# EE 565 Machine Learning Project 4 Report

Mehrdad Ghyabi

*Abstract*—**This is a summarized report of the project for of the machine learning course. The main focus of this project is on the performance of the multi-layer perceptron, autoencoders and principal component analysis. The training part of the multilayer perceptron is done by a predefined function.**

*Index Terms*— **Multi-layer perceptron (MLP), Mean square error (MSE), Decision boundaries, Training error, Validation error, autoencoders, principal component, dimension reduction**

## I. Introduction

This report is consisted of seven sections. The first two sections after the introduction focus on the training of MLPs using training data with different distributions and the effect of the network architecture on the training performance. The next section investigates the effect of training on the train MSE and validation MSE. After that autoencoders and their abilities to compress data as well as denoising it are studied. Finally, data dimension reduction using PCA is investigated and a sample dataset is clustered after performing PCA.

## II. Problem 1: The Multi-Layer Perceptron (MLP) and Double Moon Dataset

In this part, a small study was conducted on the multi-layer perceptron using a dataset consisting of 300 datapoints drawn from a double moon distribution with parameters $r = 1$ and $w = 0.6$ and $d = 1$.

### A. One hidden layer

In this part, one hidden layer consisting 10 neurons was trained using the provided training function. To be sure about the results, the training was performed 20 times, each time with a new training set and the mean square error (MSE) over different iterations was averaged. The resulting plot of MSE versus number of epochs is illustrated in the Figure 1. After 2000 epochs, it seems the MSE has become reasonably small.

### B. One hidden layer performance

The performance of the trained network was evaluated using an unseen dataset drawn from a double moon distribution with the same parameters. Figure 2 shows the resulting classification as well as contours showing decision boundaries of the trained MLP. In this figure, correctly classified members of the class 1 are plotted as “blue +”, correctly classified members of class 2 as “green x” and incorrectly classified points are plotted as “red *”. The model is obviously over trained in this case and there are no misclassified datapoints.

### C. Two hidden layers

In this part, two hidden layers, each consisting 5 neurons were trained using the provided training function. To be sure about the results, the training was performed 20 times, each time with a new training set and the mean square error (MSE) was averaged. The resulting plot of MSE versus number of epochs is illustrated in the Figure 3. After 2000 epochs, it seems the MSE has become reasonably small with many fluctuations in this case.

### D. Two hidden layers performance

The performance of the trained network was evaluated using an unseen dataset drawn from a double moon distribution with the same parameters. Figure 4 shows the resulting classification as well as contours showing decision boundaries of the trained MLP. In this figure, correctly classified members of the class 1 are plotted as “blue +”, correctly classified members of class 2 as “green x” and incorrectly classified points are plotted as “red *”. It seems the model is well trained to the shape of double moon. There is only one misclassified datapoint.

### E. Three hidden layers

In this part, three hidden layers, with 3,4, and 3 neurons in them were trained using the provided training function. To be sure about the results, the training was performed 20 times, each time with a new training set and the mean square error (MSE) was averaged. The resulting plot of MSE versus number of epochs is illustrated in the Figure 5. After 2000 epochs, it seems the MSE has become reasonably small with many more fluctuations.

### F. Three hidden layers performance

The performance of the trained network was evaluated using an unseen dataset drawn from a double moon distribution with the same parameters. Figure 6 shows the resulting classification as well as contours showing decision boundaries of the trained MLP. In this figure, correctly classified members of the class 1 are plotted as "blue +", correctly classified members of class 2 as "green x" and incorrectly classified points are plotted as "red *". Out of the three cases, this model is the most well behaving to the nonlinearities of the double moon distribution.

## III. Problem 2: The multi-Layer Perceptron (MLP) and Gaussian XOR Dataset

In this part, a small study was conducted on the multi-layer perceptron using a dataset consisting of 300 datapoints drawn from a Gaussian XOR dataset with parameters $\sigma = 1$.

### *A. One hidden layer*

In this part, one hidden layer consisting 15 neurons was trained using the provided training function. To be sure about the results, the training was performed 20 times, each time with a new training set and the mean square error (MSE) over different iterations was averaged. The resulting plot of MSE versus number of epochs is illustrated in the Figure 7. After 2000 epochs, it seems the MSE has become reasonably small.

### *B. One hidden layer performance*

The performance of the trained network was evaluated using an unseen dataset drawn from a Gaussian XOR distribution with the same parameters. Figure 8 shows the resulting classification as well as contours showing decision boundaries of the trained MLP. In this figure, correctly classified members of the class 1 are plotted as "blue +", correctly classified members of class 2 as "green x" and incorrectly classified points are plotted as "red *". There are some misclassified datapoints.

### *C. Two hidden layers*

In this part, two hidden layers, consisting 8 and 7 neurons each, were trained using the provided training function. To be sure about the results, the training was performed 20 times, each time with a new training set and the mean square error (MSE) was averaged. The resulting plot of MSE versus number of epochs is illustrated in the Figure 9. After 2000 epochs, it seems the MSE has become reasonably small with many fluctuations in this case.

### *D. Two hidden layers performance*

The performance of the trained network was evaluated using an unseen dataset drawn from a Gaussian XOR distribution with the same parameters. Figure 10 shows the resulting classification as well as contours showing decision boundaries of the trained MLP. In this figure, correctly classified members of the class 1 are plotted as "blue +", correctly classified members of class 2 as "green x" and incorrectly classified points are plotted as "red *". It seems the model is well trained to the shape of Gaussian XOR. There is no misclassified datapoints.

### *E. Three hidden layers*

In this part, three hidden layers, consisting of 4,7, and 4 neurons in each, were trained using the provided training function. To be sure about the results, the training was performed 20 times, each time with a new training set and the mean square error (MSE) was averaged. The resulting plot of MSE versus number of epochs is illustrated in the Figure 11. After 2000 epochs, it seems the MSE has become reasonably small with many more fluctuations.

### *F. Three hidden layers performance*

The performance of the trained network was evaluated using an unseen dataset drawn from a Gaussian XOR distribution with the same parameters. Figure 12 shows the resulting classification as well as contours showing decision boundaries of the trained MLP. In this figure, correctly classified members of the class 1 are plotted as "blue +", correctly classified members of class 2 as "green x" and incorrectly classified points are plotted as "red *". Out of the three cases, the second model is the most well behaving to the Gaussian XOR distribution.

## IV. Problem 3: Model Validation

In this section, model validation for the models trained in the last two problems are investigated. In both cases models are trained using only 300 datapoints. The validation sets, on the other hand, consist of 3000 datapoints. This way, a synthetic noise is added to the decision boundaries of the models. In each part a different network architecture is used.

### *A. Model validation for double moon distribution*

A training dataset with 300 datapoints was drawn from a double moon distribution with parameters $r = 1$ and $w = 0.6$ and $d = 1$. A validation dataset with 3000 data points was drawn from the same distribution. A network with one hidden layer containing 10 neurons was trained with 7000 epochs. MSE for both training set and validation set was calculated at each epoch and is plotted in the Figure 13. According to MSE plots, a stopping point at epoch number 5000 was chosen and performance of the network at that epoch is shown for both training set and validation set in Figure 14 and Figure 15. Accuracy of classification for the training set is 0.997 and for validation set is 0.995.

### *B. Model validation for double moon distribution*

A training dataset with 300 datapoints was drawn from a Gaussian XOR distribution with parameter $\sigma = 1$. A validation dataset with 3000 data points was drawn from the same distribution. A network with two hidden layers, each containing 8 neurons was trained with 5000 epochs. MSE for both training set and validation set was calculated at each epoch and is plotted in the Figure 16. According to MSE plots, a stopping point at epoch number 1000 was chosen and performance of the network at that epoch is shown for both training set and validation set in Figure 17 and Figure 18.

Accuracy of classification for the training set is 0.993 and for validation set is 0.921.

## V. Problem 4: Autoencoder-based Image Compression/Filtering

In this part of the project an autoencoder neural network is trained using the first 50 images in the CIFAR-10 dataset. Each image has a size of 32 by 32 pixels in three layers (RGB). The main usage of autoencoders is in compressing and denoising data.

### *A. Toolbox description*

To solve this problem the "Keras" library was chosen to develop codes on Python platform. Keras is a neural network library capable of training deep neural networks as well as convolutional neural networks. There are numerous built-in options in model builder, compiler and fitting algorithm. In this report the options that are used to solve the problem are explained.

The input of the training module is a set of 50 vectors each consisting 32×32×3=3072 pixels. The outputs of the algorithm then have to be reshaped in the original 32 by 32 by 3 images to be visualized.

As instructed in the project definition, one hidden layer (50 neurons) and one output layer (3072 neurons) have been defined, resulting in 310,322 trainable parameters. Activation function for the hidden layer is set to be "relu" and that of the output layer is set to be "sigmoid". The optimizer "adadelta" and loss function "binary_crossentropy" was selected after examining different options.

### *B. Implementation*

The autoencoder was defined and trained using 50,000 epochs. Figure 19 shows the 50th image in the dataset at the top, as well as the visualization of the encoded image in the middle and the decoded image at the bottom. Figure 20 shows the same information for four other randomly selected images.

### *C. Mean Square Error (MSE)*

The MSE of the whole 50 image dataset was computed using the equation 4.1, in which $\boldsymbol{y}$ is the original image reshaped into a vector and $\hat{\boldsymbol{y}}$ is the encoded version of it. The MSE calculated over the training dataset was 0.100. The pixel values in this problem are between 0 and 1.

$$MSE = \sum_{images} \frac{(\boldsymbol{y} - \hat{\boldsymbol{y}})^2}{3072} \tag{4.1}$$

### *D. Denoising*

The trained autoencoder was used to denoise a few noisy images. The train set was corrupted with a Gaussian noise with $\mu = 0$ and $\sigma = 0.098$ (adjusted for values between 0 and 1). Then the noisy data was fed to the autoencoder and decoded denoised images were stored. Figure 21 shows the noisy and denoised versions of the 50th image in the dataset. Figure 22 presents the same illustration for four other randomly selected images.

## VI. Problem 5: Principle Component Analysis

This part of the report is dedicated to an example of principle component analysis (PCA). The neural spike trace dataset is analyzed in this section. This dataset contains 200 neural spike traces and is stored in a .csv file. The goal is to use PCA to cluster the 200 traces and correspond each cluster to a single neuron.

### *A. Toolbox description*

To perform PCA, the "sklearn" library is chosen to be used in Python. It is a linear dimensionality reduction algorithm which uses singular value decomposition (SVD) to project the data to a lower dimensional space. The input data is centered but not scaled.

The desired number of principal components can be easily defined in the code. According to the instructions, this number should be less than the minimum of number of samples and the number of features.

There is an option of copy which, if set "True" will copy the result on the input data. Another option "whiten", if set "True", multiplies the components vectors to the square root of number of samples and then divide the result by the singular values to ensure uncorrelated outputs with unit component-wise variances.

There are various options to customize the SVD solver. Tolerance of the singular values can also be set using the "tol" option. Two important attributes of this toolbox are variance and singular values. Variance can be reported in the form of variance ratio as well. Moreover, variance of the noise can be reported too.

### *B. First principal component*

The PCA analysis was done using the selected toolbox to get the first principal component, and the result is presented in the Figure 23. 37.87 percent of the variance of the dataset is captured by its first principal component. It is hard to cluster the data projected to one axis using eye inspection. Four to six clusters are visible in the Figure 23, depending on how the clusters are defined.

### *C. First two principal components*

The PCA analysis was done using the selected toolbox to capture the first two principal components of the dataset, and the result is presented in the Figure 24. 57.03 percent of the variance of the dataset is captured by its first two principal components. Using eye inspection, it seems that there are three clusters in the dataset.

### *D. First three principal components*

The PCA analysis was done using the selected toolbox to capture the first three principal components of the dataset, and the result is presented in a 3D plot in the Figure 25. 69.52

percent of the variance of the dataset is captured by its first three principal components. Using eye inspection, it seems that there are three clusters in the dataset.

### *E. Clustering the data*

The same code developed for the project1 to perform the batch-Kmeans algorithm was used in this part to identify the clusters in the 2D space of the first two principal components of the dataset. Since eye inspection suggested existence of three clusters, three clusters were identified by the batch-Kmeans. The resulting 2D plot is presented in the Figure 26.

### *F. Showing different clusters*

Each cluster of spike traces are shown together in Figure 27 to Figure 29. Clustering the dataset into three groups seems to be reasonable. The first cluster of traces show a global maximum followed by a global minimum. The global maximum is lower than the other two clusters. The second Cluster shows only a global maximum in the middle of the trace, and some datapoints which seem to belong to the third group. The third cluster shows a global minimum followed by a global maximum.

## VII. Conclusions

Multi-layer perceptron is a powerful classifier, but special care should be taken when working with it. MLP is susceptible to overfitting. A good practice is to check MSE of the validation set along with training set to select a reasonable stopping point. Results of this study showed that this could be a problem in the regions furthest from the center of data clusters.

Autoencoders proved to be reliable tools for compressing and denoising data. This study showed that a good performance is achievable using only one hidden layer.

Principal component analysis was proved to be a powerful pre-analyzer prior to clustering datasets with high dimensionality.

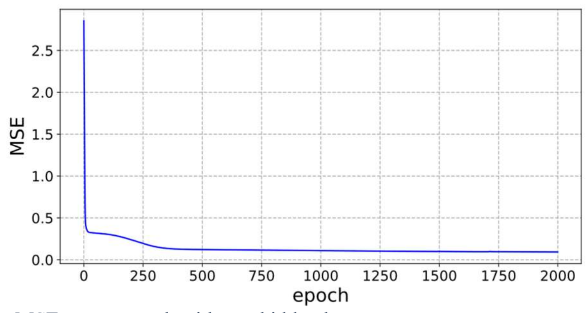

Fig. 1. MSE versus epoch with one hidden layer

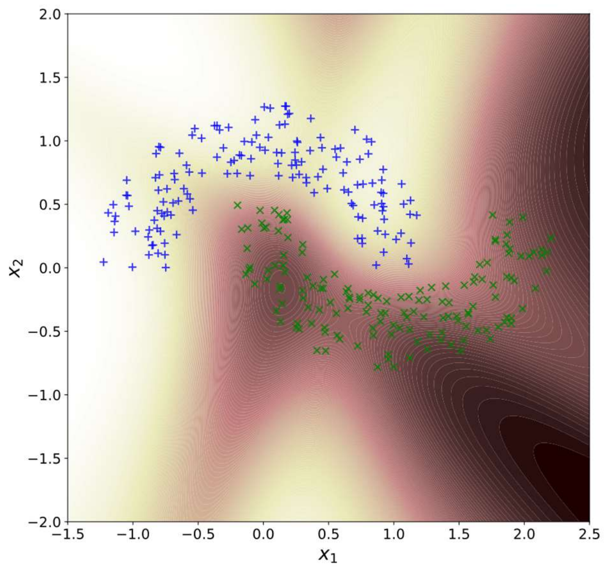

Fig. 2. Decision boundary with one hidden layer

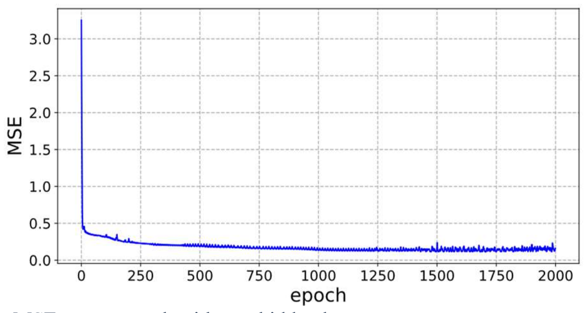

Fig. 3. MSE versus epoch with two hidden layers

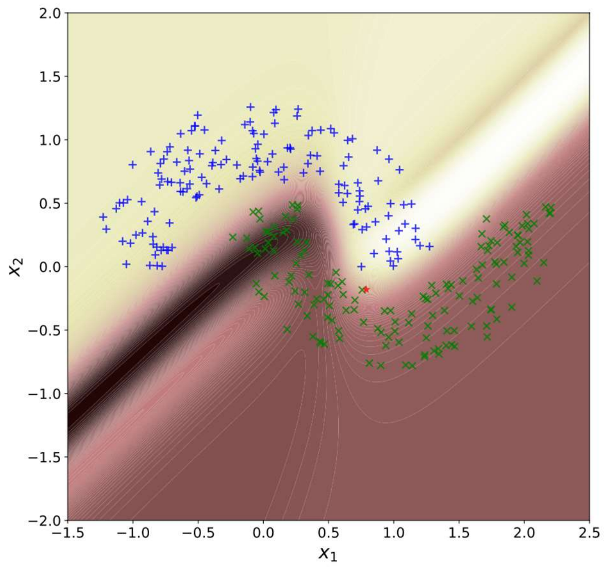

Fig. 4. Decision boundary with two hidden layers

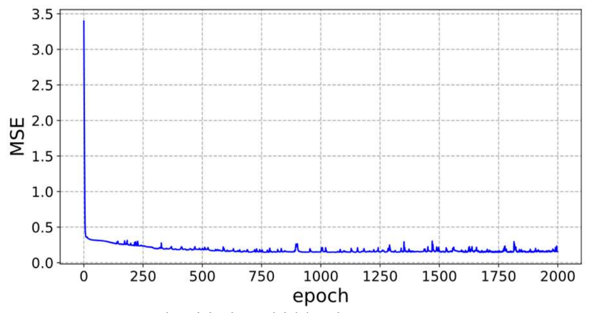

Fig. 5.  MSE versus epoch with three hidden layers

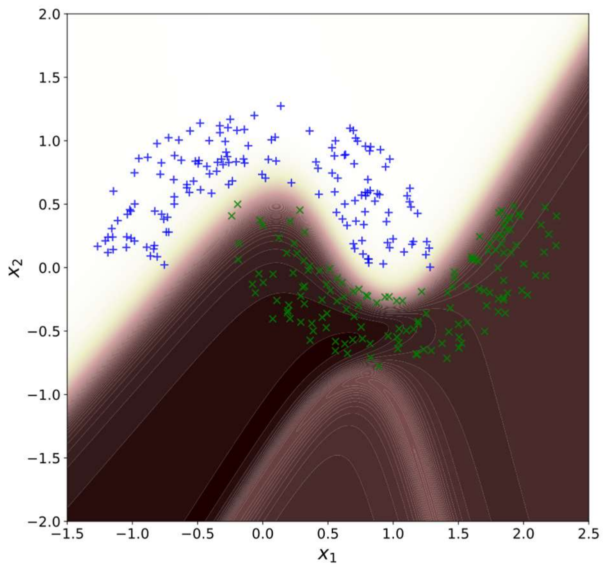

Fig. 6.  Decision boundary with three hidden layers

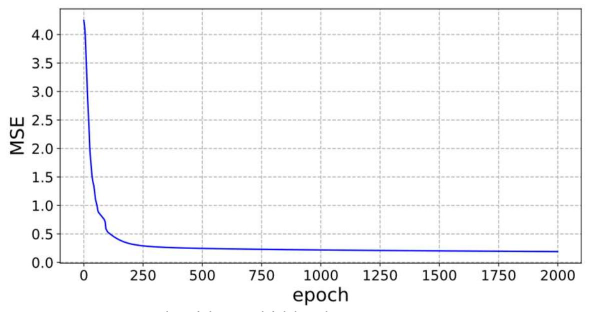

Fig. 7.  MSE versus epoch with one hidden layer

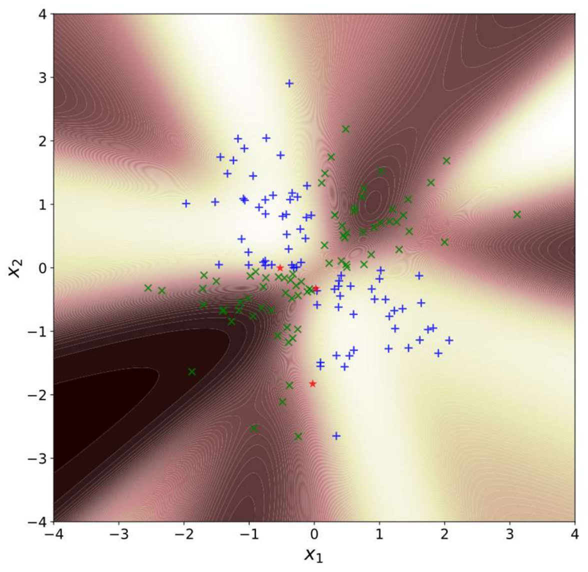

Fig. 8.  Decision boundary with one hidden layer

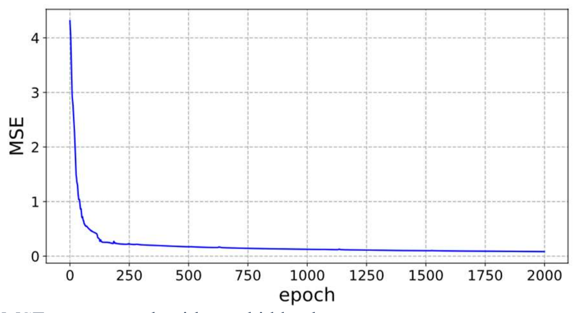

Fig. 9.  MSE versus epoch with two hidden layers

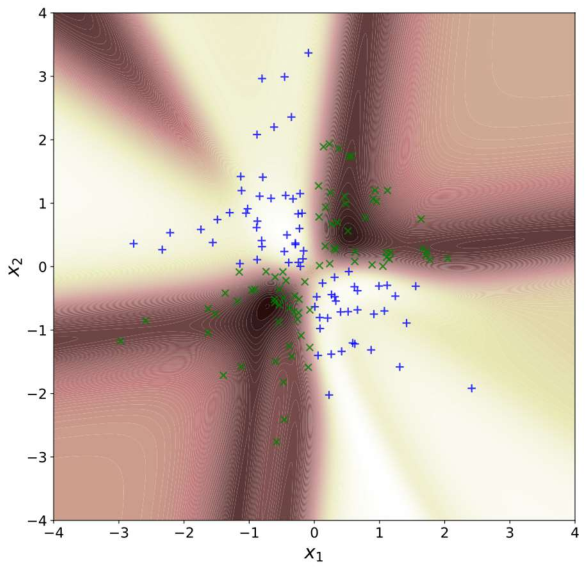

Fig. 10.  Decision boundary with two hidden layers

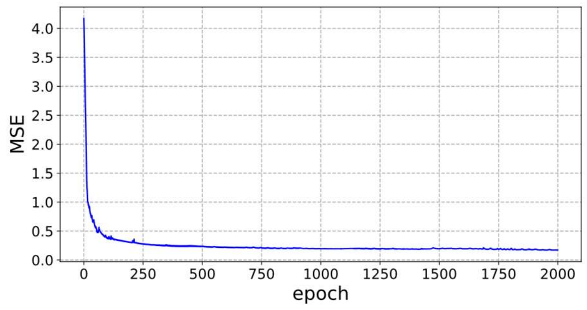

Fig. 11. MSE versus epoch with three hidden layers

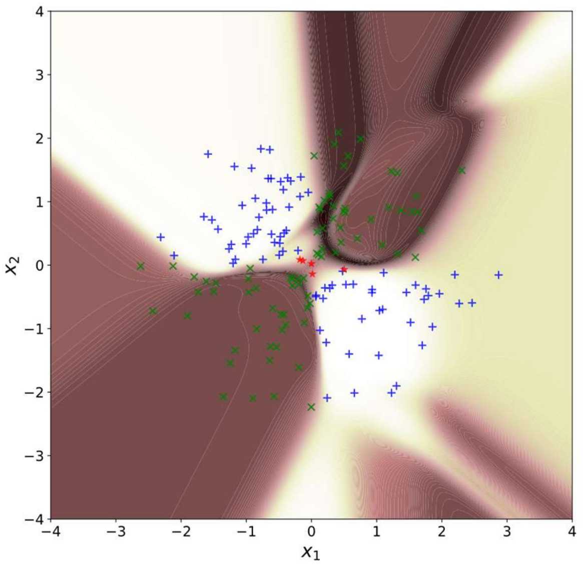

Fig. 12. Decision boundary with three hidden layers

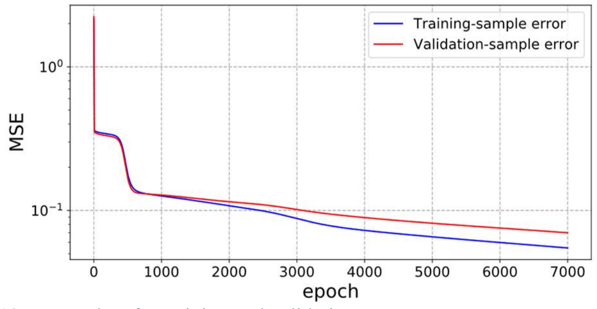

Fig. 13. MSE plots for training and validation sets

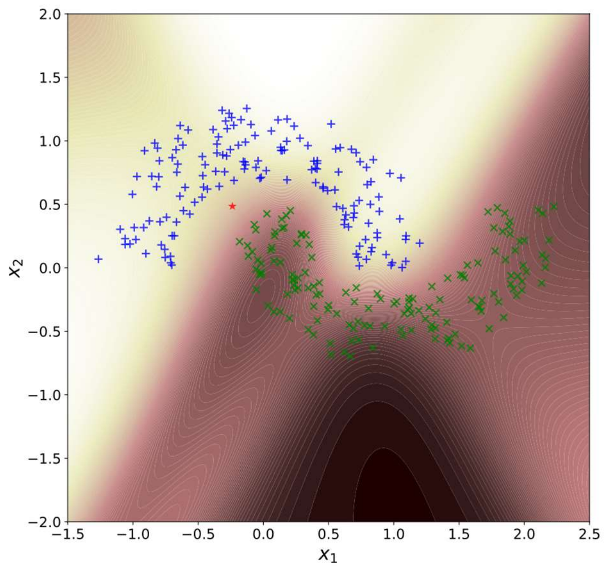

Fig. 14. Training set accuracy

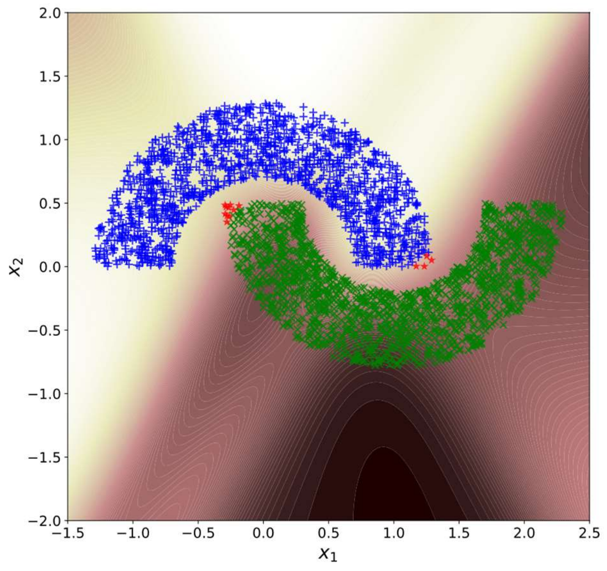

Fig. 15. Validation set accuracy

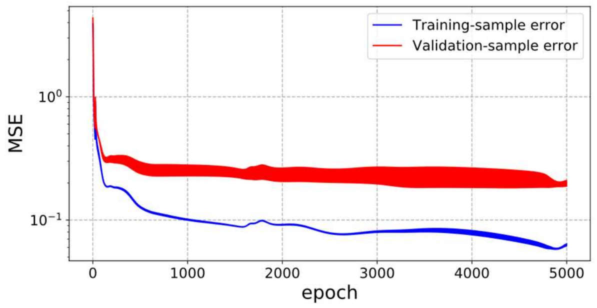

Fig. 16. MSE plots for training and validation sets

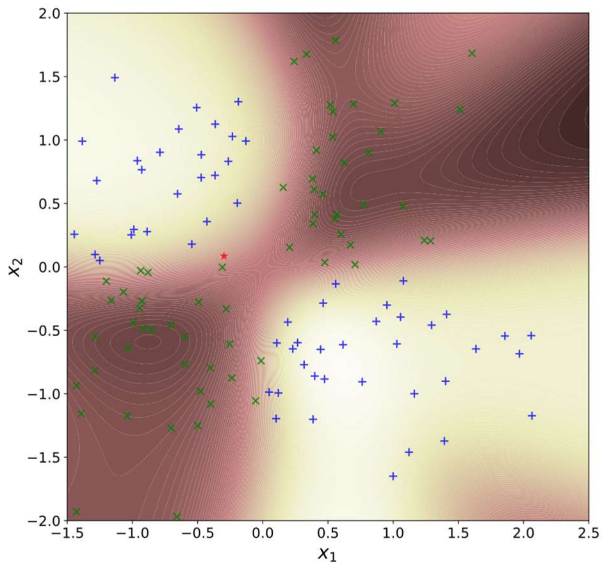

Fig. 17. Training set accuracy

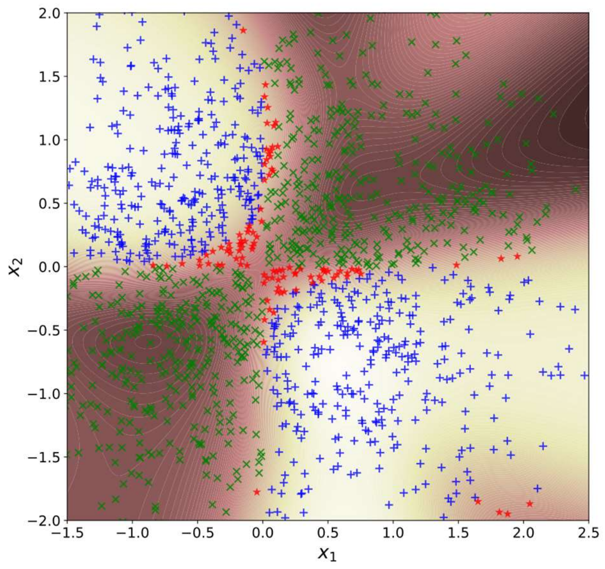

Fig. 18. Validation set accuracy

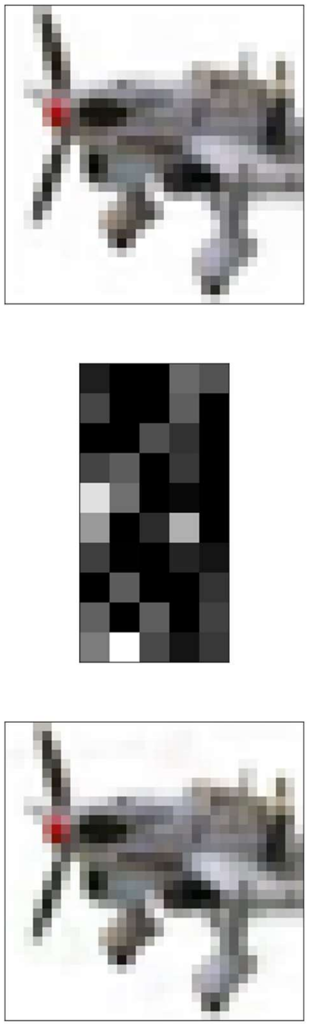

Fig. 19. Original Image (top), Encoded image (middle), Decoded Image (bottom)

Fig. 20. Original Images (top), Encoded images (middle), Decoded Images (bottom)

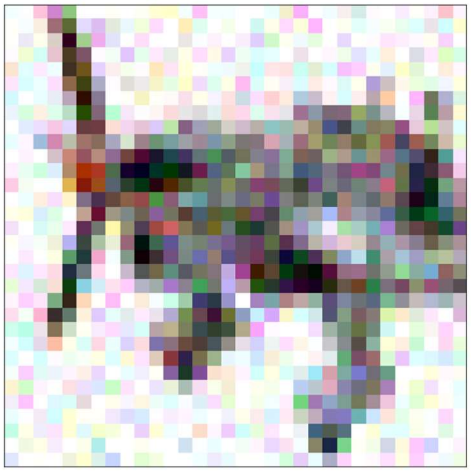
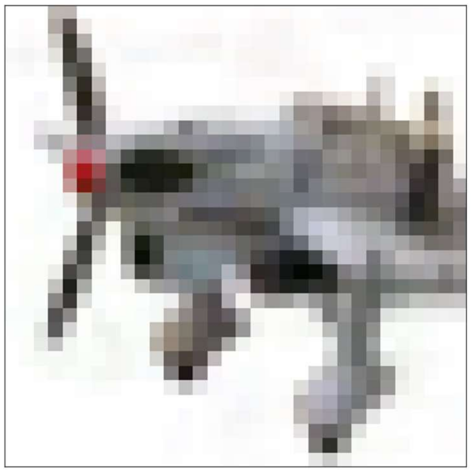

Fig. 21. Noisy image (top), Denoised image (bottom)

*Fig. 22. Noisy images (top), Denoised images (bottom)*

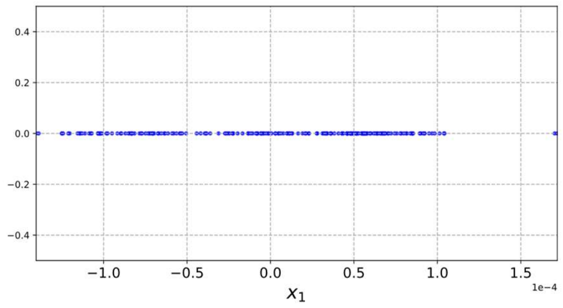

Fig. 23. Spike traces projected on the first principal component

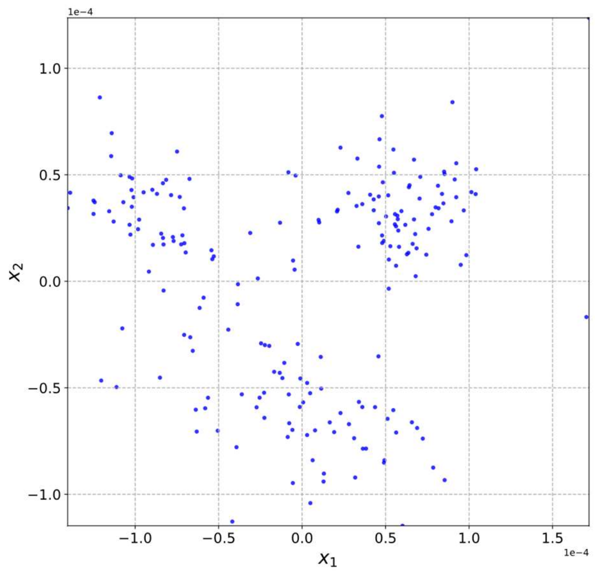

Fig. 24. Spike traces projected on the first two principal components

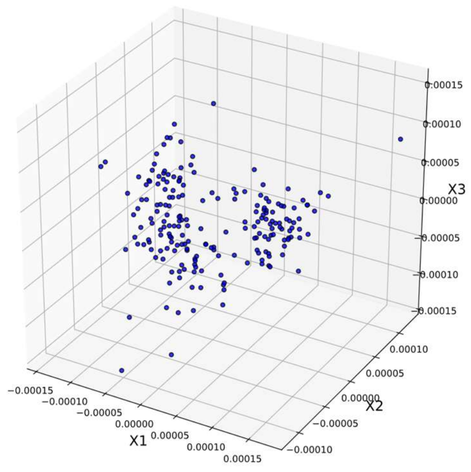

Fig. 25. Spike traces projected on the first three principal components

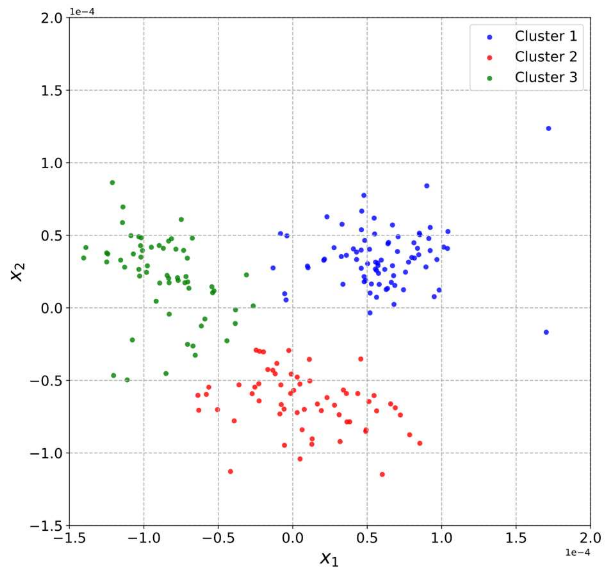

Fig. 26.  Spike traces, clustered in their first two principal component 2D space

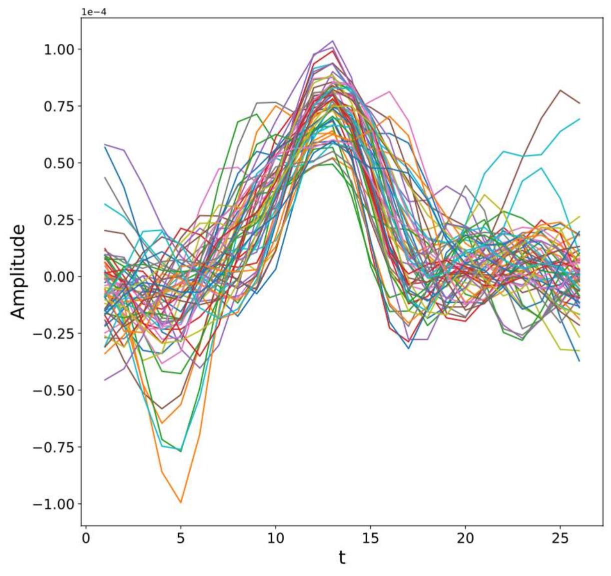

Fig. 28.  Spike traces second cluster

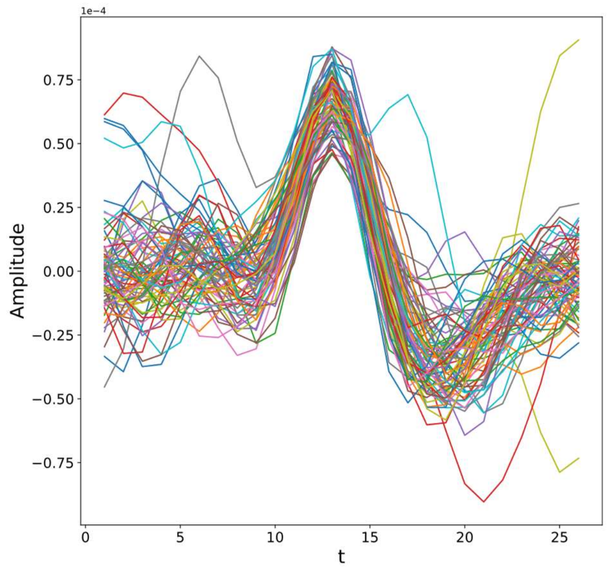

Fig. 27.  Spike traces first cluster

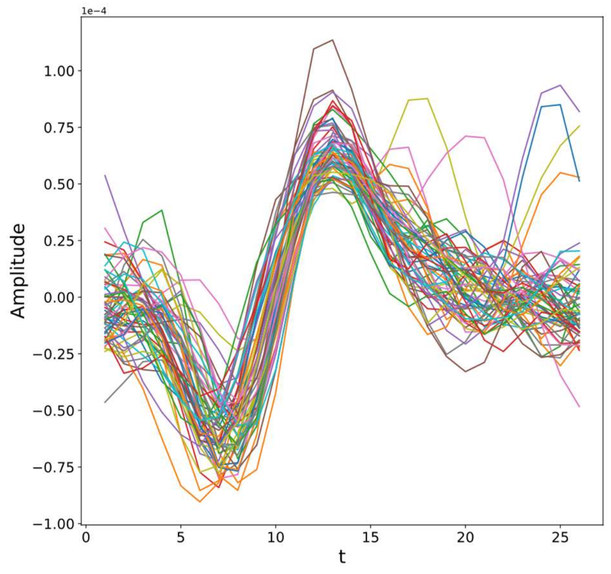

Fig. 29.  Spike traces third cluster