LavaNet - a neural network modelling toolkit for mine planning and environmental modelling

I.K. Kapageridis
Laboratory of Mining Information Technology and GIS Applications, Department of Geotechnology and Environmental Engineering, Technological Educational Institute of Western Macedonia, Greece

ABSTRACT

LavaNet consists of a series of scripts written in Perl that give access to a neural network simulation environment inside a general mine planning package. LavaNet enables easy development of neural network training datasets using information from any of the data and model structures available, such as block models and drillhole databases. Neural networks can be trained inside the mine planning environment and the results be used to generate new models that can be visualised in 3D. Direct comparison of developed neural network models with conventional and geostatistical techniques is now possible within the same environment. LavaNet supports Radial Basis Function networks, Multi-Layered Perceptrons and Self-Organised Maps. The flexibility of LavaNet in developing neural network based models and its integration with the mine planning environment is demonstrated using two different examples: a geological structure model and a 3D block model attribute estimation.

1. INTRODUCTION

A number of mining planning related problems have been approached using artificial neural network (ANN) technology in the last couple of decades. These problems commonly relate to pattern classification, prediction and optimization. The general trend in the mining industry for automation to the greatest degree calls for technologies such as ANNs that can utilize large amounts of data for the development of models which, otherwise, are very difficult or sometimes even impossible to construct.

One particular area were ANNs have been applied in the mine planning sector is in spatial analysis problems (Kapageridis, 2002). Exploration and resource estimation commonly involves the prediction of various parameters characterizing a mineral deposit or a reservoir. Input data usually come in the form of samples with known positions in 3D space. The most common practice when developing training patterns sets for an ANN, is to generate input-output pairs with the input being the sample location and the desired output being the value of the modeled parameter at that location. Some other systems go a step further to exploit information hidden in the relationship between neighboring samples (Kapageridis, 1999, 2005). The estimation of a parameter at a specific location in 3D space is, in this case, depending on information from samples around that location.

Regardless of the approach, a common obstacle in developing neural network solutions to mining problems is the development of appropriate data sets that can be used for neural network training as well as the transfer of neural network application results back into the mine planning process. The neural network development environment described in this paper addresses these problems and provides a platform for fast and comprehensive development of neural network models within a general mine planning package.

2. LAVA SCRIPTING AND THE STUTTGART NEURAL NETWORK SIMULATOR

2.1 Integration with Vulcan

It is very common in mine planning packages to

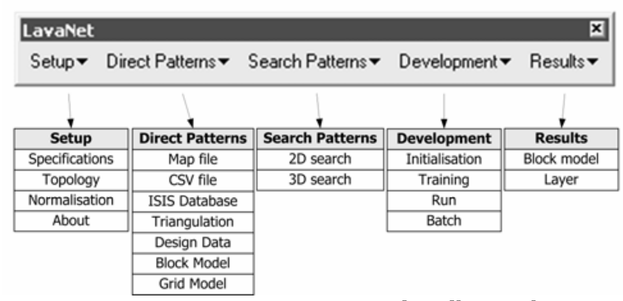


Figure 1: LavaNet menu structure and toolbar within Envisage.

integrate a scripting language that allows automation of repetitive tasks and the extension of built-in functionality of the software. Lava is a module for a version of the popular scripting language Perl that is built into VULCAN™.

LavaNet consists of a number of scripts that, once copied in the VULCAN™ installation directory, can be accessed as a menu option or through the provided toolbar. The scripts run and behave as any other standard option and the user has no feeling of the script compilation process that takes place in the background. The toolbar groups LavaNet options depending on their purpose as shown in Figure 1. The user can determine the project name and network topology architecture through the setup options. Training, validation and application patterns can be generated through the Patterns menus using any of the data and model structures used by VULCAN™. Development of the network follows, split into multiple stages that can be combined in a single batch development script that can be used more than once. Finally, the results from running a developed network can be imported through the Results menu.

2.2 Interface to SNNS

The Stuttgart Neural Network Simulator (SNNS) is a well developed and complete environment that has been around since 1990. It is a multi platform package that allows development of neural network systems using a wide variety of topology architectures and training algorithms. A Java based version of SNNS called JavaNNS is also available. The original X-Windows version comes complete with a batch development language called Batchman. The LavaNet interface presented in this paper utilizes this tool for the development and application of neural network systems from within VULCAN™ (Fig. 2). The current (4.3) and previous versions of SNNS can be downloaded for free and be used on different operating systems.

3. LAVANET OPERATION

3.1 Network Topology Specification

LavaNet is based on SNNS and, therefore, is using the particular network structures and algorithms provided by this simulator. Currently three network models are implemented in LavaNet, Multi-Layer Perceptrons (MLP), Radial Basis Function Networks (RBFN), and Self Organizing Maps (SOM). Networks can be designed with particular input, hidden and output nodes, and specific initialization, learning and updating functions as in SNNS.

Network design panels in LavaNet are meant to be complimentary to the original SNNS environment - LavaNet can receive any SNNS compatible network file and use it for development and application. Network and pattern files developed through LavaNet can also be examined, analysed and further developed in SNNS or JavaNNS. All Batchman and SNNS messages are shown in Vulcan’s report window during LavaNet operation.

3.2 Data Sources

Probably the most important aspect of LavaNet is the pattern generation options. One of the common time consuming issues in neural network development is the generation of training and validation patterns from existing data sources such as drillhole databases, triangulation models, block models, formatted data files, and vector data. LavaNet can handle any of the available data sources in VULCAN™ and practically cover any possible source of information in a general mine planning package.

Each data source, whether it is a database or a model, normally hosts a number of variables or parameters, such as coordinates, assay values, rock codes, etc. These can be numerical or alphanumerical depending on what the variable represents. Using LavaNet Direct Patterns options, and depending on the number of inputs and outputs defined through network setup, any of the available variables from a particular data source can be assigned as an input or an output. Certain data sources have a very specific dimensionality, such as grid models where only three

Table 1: Data sources and available parameters for network inputs and outputs.

Data source	Inputs description	Inputs count	Outputs description	Outputs count
MAP file	Sample co-ordinates, sample volume, assays, codes, etc.	Un-limited	Sample co-ordinates, sample volume, assays, codes, etc.	Un-limited
CSV file	Sample co-ordinates, sample volume, assays, codes, etc.	Un-limited	Sample co-ordinates, sample volume, assays, codes, etc.	Un-limited
ISIS database	Sample co-ordinates, sample volume, assays, codes, etc.	Un-limited	Sample co-ordinates, sample volume, assays, codes, etc.	Un-limited
Triangulation model	Node X,Y coordinates	2	Node Z co-ordinate	1
Block model	Block centroid coordinates, volume, variables	Un-limited	Block centroid coordinates, volume, variables	Un-limited
Grid model	Node X,Y coordinates	2	Node Z co-ordinate	1
Vector data	Point X,Y or X,Y,Z coordinates	2 or 3	Point Z, W* value	1

parameters are available - the two coordinates for each node and its value (Table 1). This must be considered before defining the network architecture. VULCAN™ provides a number of ways to combine information from multiple sources in a single model or file, so data source variable limitations should not be a problem for users of LavaNet.

LavaNet can also generate pattern files for network development and application through searching in 2D or 3D space for sample pairs. Search patterns can be generated by searching the data source for sample pairs in the space defined by user selectable dimensions.

3.3 Direct Patterns

Direct patterns can be formed by converting the information contained in a layer, triangulation, grid or any data source into patterns. In case of a triangulation, for example, each node will produce a pattern consisting of three parameters - the node coordinates. Information from multiple models or databases can be combined using existing VULCAN™ functionality. Typically, a database will be the source of information for the generation of training and validation patters, while a model structure can provide the desired application patterns. It is important, of course, that both database and model have the information to produce the same input and output space.

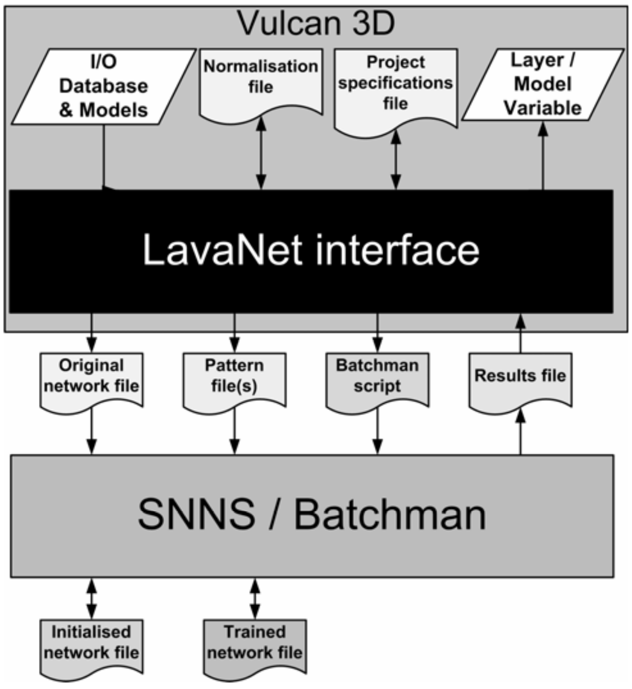
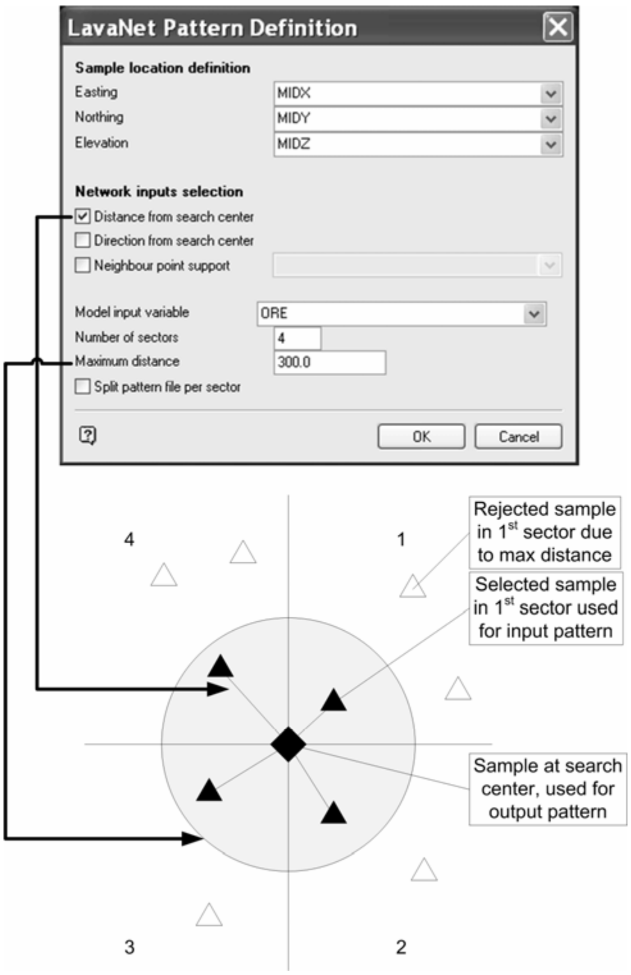


Figure 2: Information flow between Vulcan, LavaNet and SNNS during neural network development and application.

3.4 Search Patterns

In addition to direct patterns, LavaNet can help explore the relationship between database samples and/or model nodes through the calculation of distance and direction between them, either in two or three dimensional space (Fig. 3). The actual dimensions used in calculating distance and direction between samples or nodes can be anything from standard coordinates to assay values. In other words, searching can be performed in spaces other than the 2D or 3D coordinates space.



LavaNet Pattern Definition

Sample location definition

Easting	MIDX
Northing	MIDY
Elevation	MIDZ

Network inputs selection

[✓] Distance from search center
[ ] Direction from search center
[ ] Neighbour point support

Model input variable	ORE
Number of sectors	4
Maximum distance	300.0

[ ] Split pattern file per sector

OK Cancel

4
1
Rejected sample in 1st sector due to max distance
Selected sample in 1st sector used for input pattern
Sample at search center, used for output pattern
3
2

Figure 3: Search pattern panel options and effects.

3.5 Normalisation

LavaNet has the ability to normalize data between an upper and lower bound. Each sample of data is multiplied by an *Amplitude* value and shifted by an *Offset* value. The amplitude and offset are referred to as normalization coefficients. These coefficients are computed per input and output, meaning that there is a unique amplitude and offset value for each input and output. The coefficients are stored in a normalization file named after the pattern file name and a .txt extension within the working area directory of Vulcan.

The normalization coefficients are computed based on the *Minimum* and *Maximum* values found across all data in the selected data source. As one data source may be used for training and validation, and another be used for running the trained network, it is important to generate and use the normalization file from the data source with the greater extents in the given input and output space. For example, if a drillhole database is used as the data source for training and validation, and a block model that includes all drillholes is used for running the network then the block model normalization file should be used to normalize training, validation and run pattern files. This will guarantee that all of the patterns of the selected data sources will fall between the upper and lower bounds. The normalization file is also used by LavaNet to denormalize the network data, i.e. to put it in terms of the original data. Normalization coefficients are calculated using the following formulae:

$$Amplitude(i) = (UpperBound - LowerBound) / (Maximum(i) - Minimum(i))$$

$$Offset(i) = UpperBound - Amplitude(i) * Maximum(i)$$

The values 0.1 and 0.9 are used as the Upper and Lower bounds. The pattern generation options normalize and de-normalise data using the following formulae:

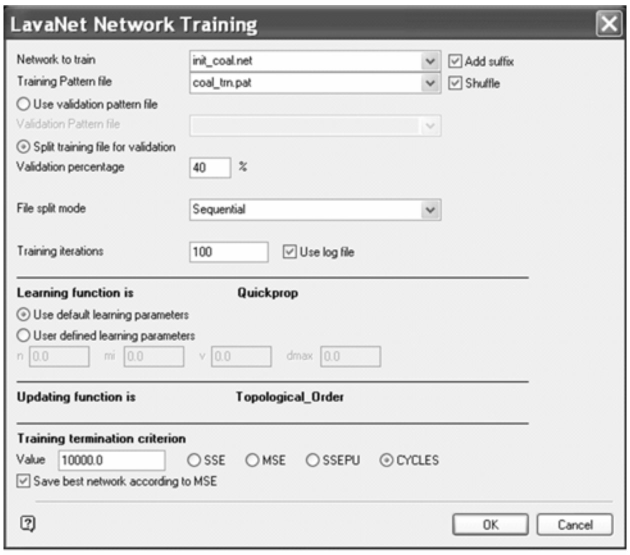
$$NormData(i) = Amplitude(i) * Data(i) + Offset(i)$$

$$DenormData(i) = (Data(i) - Offset(i)) / Amplitude(i)$$

LavaNet also provides the ability to generate a normalization file using user defined upper and lower bounds. This is important when training, validation and application patterns have different input and output space limits.

3.6 Network Development and Application

Network development is a multiple stage process in LavaNet. After the generation of the undeveloped network file, the network is initialised using one of the available initialisation functions in LavaNet. Initialisation is important as it provides the starting point for further development of the network, by giving appropriate values to the free parameters of the network (initial weights, biases, function centers, etc.) After initialisation, the network can be trained using an appropriate learning function depending on the type of network. LavaNet provides the ability to split the training pattern file to two subsets that will be used for training and validation (Fig. 4). This can be done with training patterns first and validation last or evenly. The per-



LavaNet Network Training

Network to train: init_coal.net ☑ Add suffix
Training Pattern file: coal_trn.pat ☑ Shuffle
○ Use validation pattern file
Validation Pattern file:
[✓ Split training file for validation]
Validation percentage: 40 %
File split mode: Sequential
Training iterations: 100 ☑ Use log file

Learning function is **Quickprop**
[✓ Use default learning parameters]
○ User defined learning parameters
n 0.0 mi 0.0 v 0.0 dmax 0.0

Updating function is **Topological_Order**

Training termination criterion
Value 10000.0 ○ SSE ○ MSE ○ SSEPU [✓ CYCLES]
☑ Save best network according to MSE

Figure 4: Network training and validation panel with choice of pattern files, learning parameters and training termination criterion.

centage used for each subset is also defined by the user.

Termination of training can be based on one of the four criteria provided by SNNS (sum or mean of squared error, sum of squared error per unit or number of training cycles). LavaNet can store the network state that produces the lowest mean squared error (MSE) when termination is based on cycles, e.g. the network can be trained for 10,000 cycles but MSE was lowest at 7,500 leading to LavaNet saving the network at that point and not at the end of training.

As all network development actions in LavaNet are stored in Batchman compatible script files, the user can combine them using a text editor to further automate the development process once it is finalised.

4. CASE STUDIES

4.1 Modeling of Lignite Thickness

The first example is a simple 2D application of a MLP network for the construction of a mineable thickness model of a lignite deposit. Mineable thickness was evaluated in each drillhole available and a pattern file is generated with two inputs, the X and Y coordinates of the drillhole, and one output, the mineable lignite thickness. Part of the SNNS pattern file built by LavaNet using the drillhole database thickness information is shown below. Each pattern consists of two lines, the first giving the inputs and the second giving the output (normalised).

```
SNNS pattern definition file V1.4
generated at Thu Jun 22 15:01:23 2006

No. of patterns     : 1656
No. of input units  : 2
No. of output units : 1

# Pattern 631
0.536582871292458 0.35064173900141
0.420175631174534
```

Figure 5: Example of pattern file generated by LavaNet with normalisation of inputs and output.

```
loadNet("init_cotk.net")
loadPattern("lavanet_trn.pat")
loadPattern("lavanet_vln.pat")
setPattern("lavanet_trn.pat")
min_error = 1
setShuffle (TRUE)
setLearn-
Func("BackpropMomentum",0.01,0.2,0,0.01)
setUpdateFunc ("Topological_Order" )
repeat
   for i:=1 to 500 do
      trainNet ()
   endfor
   setPattern ("lavanet_vln.pat")
   testNet()
   if min_error > MSE then
      min_error = MSE
      saveNet("train_init_cotk.net")
      print ("Best MSE = ", MSE)
   endif
   criterion := CYCLES
   print ("Cycles=",CYCLES,"CYCLES=",CYCLES)
   setPattern ("lavanet_trn.pat")
until criterion >= 20000
```

Figure 6: Batchman training script generated by LavaNet.

The pattern file was split 60%-40% for training and validation. The network was first initialised and then trained using the script that was generated by LavaNet (Fig. 6). Training lasted 20,000 cycles while the network's performance was validated every 500. The best network state according to MSE was saved to a separate network file. This network was used to generate a grid model of mineable lignite thickness using the appropriate LavaNet options. A separate ap-

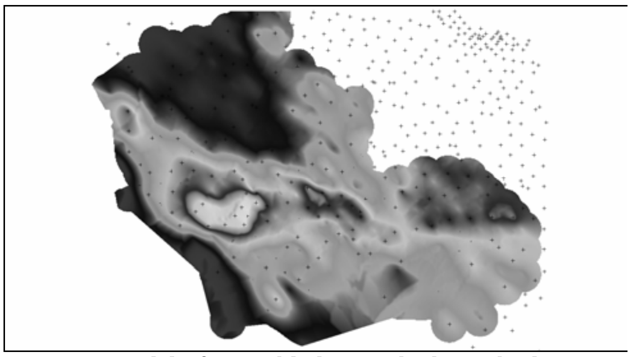
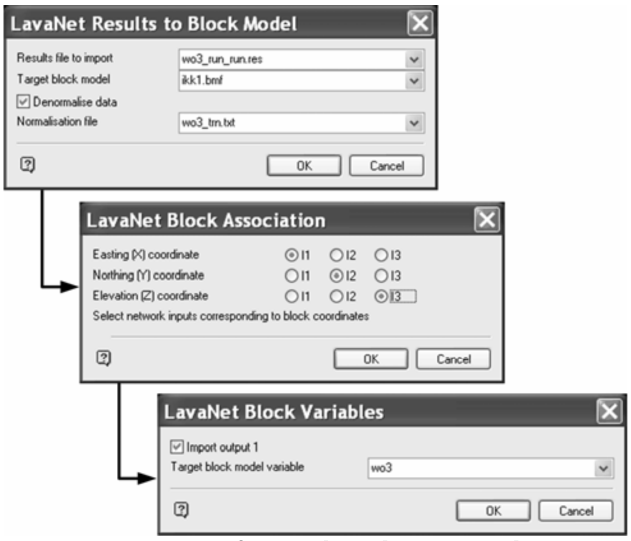


Figure 7: Model of mineable lignite thickness built using LavaNet.


LavaNet Results to Block Model
Results file to import: wo3_run_run.res
Target block model: ikk1.bmf
[✓] Denormalise data
Normalisation file: wo3_trn.txt

LavaNet Block Association
Easting (X) coordinate: [✓ I1] I2 I3
Northing (Y) coordinate: I1 [✓ I2] I3
Elevation (Z) coordinate: I1 I2 [✓ I3]
Select network inputs corresponding to block coordinates

LavaNet Block Variables
[✓] Import output 1
Target block model variable: wo3

Figure 8: Importing of network application results into block model in Vulcan using LavaNet Results options.

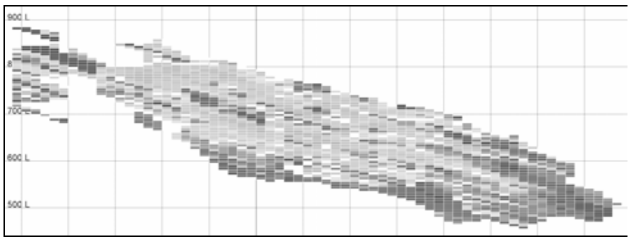


Figure 9: Block model section showing WO_3 estimates from LavaNet RBF network.

plication pattern file was generated for this, using an existing grid model to provide the inputs for network application. Figure 7 shows the produced model of lignite thickness.

4.2 Modelling of WO_3 Grade in 3D

In this second example, an RBF network is used to estimate WO_3 grade of a tungsten deposit in 3D (X,Y,Z coordinates as inputs). Drillhole composite data are used to generate the pattern files to train and validate the network, while a block model is used to generate the application pattern file. The development of the network is similar to the previous example - the initialization, learning and updating functions differ as a different type of network is now used. The trained network is applied on the block model pattern file and the results are de-normalized and imported back into the block model (Fig. 8). Figure 9 shows a section through the block model colored by WO3 estimates of the LavaNet RBF network.

5. CONCLUSIONS

LavaNet is an interface to a neural network development environment, built inside a general mine planning package, which allows quick and effortless development and application of artificial neural networks for mining and environmental problems. LavaNet is constantly under development with further options and functionality being added that take advantage of new capabilities provided by new versions of Vulcan's LAVA scripting language. LavaNet is available for downloading from:
airlab.teikoz.gr/geope/downloads/kapageridis/lavanet.htm

REFERENCES

Kapageridis, I., 2005. Input Space Configuration Effects in Neural Network Based Grade Estimation. Computers & Geosciences, Vol. 31, Issue 6, International Association for Mathematical Geology, Elsevier, pp. 704-717.

Kapageridis, I., 1999. Application of Artificial Neural Network Systems in Grade Estimation from Exploration Data, PhD Thesis, Department of Mineral Resources Engineering, University of Nottingham, Nottingham.

Kapageridis, I., 2002. Artificial Neural Network Technology in Mining and Environmental Applications. In: 11th International Symposium on Mine Planning and Equipment Selection (MPES 2002), VŠB - Technical University of Ostrava, Prague.