Generating a database for DEERNet training

Hands-on session instructions, 29/10/2024

If you plan to submit assessed work, **questions highlighted in blue** must be answered in your report; there is no target word count – be reasonable and succinct. If you are not required to submit a report, simply explore those questions until you find satisfactory answers.

1. Clone the DEERNet repository:

```
https://github.com/IlyaKuprov/DEERNet
```

In *Matlab*; right-click the `DEERNet/kernel` folder and click *Add to Path / Selected folders*. Navigate into the *DEERNet* example folder and run a few example scripts.

In this tutorial, we will trust the program to have the quantum mechanics implemented already; we simply tell it what to do. If you would like further information on the technicalities, look into the source code of the functions we are going to call, and read the three paper PDF files provided.

2. To obtain default *DEERNet* training database parameters, type:

```
parameters=library_dd()
```

Matlab will report a number of parameters to the console. If you are specialising in EPR spectroscopy and you are interested in their physical meaning, type

```
edit library_dd
```

and take a look at the code comments. Further explanations are provided in the *Materials and Methods* section of *deernet_paper_a.pdf* file.

(A) How many distinct simulated question-answer pairs do you expect to be necessary for unambiguous training of a fully connected neural network with 10^6 parameters? [10 marks]

The network we are going to train is much smaller; we will only need 50,000 question-answer pairs with questions (time domain) and answers (distance distribution domain) 256 points long. We will also restrict the network to DEER experiment and not train it to process data from a related RIDME experiment. We therefore set the following additional parameters (copy and paste these lines into the *Matlab* console window and press `Enter` to run them):

```
parameters.ntraces=50000;
parameters.expt='deer';
parameters.np_time=256;
parameters.np_dist=256;
```

The following command will run for a while and generate the training database:

```
library=deer_lib_gen([],parameters)
```

If *Matlab* complains about not being able to write a cache file, point the file browser (on the left of *Matlab* window) to your home directory and run this command again.

Inspect the resulting library object. It contains a vector of time axis ticks for the time-domain data, a vector of distance axis ticks for the distance distribution data, and several arrays with 50,000 instances of DEER data in columns. The time grid is in seconds and the distance grid is in Angstrom.

(B) Plot a few of them (use *Matlab*'s `plot` command, read the manual to find out how to use it) to get an idea of the time and distance scale of DEER experiment. [5 marks]

An example of the plotting commands (for the first data pair) for DEER trace in the time domain and the corresponding distance distribution is:

```
figure(); plot(library.time_grid,library.deer_noisy_lib(:,1))
figure(); plot(library.dist_grid,library.dist_distr_lib(:,1))
```

(C) Type `whos` and take a look at the amount of memory used by the library array. How much memory would a library with 10^{10} DEER traces consume? How did the authors of DEERNet work around this storage problem (see the JMR paper)? [10 marks]

3. To create an untrained fully connected two-layer neural network, copy and paste the following command into *Matlab*'s command line:

```
layers=[featureInputLayer(256);
        fullyConnectedLayer(128);
        batchNormalizationLayer();
        softplusLayer('Name','SP1');
        fullyConnectedLayer(256);
        batchNormalizationLayer();
        softplusLayer('Name','SP2');
        renormLayer();
        regressionLayer()]
```

Ask ChatGPT 4o to explain this syntax to you line by line; its explanations are surprisingly good.

(D) Google up and describe briefly the mathematical nature of a fully connected layer and a batch normalisation layer. Renormalisation layer makes sure that the signal integrates to 1. Why is this necessary for DEER data? [15 marks]

4. To train the network, copy and paste the following commands into *Matlab*'s command line:

```
train_opts=trainingOptions('adam','MaxEpochs',100,'Verbose',true,...
                           'MiniBatchSize',256,'ExecutionEnvironment',...
                           'cpu','Plots','none')

[net,train_stats]=trainNetwork(library.deer_noisy_lib',...
                               library.dist_distr_lib',...
                               layers,train_opts);
```

Watch the training process (careful, watching optimisations is addictive). Repeat the above steps with:

(a) a larger network (insert one or more additional [fully connected]-[batch norm]-[softplus] triads after the first one);

(b) a larger training database;

(c) a larger minibatch size (ask ChatGPT 4o to explain you what that is);

(d) the SGDM training algorithm (see the manual for `trainingOptions`).

You may find it convenient to place all of the above commands into a script and run the script. Observe the changes in the resulting final root-mean-square error (the lower, the better).

(E) Rationalise your observations. Google up and describe the difference between conventional and stochastic gradient descent. What is the difference between SGDM and ADAM methods? What is batch size and how does it influence the training? [20 marks]

5. To use the resulting neural network for data processing, use `predict` command. For example, to process the first dataset in the library:

```
ans=predict(net,library.deer_noisy_lib(:,1)')
```

(F) Plot a few network predictions (`ans` variable from above, use "`hold on`" command to overlay the next plot on the previous one) and compare them to the known right answers from the library (contained in the corresponding columns of the `library.dist_distr_lib` array). [10 marks]

(G) Compare the performance of your network with one of the highly trained networks supplied with *Spinach*. Use one of the networks from the following location:

`/usr/software/spinach/experiments/deernet/netset/net_distan_dd/deer-(256)-256-512`

Its output is 512 points long; use `linspace` command to make a distance axis manually. [20 marks]

(H) From the run time of your training process and the memory utilisation of the training library array, estimate the computing resources required to train one production-grade DEERNet network (six FC-BN-SP triads, 10^{10} question-answer pairs in the database). [10 marks]

6. Run a few examples from `~/DEERNet/examples` directory and inspect the output. Read the three papers from the workshop folder.