 1: Context diagram of the generic software

FUNCTIONAL PROCESS 2: EXPAND THE IMAGES

Functional process 2 – FP2: Expand the images

The data analyst inputs the required expansion instruction, then this functional process copies each image, distorts it and adds the result to the training set. **Size = 4 CFP.**

DM	Data group/ data attributes
Entry	Expansion instruction
Read	Image data
Write	Training image
Exit	Error/confirmation message

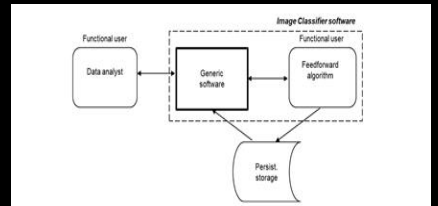


Figure 1: Context diagram of the generic software

FUNCITONAL PROCESS 3: DIVIDE THE IMAGES INTO 3 SUBSETS

Functional process 3 – FP3: Divide the images into three sub-sets

The data analyst inputs the number of images within each set, then this functional process divides the images into the three sub-sets of training, test and validation images, adds the sub-set name attribute value to each image and stores the image data. **Size = 4 CFP.**

DM	Data group/ data attributes
Entry	Sub-set of images (sub-set name, number of images)
Read	Image data
Write	Image data with sub-set name
Exit	Error/confirmation message

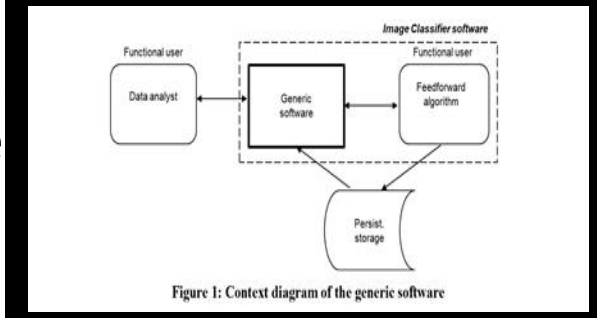


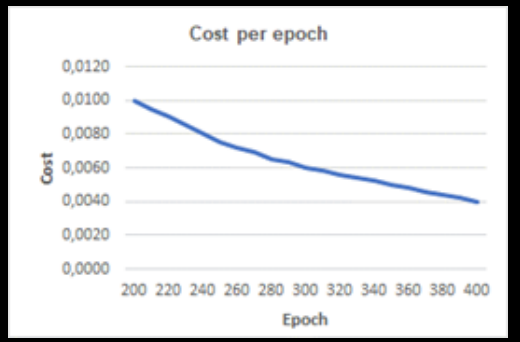
Figure 1: Context diagram of the generic software

FUNCTIONAL PROCESS 4: DISPLAY COST PER EPOCH

Functional process 4 – FP4: Display cost per epoch (Figure 2)

The data analyst requests to display the graph of the cost ('error') of (the last mini-batch of) each epoch. **Size = 6 CFP.**

DM	Data group/ data attributes
Entry	Epoch ID range
Read	Epoch ID, epoch cost stored by the feedforward algorithm
Exit	Epoch ID (x-axis, multiples of 50)
Exit	Cost (y-axis, multiples of 0,001)
Exit	Epoch cost
Exit	Error/confirmation message

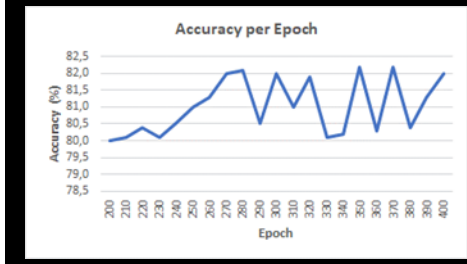


FUNCTIONAL PROCESS 5: DISPLAY CLASSIFICATION ACCURACY PER EPOCH

Functional process 5 – FP5: Display classification accuracy per epoch (Figure 3)

The data analyst requests to display the graph of the classification accuracy of the images. **Size = 6 CFP.**

DM	Data group/ data attributes
Entry	Epoch ID range
Read	Epoch ID, epoch accuracy stored by the feedforward algorithm
Exit	Epoch ID (x-axis, multiples of 10)
Exit	Epoch accuracy (multiples of 0.5%)
Exit	Epoch ID, epoch accuracy
Exit	Error/confirmation message



FUNCTIONAL PROCESS 6: DETERMINE MINI-BATCH SIZE

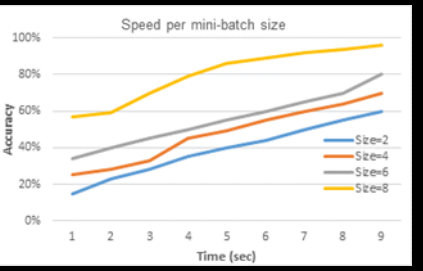
Functional process 6- FP6: Determine mini-batch size (Figure 4)

The data analyst inputs a number of epochs and a number of mini-batch sizes to be examined and determines the desired mini-batch size visually on the basis of the graph. The software:

- executes the neural network algorithm with the validation data,
- graphically plots the last known epoch accuracy at each point of time.

Size = 10 CFP

DM	Data group/ data attributes
Entry	Training session parameters (without number of images per mini-batch)
Exit	Training session parameters (without number of images per mini-batch) to feedforward algorithm
Entry	Mini-batch size, input the sizes to be compared
Exit	Mini-batch size, sizes to be compared to feedforward algorithm
Read	Elapsed time, epoch accuracy per mini-batch size stored by the feedforward algorithm
Exit	Elapsed time (x-axis, in seconds)
Exit	Epoch accuracy (y-axis: multiples of 20%)
Exit	Mini-badge denotation from entry above
Exit	Epoch accuracy per mini-batch size at point of time
Exit	Error/confirmation message

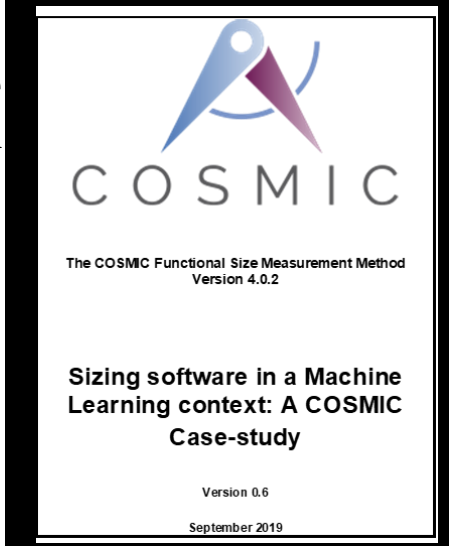


FUNCTIONAL PROCESS 6: TRAIN THE NETWORK

Functional process 7 – FP7: Train the network

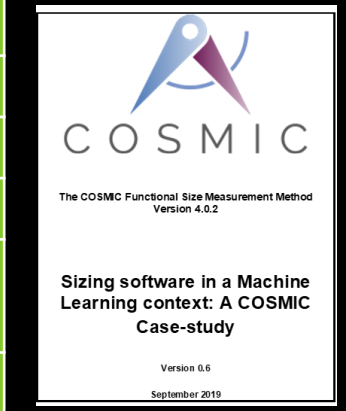
The data analyst inputs the training session parameters (mean, standard deviation, number of epochs, mini-batch size, learning rate η, regularization parameter λ) to train the network. For monitoring the training the epoch ID, number of correctly classified images, total number of images and elapsed time must be printed. **Size - 5 CFP.**

DM	Data group/ data attributes
Entry	Training session parameters (mean, standard deviation, number of epochs, number of images per mini-batch, learning rate (η), regularization parameter (λ), sub-set name)
Exit	Training session parameters to feedforward algorithm, sub-set name
Entry	Epoch ID, number of correctly classified images, total number of images, elapsed time from feedforward algorithm
Exit	Print epoch ID, number of correctly classified images, total number of images, elapsed time
Exit	Error/confirmation message



CASE STUDY SIZE IN COSMIC FUNCTION POINTS

ID	Functional Process Name	Size in CFP
FP1	Initialize the feedforward network architecture	4
FP2	Expand the images	4
FP3	Divide images into 3 sub-sets	4
FP4	Display cost per epoch (Fig. 2)	6
FP5	Display classification accuracy per epoch (Fig. 3)	6
FP6	Determine mini-batch size (Fig. 4)	10
FP7	Train the network	5
Total Size		**39 CFP**



SUMMARY

ML image classifier case study:

- Used to illustrate how to move from a 'system viewpoint' of ML functionalities to the description of the generic software development tasks for which expertise is more widely available & less specialized.

COSMIC function points – ISO 19761:

- ✓Used to address 2 objectives:
  1. Segregation of the generic functionality to be allocated to software & not specific to ML.
  2. Use of the international standard to size the generic functionality identified.

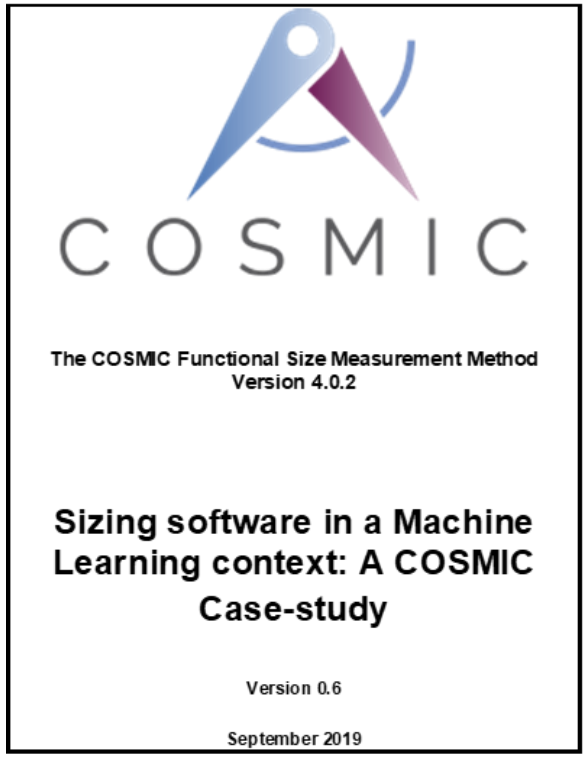
NEXT STEPS

1. Additional steps to verify the breadth & depth of the generic software functions described in sets of ML requirements.
2. Identification of ambiguities & updates with corresponding size adjustments.
3. Further validation, including verification with actual ML software already developed by industry.
4. More case studies from other domains may provide additional types & sources of generic functionality that could then be considered for scaling purposes.

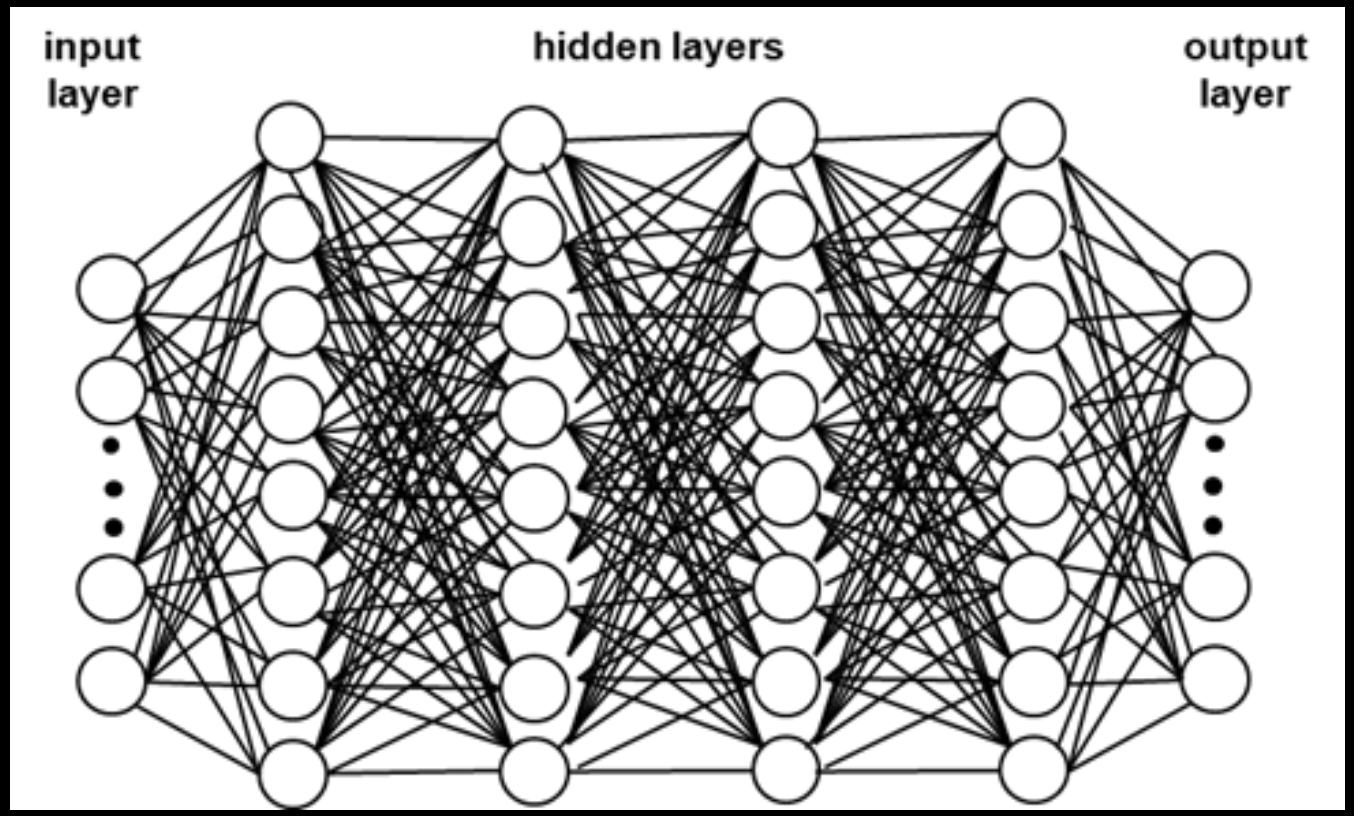
Sizing software in a Machine Learning context: A COSMIC case study

Q & A ?

CASE STUDY – TABLE OF CONTENTS



ARCHITECTURE OF FEED-FORRWARD NEURAL NETWORK



THE COST FUNCTION IN NEURAL NETWORKS

During training, a so-called cost function C of the neural network quantifies the average over the error of all individual training examples in a mini-batch.

Example of a cost function is $C(w,b) = (1/2n).\sum x(y(x)-a(x))2$, where:

- n = the number of training examples in a mini-batch,
- x = a training example,
- y(x) = desired output,
- a(x) = actual output
- w = weights in the hidden layers, and
- b = the biases in the hidden layers.

- During training each training example is input together with its desired value, the latter being used to determine the error between desired and actual output.

RREQUIREMENT 2: DETERMINING A SUITABLE NUMBER OF EPOCHS

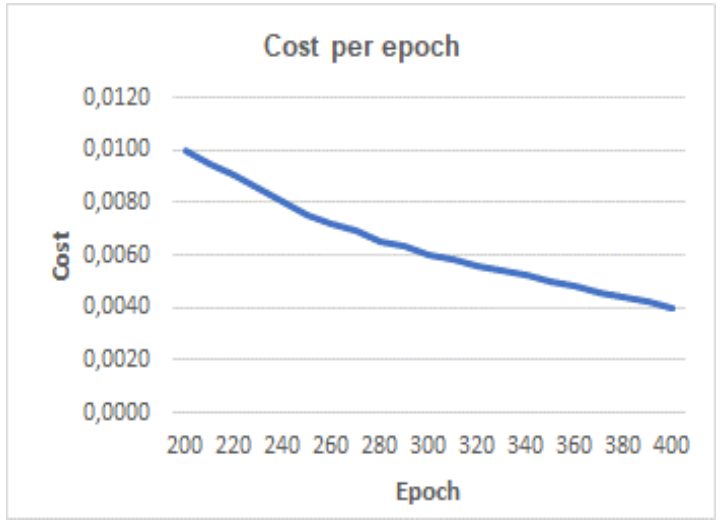


Figure 2: Cost per epoch

REQUIREMENTS 2: DETERMINING A SUITABLE NUMBER OF EPOCHS

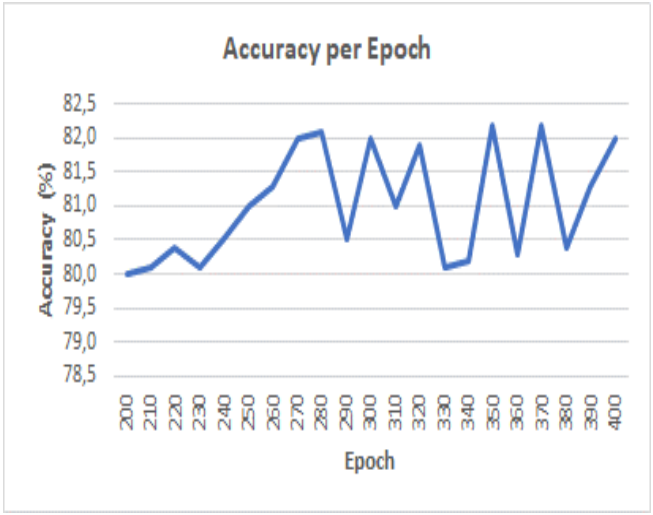


Figure 3: Accuracy per epoch

REQUIREMENT 2: PREPARE TRAINING – DETERMINE LEARNING RAGE

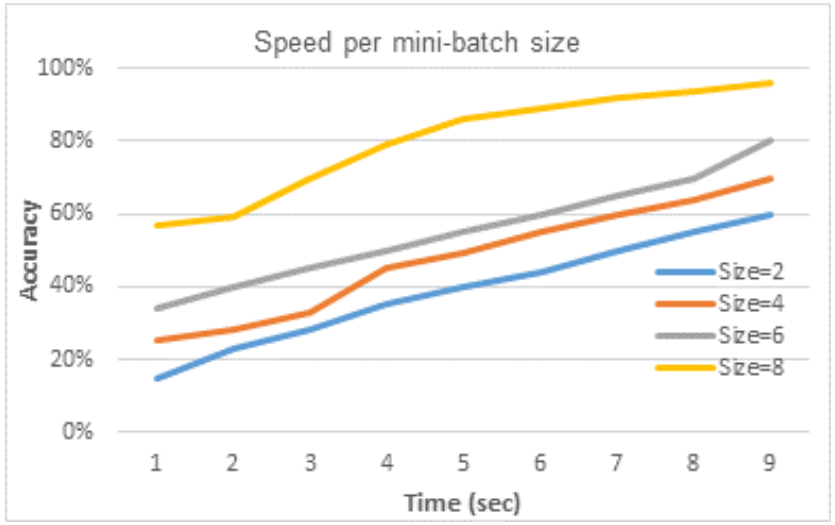


Figure 4: Speed per mini-batch size