1 Results

In the present section, the same datasets from the previous chapter were used. First it was necessary to prepare the data to be fed to the DNN.

1.1 Data Preparation

Each dataset was first filtered using the same filter as phy: a forwards-backwards Butterworth filter of order 3 with cutoff frequency set to 500Hz.

Each example was defined to be the array of time windows of 100 samples for each of the 127 channels in the probe. Therefore, the input data has 12700 dimensions. However, due to constraints on the available hardware, not all windows were used. Each window is shifted by 5 samples from the previous one, i.e., the first example are the 127 time windows with $t \in [0, 99]$ and the second example are the 127 time windows when $t \in [5, 104]$. Furthermore, only samples in $t \in [1000000, 2000000]$ were utilized. This yields, at this stage, 199980 examples per dataset

The windows whose central sample was closer to the Juxta Times were labeled as "1" (positive examples). Otherwise they were labeled as "0" (negative examples). The label "1" should be interpreted as "contains a spike from the juxta cell" and "0" as "doesn't contain a juxta spike". It is possible that a spike from the juxta cell be partially contained in the window of another spike from the juxta cell. However this situation is highly unlikely since for that to happen it would mean that the two spike were separated by less that 50 samples, corresponding to 1.67 ms which is less than the refractory period of cells under consideration.

This input data was split in two set: the training set (TS), with which the DNN will train, and the validation set (VS), where the resulting trained DNN is tested. The TS held 70% of the input data (139986 examples) and VS held the remaining 30% (59994 examples)

In table 1 is result of this splitting.

	Input Data		Training Set		Validation Set	
cell ID	No. of 1	Fraction	No. of 1	Fraction	No. of 1	Fraction
8213	292	0.15%	202	0.14%	90	0.15%
936	127	0.06%	83	0.06%	44	0.07%
939	298	0.15%	207	0.15%	91	0.15%
945	14	0.01%	9	0.01%	5	0.01%
997	38	0.02%	29	0.02%	9	0.02%

Table 1: In this table are presented, for each recording, the number of examples labeled as "1" and its fraction in the Input Data, and separated in the Training Set and Validation Set. The total number of examples in the Input Data, Training Set and Validation Set are 199980, 139986 and 59994, respectively

As can be seen in in Table 1, all recordings reveal a very large unbalance:

there are always many more examples belonging to the class "0". In these situations, the most likely scenario is the convergence of the DNN to a trivial solution where it outputs "0" regardless of the input, yielding an accuracy equal to the fraction of examples labeled as "0".

To address this problem, it was necessary to perform upsampling: the positive examples were repeated by the same factor in both the TS and the VS. The upsampling factor was determined so that the positive examples represent around 30% of the total number of examples in both the TS and the VS.

The results of the operation are presented in table

	Training Set			Validation Set		
cell ID	No. of examples	No. of 1	Fraction	No. of examples	No. of 1	Fraction
8213	195334	55550	28.4%	84654	24750	29.2%
936	192276	52373	27.2%	87714	27764	31.7%
939	195669	55890	28.6%	84473	24570	29.1%
945	191412	51435	26.9%	88564	28575	32.3%
997	201060	61103	30.4%	78948	18963	24.0%

Table 2: In this table are presented the details of the Training Set and Validation Set after upsampling was performed.

This procedure artificially increases the size of datasets. However the resulting sizes are relatively close to each other and therefore DNNs trained with all the TS can be compared fairly.

1.2 Parameter Optimization

In this project, we planned to study how a feed-forward deep Neural Network performs as a spike detection for neural data. In the context of machine learning this is a binary classification problem.

It was necessary to define the basic architecture of the DNN. First and foremost, the dimension of the input layer was set to 12700 and the output layer should have only one neuron. Usually the number of adjustable parameters in a DNN should be of the same order of magnitude of the number of examples to be used and therefore the following layer had to be much smaller. In fact, to follow this rule, the second layer should have around 15 neurons. This would be 800-fold compression which seemed excessive. Therefore, it was necessary to compromise. The second layer was define with 200 neurons. Comparatively the size of the following layer had very little influence and for that reason it was decided not to compress any further and set the dimension of layer 3 and 4 to have 200 neurons as well.

Following the results of REFERENCE Xavier Glorot, Antoine Bordes and Yoshua Bengio (2011) Rectified Linear Units (RELU) were should as the activation function, which should result in faster learning and weaker dependence on the initial conditions. For the last layer a sigmoid activation function was used, since the problem under consideration is of binary classification.

As for the loss function, cross entropy was chosen given its characteristic fast converge when the a activation function of the output neuron is a sigmoid.

Regarding the regularization method, the L1-norm was utilized as well as dropout with 0.2 probability parameter.

This defines the basic model of DNN that was utilized: it was a fully-connected feed-forward DNN with:

- $n_l = 5$
- $S_1 = 12700, S_2 = S_3 = S_4 = 200, S_5 = 1$
- ReLU as activation function for hidden units and Sigmoid for the output neuron
- Cross-entropy for loss function
- L1-Regularizer along with dropout