CS596 Final project: source separation with artificial neural networks

Daniel Correa Tucunduva
Computer science department
Bishop's University
Sherbrooke-QC, Canada
dcorrea15@ubishops.ca

Siqian Liu
Computer science department
Bishop's University
Sherbrooke-QC, Canada
sliu17@ubishops.ca

Weina Zhu
Computer science department
Bishop's University
Sherbrooke-QC, Canada
wzhu18@ubishops.ca

Yan Jiang
Computer science department
Bishop's University
Sherbrooke-QC, Canada
wjiang193@ubishops.ca

***Abstract* — Separating components from mixed-sources data is a common challenge in many domains. Independent Component Analysis is the established approach, developed in the 1990s and widely employed. Given the recent advances in artificial neural networks, an exploration of possible alternative approaches relying on neural networks seems worthwhile. We take a preliminary step in that direction by comparing results for a simple synthetic time series of mixed signals. Our findings suggest this might be an interesting avenue for further experimentation.**

Keywords — independent component analysis, neural network, blind source separation

I. Methods

As a base benchmark for comparison, we use Independent Component Analysis (ICA)[1] since it is the fundamental, well established method to perform blind source separation.

As a first architecture of interest we use a simple dense network, considering it is the simplest artificial neural network (ANN) design and therefore an interesting starting point.

As a second architecture of interest we use a simple convolutional neural network (CNN)[2], considering that the core mechanism of convolutions might be valuable for learning local patterns between observation sources that would otherwise be missed.

As a third architecture of interest we use a simple Long Short-Term Memory (LSTM)[3] network, since its core mechanism of recurrent sequential processing is well-suited for time-series data.

As our dataset, we synthesize a 30,000 time points track over a 10 x 10 grid (shape: 30,000 x 10 x 10 x 1), using a sinusoidal, a square and a saw tooth signal. Each grid point is a linear combination of the three signals using random non-zero weights.

This dataset is sufficiently complex to approach possible real datasets, while simple enough to serve the purpose of a preliminary exploration.

II. Development stages

A. *Experiment design*

Our first week of work was dedicated to abstract experiment design and discussion: conceptualizing what type of dataset we would generate, what ANN architectures to explore, and so on.

B. *Data generation and ICA implementation*

Our second week of work was dedicated to implementing the synthetic data generator and the ICA algorithm. The final implementation took only a day, but roughly a week was spent on trial and error.

C. *Neural networks*

Our third week of work was dedicated to implementing the neural networks. This stage was done in parallel, with all three architectures developed simultaneously, with contributions from one or more authors to each.

D. *Revision and cleanup*

Our fourth week of work was spent reviewing and cleaning up the final experiment script.

III. Individual contributions

All authors contributed to experiment design.

D.C.T. was prominently responsible for the data generator, ICA implementation, and CNN design.

Y.J. was prominently responsible for LSTM design.

W.Z. and S.L. were prominently responsible for dense network design and results comparison.

All authors contributed to review and cleanup.

IV. Problems Encountered

A. Environment and tools standardization

The first main problem we encountered was standardizing development environments and choice of tools, since each author has a personal bias toward one option or another.

The solution was using a single Jupyter notebook for the entire experiment source code, due to its universal and portable nature, and leaving runtime details at the discretion of each author.

B. ANNs hyperparameter standardization

The second main problem we encountered was standardizing the train and test inputs and the hyperparameters for all neural networks, to ensure fair comparison.

The solution was implementing all ANNs with a single library (Keras), allowing most hyperparameters to be reused for all ANNs and maximizing fairness in comparisons.

C. ANNs versus ICA standardization

The third main problem we encountered was standardizing the predictions made by the ANNs and the ICA output. The tracks recovered by the ICA algorithm are not ordered, and the y-axis orientation of each track is randomly flipped.

The solution was implementing a simple heuristic to run the ICA algorithm several times, and keep only the lowest error result, calculated against the original ordered tracks.

Acknowledgment

We thank Dr. Mohammed Ayoub Alaoui Mhamdi for the excellent course on Deep Learning, that prepared us to conduct this experiment.

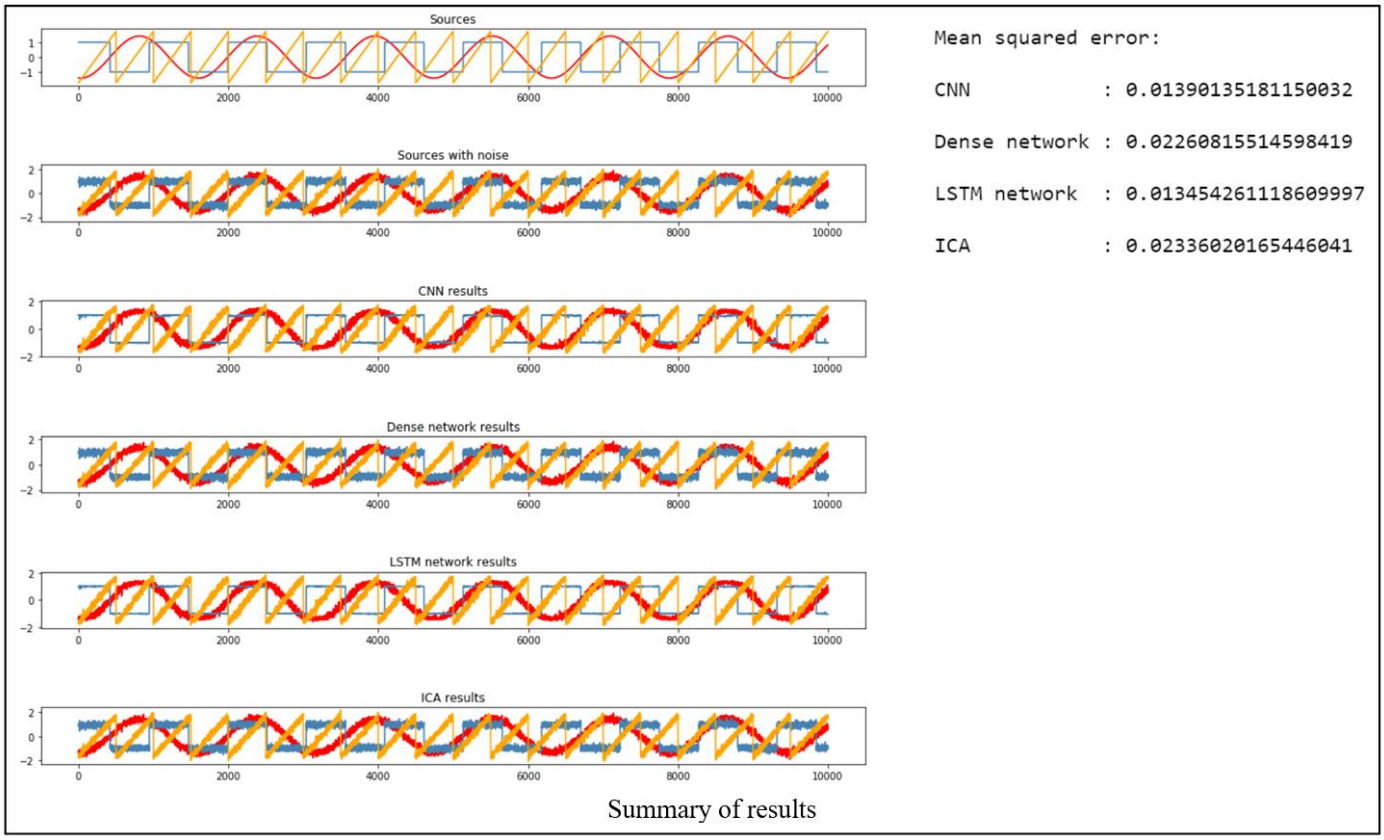
References

[1] P. Comon and C. Jutten, “Handbook of Blind Source Separation: Independent Component Analysis and Applications”, 2010.

[2] M. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014.

[3] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, Neural Computation 9 (8), pp. 1735–1780, 1997.

[4] F. Chollet and others, Keras, 2015, https://keras.io



Summary of results