Internship report

Neural networks as an alternative to i-vectors for speaker verification

Student:
Quentin Fournier

Supervisor:
Christian Raymond

August, 2017

Contents

1 Introduction

Over the last decade[1], the neural approach has been applied with success to many tasks: image and speech recognition, natural language processing, supervised learning and so on. Since then, researchers have tried to solve a wide variety of tasks with neural networks. As part of an internship at the IRISA[2], I investigated the use of neural networks for speaker verification. In particular, I studied data projections through deep encoders as an alternative to i-vector embedding.

With this internship, I aimed to determine whether I want to complete my engineering degree with a PhD. As of now, I have been offered an end-of-study internship and a PhD on the subject of deep learning at the École Polytechnique de Montréal.

In this internship report, I will briefly present the research centre and the team I worked with. Then, I will introduce the deep learning prerequisites for understanding the different experimentations. After presenting the state-of-the-art technique for speaker verification, I will finally detail my work and conclude on its possible applications.

2 IRISA

2.1 Research Centre

The IRISA is a joint computer science research centre gathering CNRS[3], ENS[4] Rennes, Inria[5], INSA[6] Rennes, Université de Bretagne Sud, Université de Rennes 1, Institut Mines-Télécom and Centrale Supélec.

Research topics span from theoretical computer science, such as formal languages, formal methods, or more mathematically oriented topics such as Information theory, optimization, complex systems... to application-driven topics like bioinformatics, image and video compression, handwriting recognition, computer graphics, medical imaging, content-based image retrieval. (Wikipedia)

2.2 Team LinkMedia

The seminal idea of LinkMedia is that of content-based media linking with the ultimate goal of enabling better multimedia applications and new innovative services. Taking a content-based perspective, we seek to create explicit links at different levels to better reflect the context: links at the signal level, e.g., with repeating patterns; links at a semantic level, e.g., to follow topics or stories; links at a paradigmatic level, e.g., to have further details or comments on a topic. (`http://www-linkmedia.irisa.fr/`)

3 Introduction to Deep Learning

3.1 Artificial Neuron

Before introducing whole neural networks, we have to define what a neuron is. A neuron as we understand it in artificial intelligence is a very simple unit that output a usually non-linear function of its weighted inputs (see figure 1).

More formally, a neuron is defined by:

- An input vector $\boldsymbol{x} = (x_1, x_2, ..., x_n)$.

1 More precisely since the deep learning renaissance of 2006 associated to greedy layerwise unsupervised pre-training applied to fully connected architectures [1].
2 Institut de Recherche en Informatique et Systèmes Aléatoires.
3 Centre National de la Recherche scientifique
4 École nationale supérieure
5 Institut National de Recherche en Informatique et en Automatique
6 Institut National des Sciences Appliquées

- A weight vector $\boldsymbol{w}$ such as w_i is the weight associated with its i-th input x_i.
- A bias scalar b shared across every input[7].
- An activation function φ such as a sigmoid, a Rectified Linear Unit (ReLU) or an Exponential Linear Unit (ELU). If you are not already familiar with these functions, I strongly suggest that you look at this article: `https://en.wikipedia.org/wiki/Activation_function`. If you need a more advanced activation function, check out the Scaled Exponential Linear Unit (SELU[8]).

The output of the neuron is then defined by:

$$o = \varphi((\sum_{i=1}^{n} w_i * x_i) + b) = \varphi(\boldsymbol{w} * \boldsymbol{x} + b) \tag{1}$$

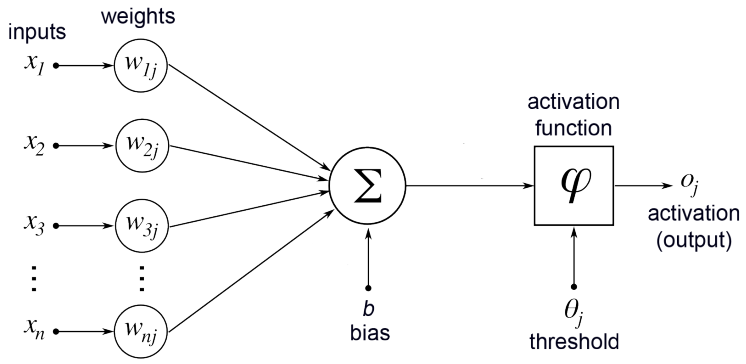


Figure 1: Structure of a neuron. The neuron outputs the result of an activation function φ applied on the biased weighted sum of its n inputs. The subscript j on its weights, output and threshold means that it is the j-th neuron of its layer.

Provided a hyperbolic tangent activation function, a single neuron can perform binary classification (see figure 2).

More complex neural architectures such as Gated Recurrent Unit [3] (GRU) or Long Short-Term Memory [4] (LSTM) units allowing deeper networks have been used during this internship but are beyond the scope of the report.

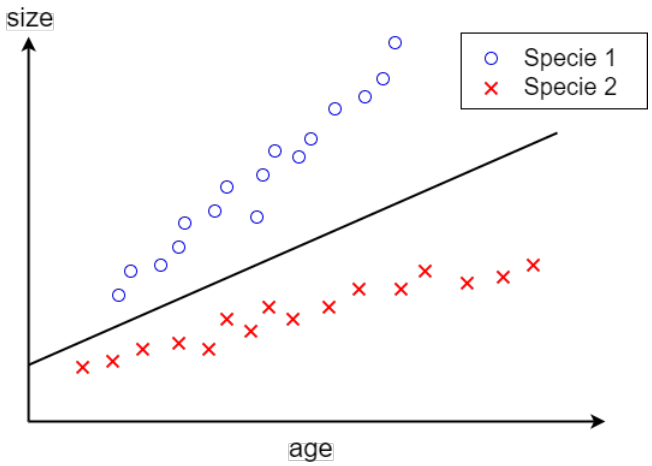


Figure 2: Binary classification performed by a single neuron with two inputs. Let $\boldsymbol{w_1}$ (resp. $\boldsymbol{w_2}$) be the weight associated to the *age* (resp. *size*) and let b be the bias. The decision is made by the sign of $tanh(\boldsymbol{w_1} * x + \boldsymbol{w_2} * y + b)$.

7 If a different bias b_i was associated with each input x_i, we could define a new bias $b = \sum_{i=1}^{n} b_i$.

8 Proposed in July 2017 [2], SELUs seem promising for deep network (8 to 32 layers).

3.2 Neural Networks

Neural networks are a collection of connected neurons usually organized in layers. Parameters of such networks are the weights and bias of every neuron. By learning the right parameters, neural networks are able to solve a wide variety of tasks.

In this section, I will present two major neural network architectures: multilayer perceptrons and autoencoders. However, how a network is able to learn the right parameters is again beyond the scope of this report, but should be well understood before designing neural networks. I recommend you chapters 5 to 6.5 of *Deep Learning* [1] as well as the chapter 2 of *Neural Networks and Deep Learning* [5]. If you were interested in GRUs or LSTM units, the chapter 10.2.2 of *Deep Learning* [1] provides a good explanation of how to compute the gradient in a Recurrent Neural Network (RNN). Both books are available online for free.

3.2.1 Multilayer Perceptron

First introduce in 1957 by Frank Rosenblatt, the perceptron is a linear classifier that can be seen as the simplest single-layer neural network in which every input is linked to a unique output.
In its generalized version, the MultiLayer Perceptron (MLP) is a class of deep fully connected feedforward networks (see figure 3). Despite being very simple, MLPs are able to solve nonlinear problems[9] when trained with backpropagation.

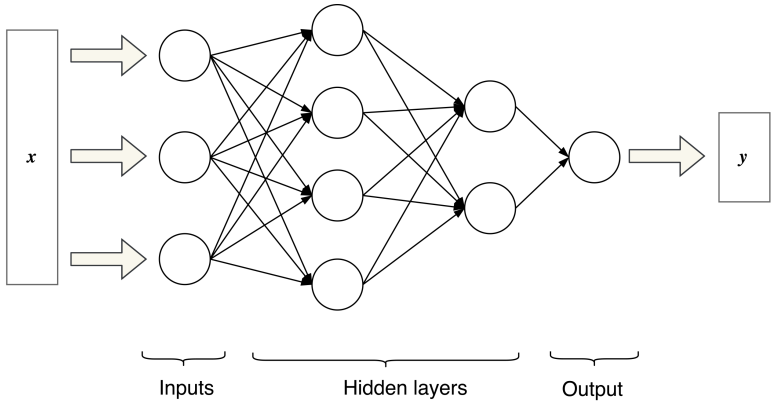


Figure 3: A multilayer perceptron with two hidden layers. Each neuron of layer $k-1$ is connected to every neuron of layer k, if layer k exists, thus this network is fully connected. There is no oriented cycle, thus it is feedforward.

3.2.2 Autoencoder

In 2006, Hinton and Salakhutdinov [6] noted that a deep learning approach could be applied to dimensionality reduction using neural networks that learn to predict their own inputs. Such networks are called *autoencoders* and, once trained, can provide a non-linear dimensionality reduction that outperforms methods based on the singular value decomposition such as PCA.

Given an input $\boldsymbol{x}$, its latent representation[10] $\boldsymbol{z}$ and its reconstruction $\boldsymbol{x'}$, an autoencoder is composed of two symmetric networks (see figure 4):

- Encoder: characterized by the function $f(\boldsymbol{x}) = \boldsymbol{z}$ such that $\boldsymbol{x}$ is the input and $\boldsymbol{z}$ is the output of the network.
- Decoder: characterized by the function $g(\boldsymbol{z}) = \boldsymbol{x'}$ such that is $\boldsymbol{z}$ the input and $\boldsymbol{x'}$ is the output of the network.

9 Discriminate data that is not linearly separable.
10 *Latent* as in latent neuron, latent layer or latent representation means hidden, not directly observable.

The training objective is to minimize the distance between the input $\boldsymbol{x}$ and its reconstruction $g(f(\boldsymbol{x})) = \boldsymbol{x'}$. It is necessary to limit the capacity of the model to copy its input on its output in order to force the autoencoder to extract useful properties.

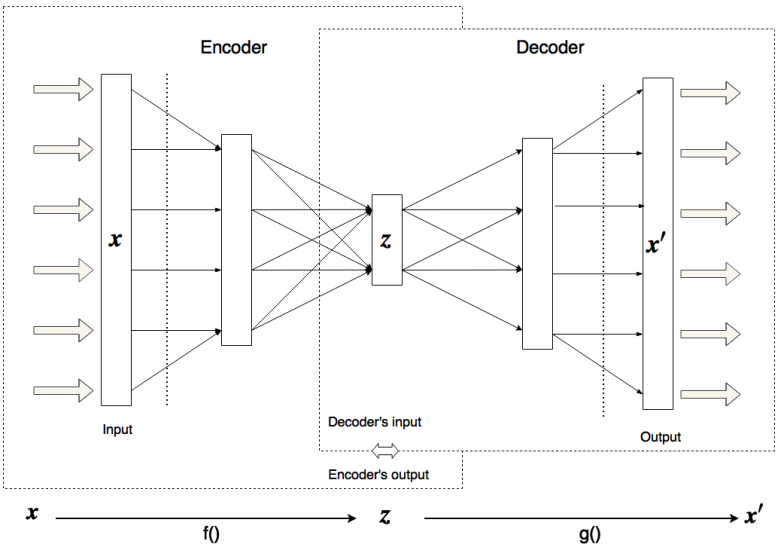


Figure 4: Framework of an autoencoder. $dim(\boldsymbol{z}) < dim(\boldsymbol{x})$, thus the autoencoder is said undercomplete. See Hinton and Salakhutdinov [6] for an in-depth review of the framework.

When applied to dimensionality reduction, the autoencoder is trained with both encoder and decoder networks, then the decoder is discarded and the output of the encoder is treated as the data's projection. This approach yields a non-linear generalization of PCA provided that both encoder and decoder have at least one hidden layer.

4 Experiments

Recurrent neural networks[11] are state-of-the-art for some speech-related tasks like speech recognition [7] since 2012. However – to my knowledge – i-vectors remain the state-of-the-art technique for speaker verification although promising architecture like *TristouNet*[8] have been proposed.

Before introducing the subject any further, please note the difference between:

- Speech recognition: what is being said.
- Speaker recognition: who is speaking.
- Speaker verification: are two utterances from the same speaker.

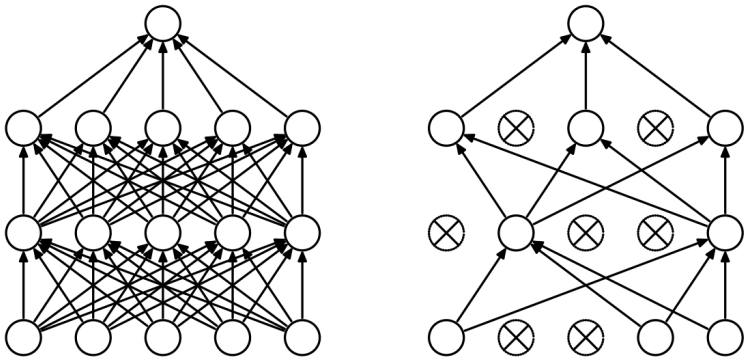
4.1 Regularization Techniques

Every network was regularized using:

- Dropout: the key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting too much. [...] This significantly reduces overfitting and gives major improvements over other regularization methods. (N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov in *Dropout: A simple way to prevent neural networks from overfitting* [9]).

11 More precisely deep RNNs with LSTM units and a proper regularization.

- Batch Normalization: in order to avoid *internal covariance shift*[12], batch normalization layers maintain the mean activation close to 0 and the activation standard deviation close to 1 at each batch[13]. [10]



(a) Standard Neural Net (b) After applying dropout.

Figure 5: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped. Source: *Dropout: A simple way to prevent neural networks from overfitting* [9]

.

Note that the low depth of the neural networks used during this internship made dropout and batch normalization optional. However dropout still reduced the impact of too wide layers and batch normalization speeded up the training.

In the next section, I will present the state-of-the-art procedure for speaker verification as well as the dataset and the metrics. Then, I will present the results of various deep neural architectures that have been applied to the task.

4.2 State-of-the-art

4.2.1 Supervectors

Given a dataset of audio shows:

1. Each show is divided into single-speaker segments (each segment corresponding to one utterance).
2. Each segment is transformed into a vector of Mel-Frequency Cepstral Coefficients (MFCCs, see figure 6a). For simplicity, we refer to MFCCs vectors as *m-vector*.
3. A Gaussian Mixture Model[14] (GMM) called the Unified Background Model (UBM) is trained on every m-vectors with Esperance Maximization[15] (EM) algorithm.
4. For each segment, a speaker-GMM is obtained by adjusting the UBM parameters using Maximum A Posteriori[16] (MAP) algorithm on the m-vector (see figure 6b).
5. Each segment is represented by the vector of the speaker-GMM means. We refer to this vector as the *supervector*.

12 The distribution of each layer's input changes during training, as the parameters of the previous layers change [10]. This phenomenon makes learning delicate.

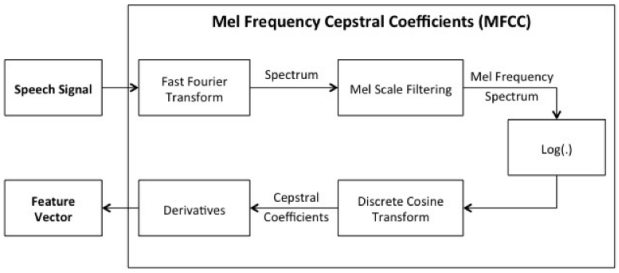
13 Set of examples processed before updating the network parameters.

14 A Gaussian mixture model is a probabilistic model used to approximate a distribution of random variables as a sum of normal distributions.

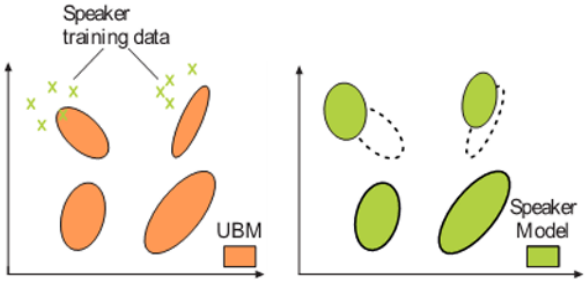
15 See chapter 19.2 of *Deep Learning* [1] for a good explanation of the EM algorithm.

16 See chapter 19.3 of *Deep Learning* [1] for a good explanation of the MAP algorithm.

To discriminate whether two supervectors are uttered by the same speaker, we wish to find a lower-dimensionality representation and evaluate the cosine similarity between them. If the cosine similarity is below (resp. above) the decision threshold, then the two segments are classified as uttered by different speakers (resp. same speaker). As of the writing of this report, the state-of-the-art lower-dimensionality representation for supervectors is i-vectors.



(a) MFCCs flowchart. MFC is a representation of the short-term power spectrum of a sound whose coefficients are commonly used as features in speech recognition systems. Source: `http://recognize-speech.com/`.



(b) GMM-UBM. **Left**: the UBM, a GMM learned on every m-vectors using the EM algorithm. **Right**: the speaker-GMM obtained by adjusting the UBM to one m-vector using the MAP algorithm. Source: `http://dynadmic-lab.com/`.

4.2.2 i-vectors

The core idea behind i-vectors is that a supervector should be decomposable into a speaker independent component and a low-dimension total-variability component. Formally:

$$\boldsymbol{s} = \boldsymbol{m} + \boldsymbol{T} * \boldsymbol{w} \tag{2}$$

Where:

- $\boldsymbol{s}$: the supervector.
- $\boldsymbol{m}$: the speaker independent supervector (vector composed of the UBM Gaussian means).
- $\boldsymbol{T}$: the total-variability matrix.
- $\boldsymbol{w}$: the i-vector.

In order to keep this report short, I will not detail how to evaluate T. A clear explanation can be found in Howard Lei's introduction to joint factor analysis [11].

4.3 Dataset

For the experiments, I have been provided with 17,172 single speaker supervectors from the audio tracks of the *REPERE* corpus [12]. These supervectors come from samples that have been extracted from 9 distinct TV shows and uttered by 881 distinct speakers.

The dataset is quite challenging. The recording takes place both inside a studio setting and outside in public and noisy environments. Apart from this, music is often played in the background during certain presentations or interviews. Additionally, there is a significant imbalance between speakers, with anchors and top politicians both being often over-represented in the dataset. (M. Budnik, L. Besacier, A. Khodabakhsh and C. Demiroglu in *Deep complementary features for speaker identification in TV broadcast data* [13])

The dataset was beforehand split into a train set (15,308 supervectors), a validation set (953 supervectors) and a test set (911 supervectors).

4.4 Metrics

A binary classifier discriminates data into two classes which I will call *positive* and *negative*. Thus, such a classifier can only commit two types of error: false positive and false negative (see table 1).

Table 1: Every binary classifier possible output. The misclassification of two sequences from two different speakers as uttered by the same speaker is called a false positive. The misclassification of two sequences from the same speaker classified as uttered by two different speakers is called a false negative. The higher (resp. lower) the decision threshold is, the higher the false negative (resp. positive) rate is.

		True label	
		Positive	Negative
Classification	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

In section 4.5, I will compare state-of-the-art i-vectors to various neural networks based on:

- Detection Error Tradeoffs (DET): False Positive Rate (FPR) in function of False Negative Rate (FNR). To plot the DET curve, one evaluates FNR and FPR for different thresholds. Note the logarithmic scale.
- Equal error rate (EER): when FPR and FNR are equal. EER is evaluated within a margin of error of $\pm$ 0.1%.
- Distribution of all pairs in function of the cosine similarity[17]. One wishes to have a cosine similarity close to 0 for all pairs uttered by different speakers, and close to 1 for all pairs uttered by the same speaker. However, a clear separation between the distribution of the two classes suffice to ensure a good classification.

Note that I will use the formulation *"A has better results than B"* which should be understood as A having a DET curve – mostly – under the B's and a lower EER.

4.5 Results

The experiments were conducted on Ubuntu 16.04 with the library Keras 2.0[14] and Theano 0.9. Thanks to CUDA 8.0, a GPU (Titan XP) has been used to accelerate the training. However, I did not have the time nor the computational resources to do an automatic hyperparameters optimization, resulting in slightly higher error rates.

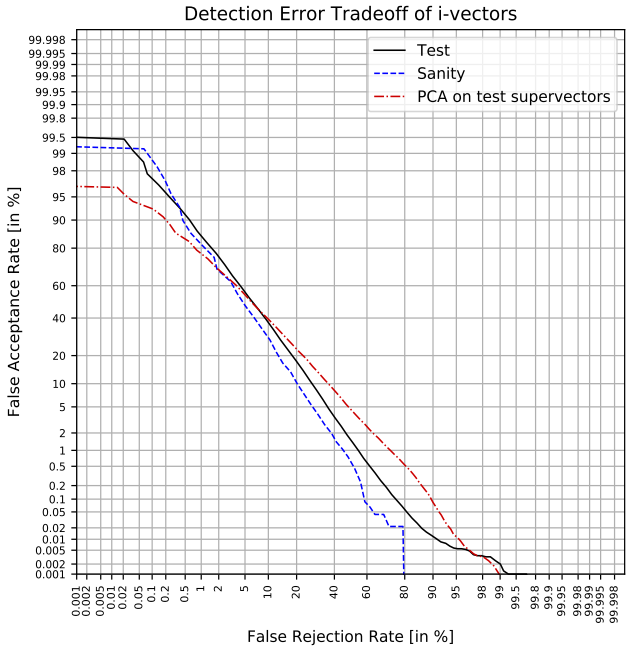
In order to keep the same notation as i-vectors, I will call *d-vectors* the projection of supervectors by neural networks.

Every network is fully detailed in appendix for future research.

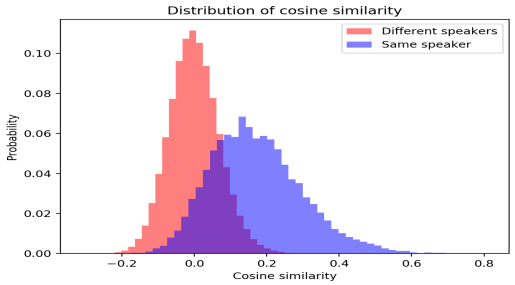
4.5.1 I-vectors

In order to justify the baseline, the model has been evaluated on a subset of the train set (see figure 7). This is called a sanity check and should always be performed to ensure the validity of a model. Ideally, a model should be able to generalize well to new data, thus providing slightly better results on any train subset than on the test set.
One can learn from a failed sanity check: if the results are significantly better on the train subset than on the test set, the model is said to *overfit* which can be solved by reducing its capacity or adding a possibly stronger regularization techniques.

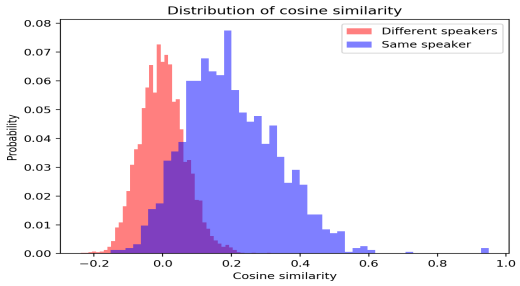
17 The cosine similarity or proximity between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is given by $\frac{\boldsymbol{a}.\boldsymbol{b}}{||\boldsymbol{a}||.||\boldsymbol{b}||}$



(a) DET curves of i-vectors on the test (EER of 18.9%) and sanity set (EER of 15.5%). The DET curve of a simple PCA on test set supervectors (EER of 21.4%) is drawn to put in perspective a simpler approach.



(b) Distribution of all i-vectors pairs in the **test set**. Surprisingly, i-vectors do not provide the clear separation that we could expect.



(c) Distribution of all i-vectors pairs in the **sanity subset**. Even though distributions are better separated, they still overlap.

Figure 7: I-vectors sanity check.

The sanity check of i-vectors is conclusive. However the baseline may not be state-of-the-art since a simple PCA yields comparable results (with the same number of composantes, see figure 7c). Note that Within-Class Covariance Normalization (WCCN) has been applied [15].
I will continue to use i-vectors as the baseline because every experiment can easily be recomputed with state-of-the-art i-vectors.

4.5.2 Autoencoders

Three different autoencoders were used for the experiments:

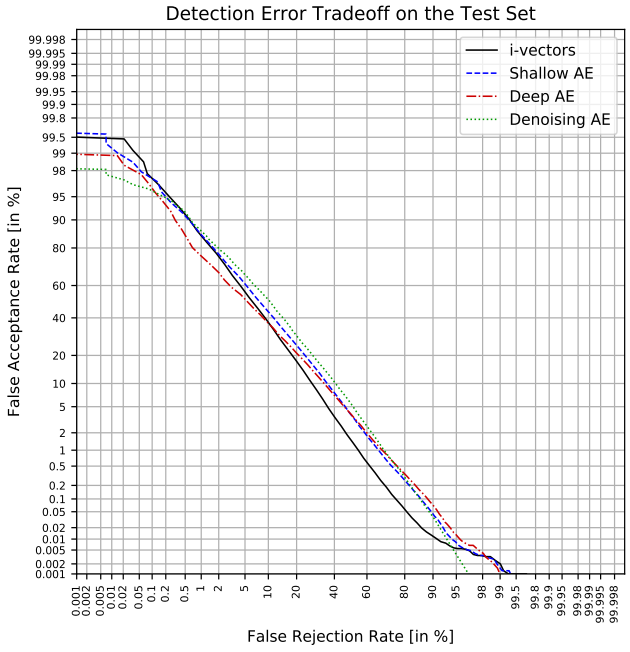
- Shallow autoencoder: it is the simplest autoencoder with only one hidden layer – which one can consider visible as it is the encoder's output (see layer $\boldsymbol{z}$ in figure 4). Provided linear activation, such a network yields a PCA.
- Deep autoencoder: a more permissive autoencoder with multiple hidden layers as presented in figure 4.
- Denoising autoencoder: an autoencoder whose input has been corrupted by some sort of noise. In our experiments, the loss function is given by $L(\boldsymbol{x_1}, g(f(\boldsymbol{x_2})))$ with $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ two distinct supervectors uttered by the same speaker. The idea behind denoising i-vectors is to learn what does not change within all same speaker utterances.

It is impossible to backpropagate – which is the common method for training a neural network – through a ReLU when it outputs zero because its gradient is not defined[18]. That means that the weights of a neuron with a ReLU activation which outputs zero will not be able to change. Such a neuron is said *dead*. Because many supervector components were negative, the first autoencoders

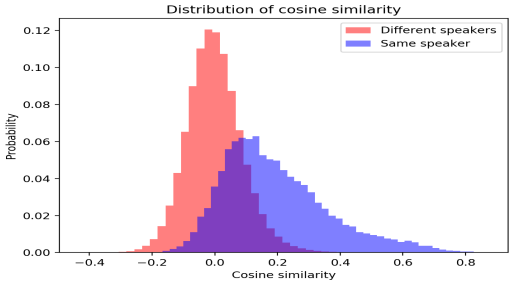
18 Note it is possible to compute the backrpopagation term as the gradient is multiplied by the incoming value which is zero. However the parameters cannot be updated as zero is a factor of the whole expression.

gave terrible results. The problem was solved using Exponential Linear Unit (ELU) with a coefficient $\alpha = 0.3$, yielding a strictly positive gradient on $\mathbb{R}$. SELUs did not provide better results as none of the networks were not deep enough.

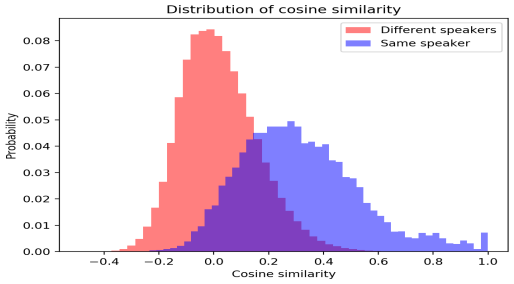
Every autoencoder's projection of supervectors is outperformed by i-vectors (see figure 8). Autoencoders probably extract additional information that is not relevant in speaker verification.
As expected, the deep autoencoder outperforms the shallow one. The denoising autoencoder should outperform the other autoencoders as it benefits from more training examples. Indeed, there are 15,308 supervectors in the train set which represent more than 3 million different same-speaker pairs. However, the denoising autoencoder is prone to overfit and needs special care in choosing regularization techniques.
It would be interesting to perform WCCN on autoencoders output, or to use a variational autoencoder whose projection provides similar properties.



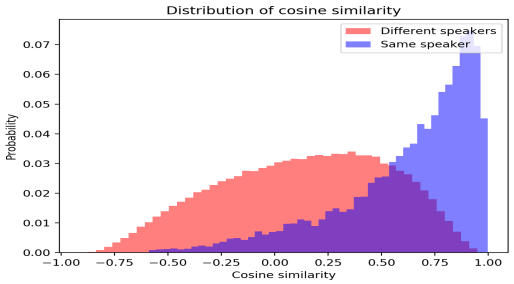
(a) DET curves of i-vectors (EER of 18.9%) and d-vectors from shallow autoencoder (EER of 22.1%), deep autoencoder (EER of 20.5%) and denoising autoencoder (EER of 24.1%) evaluated on the test set.



(b) Distribution of shallow autoencoder d-vector.



(c) Distribution of deep autoencoder d-vector.



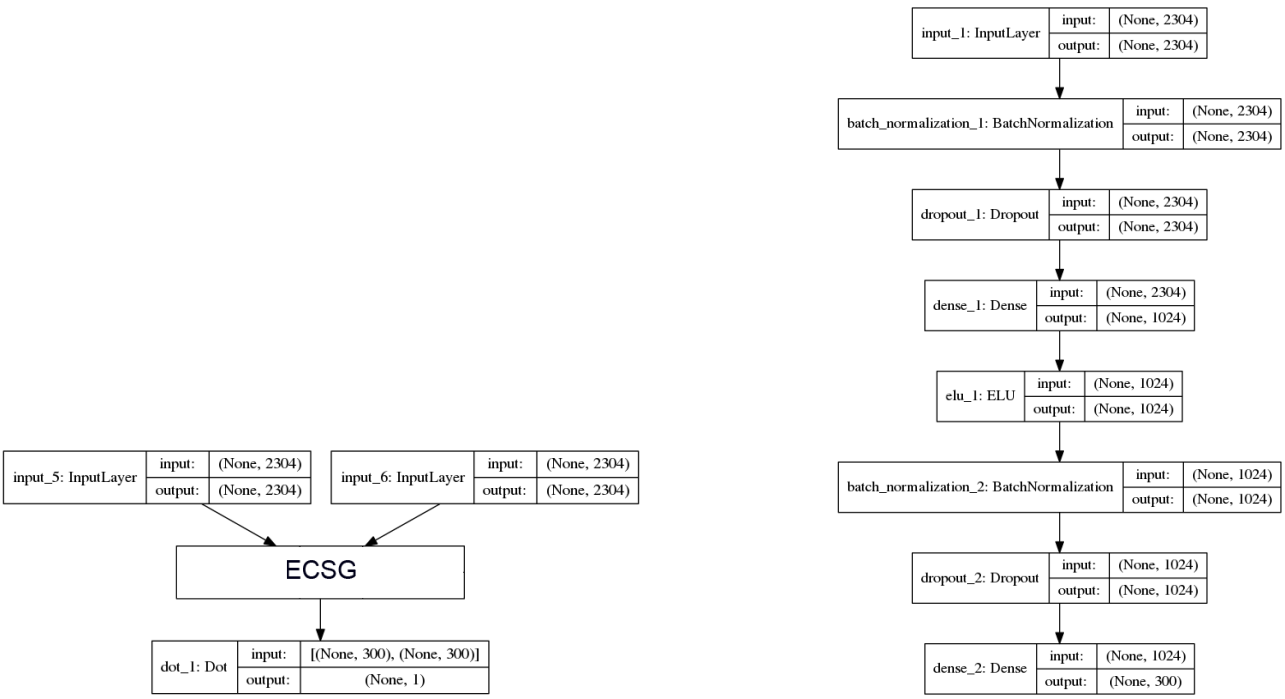
(d) Distribution of denoising autoencoder d-vector.

Figure 8: Shallow, deep and denoising autoencoders yield slightly worse – yet comparable – results than i-vectors.

4.5.3 Encoder Cosine Similarity Guided

The Encoder Cosine Similarity Guided (ECSG) is a MLP with only one hidden layer and exponential linear units (see figure 9b). It is worth noting that ReLUs performed nearly as well.
During training, the ECSG was fed a pair of supervectors (see figure 9a). Then the normalized dot

product (which equals the cosine similarity) of the two d-vectors was compared to the expected value (1 if the pair was uttered by the same speaker, else 0) using mean square error. The network was trained with an adaptive learning rate optimizer.



(a) Architecture of the training model.

(b) Architecture of the ECSG.

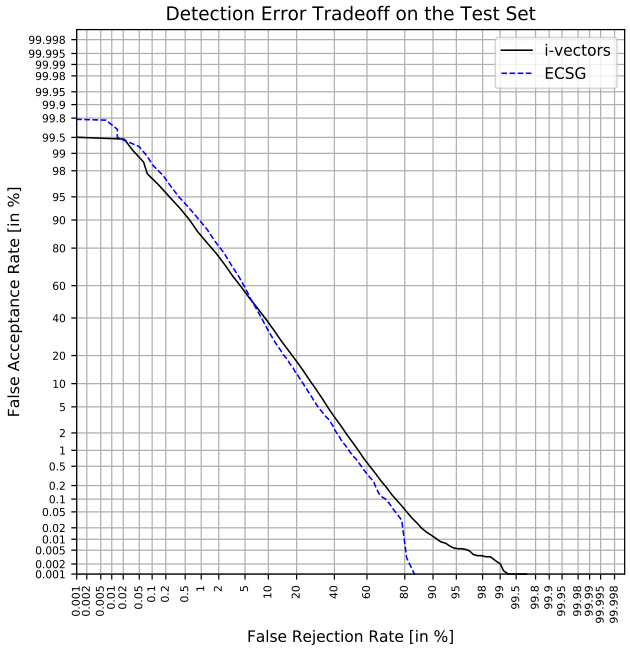
Figure 9: ECSG architecture.

The ECSG was trained with all positive pair (3,678,862 pairs) and N times more negative pairs. At the beginning of each epoch[19], the negative pairs were drawn amongst all negative pairs (230,640,694 pairs). When N increase, the network is trained with more negative pairs which can be useful to generalize as there are more negative than positive pairs by nature. But if N is too large, the probability for a couple of supervectors to be uttered by the same speaker is too low and the network always predict that two pairs are uttered by different speakers. Empirically, we found that $N = 2$ gave good results.

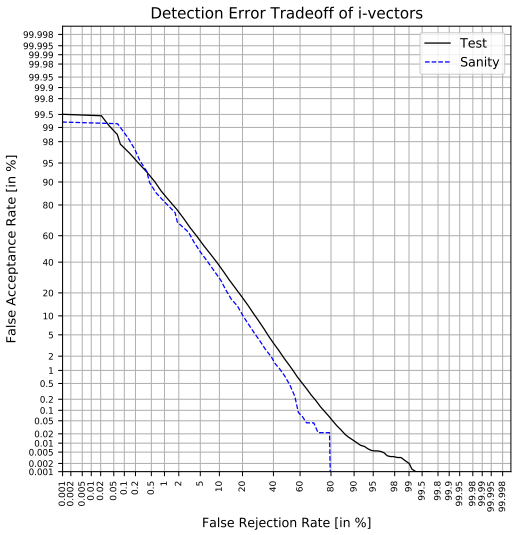
Overall, the ECSG projection outperforms i-vectors (see figure 10a). However, i-vectors provide a lower FPR for a FNR inferior to 5%. An automatic search of N, albeit expensive, should improve d-vectors for low FNR.

The sanity check on ECSG d-vectors is conclusive (see figure 10b).

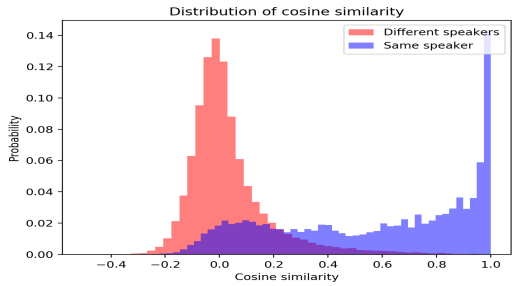
19 Not to be mistaken with iteration. An epoch corresponds to one pass of all the training examples



(a) DET curves of i-vecctors (EER of 18.9%) and ECSG d-vectors (EER of 17.0%) evaluated on the test set.



(b) DET curves of ECSG d-vectors evaluated on the test set (EER of 17.0%) and on the sanity set (EER of 12.1%). The sanity check is conclusive.



(c) Distribution of all pairs in function of the cosine similarity. There is a clear separation between the two classes. Note that the peaks of probability are much higher than for i-vectors.

Figure 10: Results of ECSG

5 Conclusion

The encoder cosine similarity guided is a neural network build only for the task of speaker verification and yields an equal error rate improvement of 1.9% over i-vectors[20]. Futhermore, the ECSG is able to improve one error rate (FPR or FNR) at the expense of the other by varying N. That makes this network suitable for critical applications like court where the false positive rate should be the lowest possible.

However, i-vectors are an unsupervised technique that may be applied to a wide variety of data. Neural networks that provide the best results are supervised, and build for a specific dataset.

20 Note that i-vectors used for the experiments may not be state-of-the-art.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[2] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, *Self-normalizing neural networks*, cite arxiv:1706.02515Comment: 9 pages (+ 93 pages appendix), 2017. [Online]. Available: http://arxiv.org/abs/1706.02515.

[3] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks", *CoRR*, vol. abs/1502.02367, 2015. [Online]. Available: http://arxiv.org/abs/1502.02367.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997, ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[5] M. Nielsen, *Neural Networks and Deep Learning*. 2017. [Online]. Available: http://www.neuralnetworksanddeeplearning.com/.

[6] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks", *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006. DOI: 10.1126/science.1127647. [Online]. Available: http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google.

[7] A. Graves, A. r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks", in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6645–6649. DOI: 10.1109/ICASSP.2013.6638947.

[8] H. Bredin, "Tristounet: Triplet loss for speaker turn embedding", *CoRR*, vol. abs/1609.04301, 2016. [Online]. Available: http://arxiv.org/abs/1609.04301.

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014, ISSN: 1532-4435. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2670313.

[10] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift", *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167.

[11] H. Lei, *Joint factor analysis (jfa) and i-vector tutorial*. [Online]. Available: http://www1.icsi.berkeley.edu/Speech/presentations/AFRL_ICSI_visit2_JFA_tutorial_icsitalk.pdf.

[12] A. Giraudel, M. Carré, V. Mapelli, J. Kahn, O. Galibert, and L. Quintard, "The repere corpus : A multimodal corpus for person recognition", in *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, N. C. (Chair), K. Choukri, T. Declerck, M. U. Doğan, B. Maegaard, J. Mariani, A. Moreno, J. Odijk, and S. Piperidis, Eds., Istanbul, Turkey: European Language Resources Association (ELRA), May 2012, ISBN: 978-2-9517408-7-7.

[13] M. Budnik, L. Besacier, A. Khodabakhsh, and C. Demiroglu, "Deep complementary features for speaker identification in tv broadcast data", in *Odyssey: The Speaker and Language Recognition Workshop*, 2016, pp. –.

[14] F. Chollet *et al.*, *Keras*, https://github.com/fchollet/keras, 2015.

[15] Y. Xing, P. Tan, and C. Zhang, "Improved i-vector speaker verification based on wccn and zt-norm.", in *CCBR*, Z. You, J. Zhou, Y. Wang, Z. Sun, S. Shan, W.-S. Zheng, J. Feng, and Q. Zhao, Eds., ser. Lecture Notes in Computer Science, vol. 9967, 2016, pp. 424–431, ISBN: 978-3-319-46653-8. [Online]. Available: http://dblp.uni-trier.de/db/conf/ccbr/ccbr2016.html#XingTZ16.

[16] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, "End-to-end text recognition with convolutional neural networks", in *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Nov. 2012, pp. 3304–3308.

List of Abbreviations

CNN	Convolutional Neural Network 13
DET	Detection Error Tradeoff 7, 8, 13
ECSG	Encoder Cosine Similarity Guided iii, 9–11, 13
EER	Equal Error Rate 7–9, 11, 13
ELU	Exponential Linear Unit 2, 9, 13
EM	Esperance Maximization 5, 6, 13
FNR	False Negative Rate 7, 10, 11, 13
FPR	False Positive Rate 7, 10, 11, 13
GMM	Gaussian Mixture Model 5, 6, 13
GPU	Graphical Processing Unit 7, 13
GRU	Gated Recurrent Unit 2, 3, 13
LSTM	Long Short-Term Memory 2–4, 13
MAP	Maximum A Posteriori 5, 6, 13
MFCC	Mel-Frequency Cepstral Coefficient 5, 6, 13
MLP	Multilayer Perceptron 3, 9, 13
MSE	Mean Square Error i–iii, 13
PCA	Principal Component Analysis 3, 4, 8, 13
ReLU	Rectified Linear Unit 2, 8, 9, 13
RNN	Recurrent Neural Network 3, 4, 13
SELU	Scaled Exponential Linear Unit 2, 9, 13
UBM	Unified Background Model 5, 6, 13
WCCN	Within-Class Covariance Normalization 8, 9, 13

A Architectures

A.1 Autoencoders

A.1.1 Shallow Autoencoder

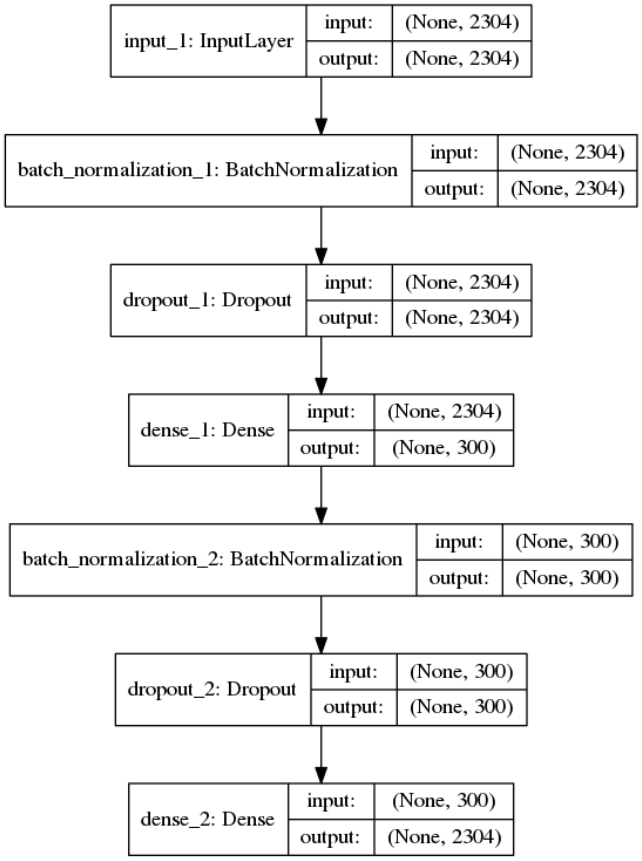


Figure 11: Architecture of the shallow autoencoder.

Epochs:	100
Batch size:	128
Activations:	linear
Optimizer:	Nadam.
Loss function:	MSE
Dropout:	0.25
Early stopping:	Yes
Kernel initializer:	He normal

A.1.2 Deep and Denoising Autoencoders

Both autoencoders have the same architecture and parameters.

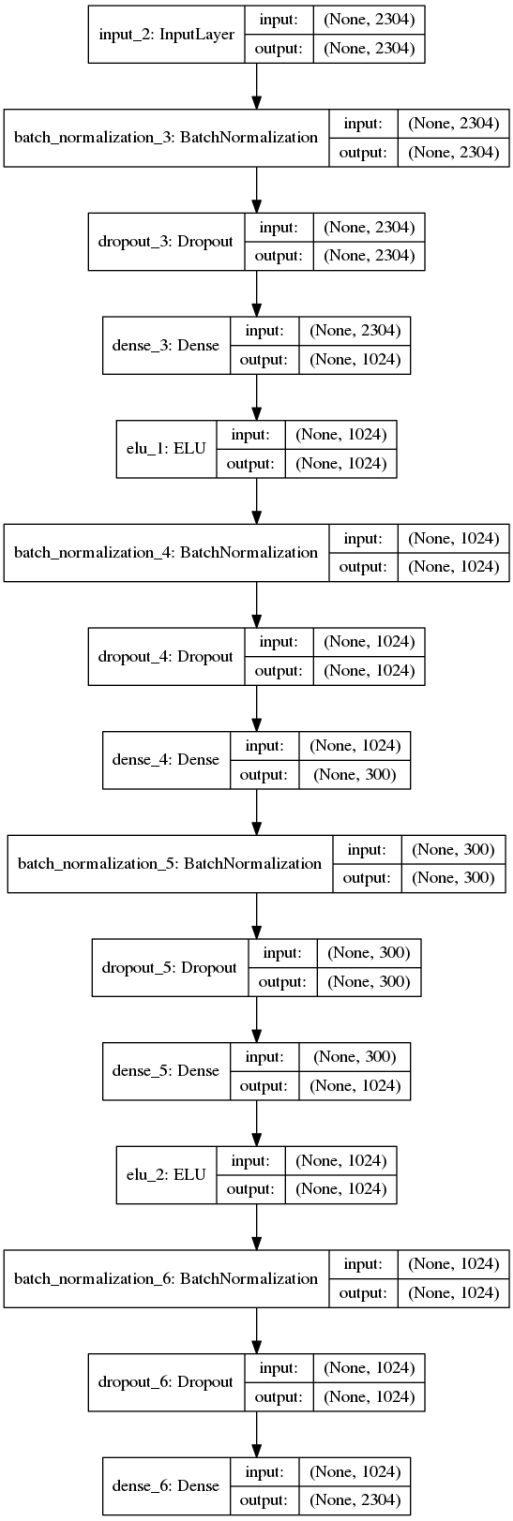


Figure 12: Architecture of the deep autoencoder.
It is the same as for the denoising autoencoder.

Epochs:	100
Batch size:	128
Activations:	linear for output layers, else ELU
Optimizer:	Nadam
Loss function:	MSE
Dropout:	0.25
Early stopping:	Yes
Kernel initializer:	He normal

A.2 Encoder Cosine Similarity Guided

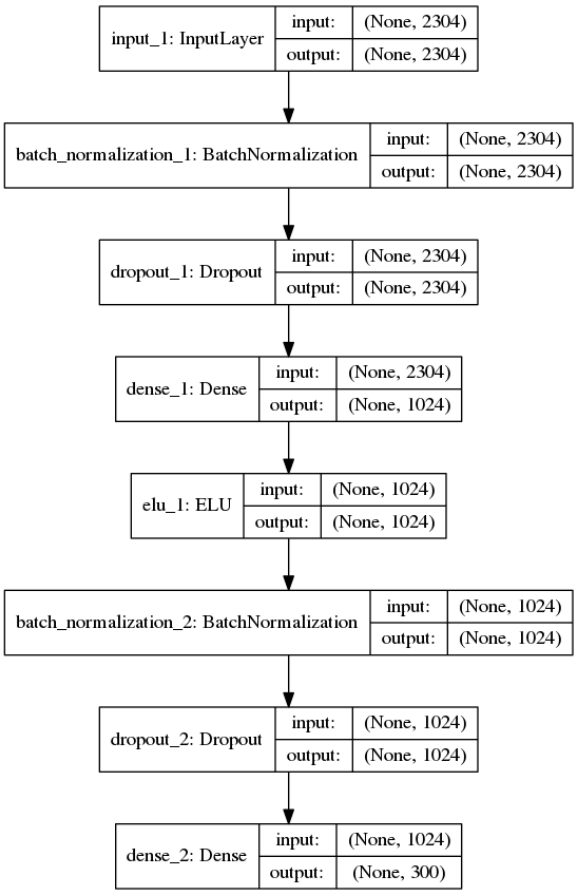


Figure 13: Architecture of the ECSG.

Epochs:	5
Batch size:	128
Activations:	linear for output layer, else ELU
Optimizer:	Nadam
Loss function:	MSE
Dropout:	0.5
Early stopping:	No
Kernel initializer:	He normal