Combining sequential deep learning and variational Bayes for semi-supervised inference

Jos van der Westhuizen
Department of Engineering
Cambridge University
Cambridge, CB2 1TN
jv365@cam.ac.uk

Dr. Joan Lasenby
Department of Engineering
Cambridge University
Cambridge, CB2 1TN

Abstract

In the application of machine learning to time series, coarse labelling of the sequence is generally known. However, often a finer granularity of the annotations is sought for improved accuracy and resolution in signal analysis. With a focus on medical time series, this study employs a bidirectional LSTM autoencoder, t-SNE, and variational Bayesian estimation in order to provide labels with finer granularity. Whether the proposed relabelling improves classification remains inconclusive due to insufficient data, but the semi-supervised approach has the potential for further refinement and widespread adoption.

1 Introduction

To enable artificial intelligent systems to equal or surpass the ability of the human brain, effective techniques to learn representations from unlabelled data are imperative (Murphy, 2012). Real life time series often lack high resolution in their annotations, and consequently, assumptions have to be made about lower-level temporal aspects. Moreover, high definition labelling of big data is too expensive and the amount of unlabelled data is still orders of magnitude larger than the amount of labelled data. Annotations can also be ambiguous or incorrect, and few real-world datasets have labels that are 100% accurate (Russakovsky et al., 2014).

Medical time series belong to a class of data that is prone to low granularity in the annotations (Längkvist et al., 2014), and these annotations are often incorrect due to human error. If labelled, these signals have a known final outcome, but the states and transitions of segments within the signals are unknown. For example, a patient with a healthy outcome that survived a cardiac arrest during their 24h stay in an intensive care unit would have had several periods in his/her signal indicative of deterioration and death. Previous studies on medical time series revert to low resolution data (Lipton et al., 2015; Choi et al., 2015) or accept the decrease in performance caused by incorrect annotations (Oresko et al., 2010; Lugovaya, 2005). This research proposes a semi-supervised approach that is able to group similar segments and, combined with the final outcomes, provide finer granularity in the annotations of medical signals.

Autoencoders are Neural Network models that are trained to learn representations of the input data in the hidden layers of the network and reconstruct the input data from the hidden layers. This provides a variant of feature selection similar to that of Principal Component Analysis (PCA) (Hinton and Salakhutdinov, 2006). Long Short-Term Memory (LSTM) Recurrent Neural Networks allow serendipitous discovery of important long and short term features in time series (Lipton et al., 2015). Thus an LSTM autoencoder is used here to improve the precision of the time series feature analysis. Additionally, the entire dataset is subjected to the relabelling process, and online analysis with the technique developed is not necessary. This enables the use of a bidirectional LSTM, which increases the accuracy of the standard LSTM by adding a backwards LSTM layer to the model.

Workshop on Bayesian Deep Learning, NIPS 2016, Barcelona, Spain.

After training the autoencoder, the data is encoded into the latent representation. The dimensionality of the latent representation is then reduced to 2 dimensions and visualised using the t-Distributed Stochastic Neighbour Embedding (t-SNE) method (van der Maaten and Hinton, 2008). The dimensionality reduction and simultaneous grouping of similar samples of t-SNE enables Gaussian Mixture Model (GMM) clustering of the samples in the new 2D manifold. The GMM clustering is used to relabel samples that are grouped based on the final outcome labels. The newly labelled dataset is then subjected to an LSTM classifier to compare classification performance with the original dataset. The efficacy of this approach was analysed using 2 different datasets. The implementation software used for this study was Python 2.7, Theano (Theano Development Team, 2016), and Tensorflow (Abadi et al., 2015).

2 Related work

In this section relevant previous research aiming to leverage unsupervised learning is reviewed.

In medicine, Miotto et al. (2016) made use of unsupervised learning to capture hierarchical regularities and dependencies in the electronic health records (EHRs) for 704,857 patients. To this end, a denoising-autoencoder was implemented. A denoising-autoencoder is trained to explicitly extract relevant information from the input data by reconstructing the original input data from input data with added noise. This implementation added noise to the input data by setting a randomly selected 5% of the data to zero. Experimentation was done with up to 7 hidden layers in the autoencoder, and the best area under the receiver operating characteristic of 0.77 was achieved with 3 hidden layers. The autoencoder outperformed the raw data baseline and PCA in this task. This study shows the merit in using autoencoders for rich latent representations, which we adopted in our work.

Unsupervised learning holds great potential for sequential data problems such as natural language processing (NLP). Ammar et al. (2014) implemented a conditional random field (CRF) autoencoder to infer structure predictors from the latent representation generated. The focus was on sequential latent structures with first order Markov properties in NLP. CRF autoencoders differ from neural network (NN) autoencoders in that they learn specific *interpretable* regularities of interest instead of feature representations. The study demonstrated that the CRF autoencoder outperforms Hidden Markov Models for unsupervised structure prediction. LSTM and CRF autoencoders are similar in that they are both sequential variants of the standard autoencoder.

Srivastava et al. (2015) made use of an LSTM autoencoder to reconstruct video frames. The encoder LSTM mapped an input sequence to a fixed length representation, and the decoder LSTM would decode the representation in a reversed order as illustrated in Figure 1. The reversed order makes optimisation easier because the model can start with attention to low range correlations. The study made use of an LSTM with 128 hidden units and it was found that a bigger LSTM did not improve the predictive accuracy for future video frames. Two reasons prevent the autoencoders from learning a trivial identity mapping: i) There is a fixed number of hidden units, making it unlikely to learn a trivial mapping for arbitrary length input sequences; and ii) the same dynamics have to be applied recursively on the same representation. For our implementation, only the latter holds because the input sequences are segmented to have equal lengths. Thus we implemented additional regularisation to mitigate identity mapping.

When online predictions are not required, the predictive accuracy of LSTMs can be improved with the addition of a backwards analysing layer, resulting in a bidirectional LSTM. Jagannatha and Yu (2016) compared bidirectional LSTMs, bidirectional gated recurrent units (GRUs), and conditional random fields (CRFs) in the extraction of medical labels from EHR notes. The LSTMs and GRUs had similar performance, and the 1% lead of the GRU performance was attributed to the combination of their simplicity and the small amount of training data. It is speculated that the LSTMs would yield superior performance on larger datasets. Both RNNs outperformed the CRFs. The machine learning task was to classify a sentence or document of medical notes into one of 9 classes, and the best performance obtained was an F-score of 0.8031 with the GRUs.

3 Methods

The semi-supervised approach consists of an autoencoder (Ammar et al., 2014) for representation inference, a dimensionality reduction step by means of the t-distributed Stochastic Neighbour Em-

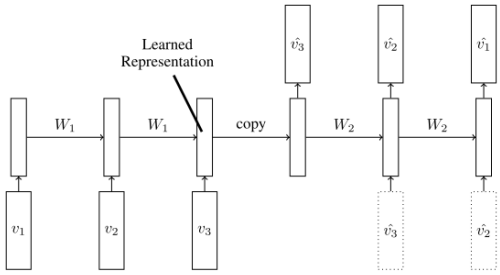


Figure 1: LSTM Autoencoder architecture. Here v denotes the input, and the subscripts indicate different time steps. The hidden unit weights for the encoder and decoder are represented by W_1 and W_2 respectively. The representation is passed to the decoder at the last time step, after which the decoder attempts to reconstruct the inputs $\hat{v}$. The dotted boxes depict the optional conditional input that can be used in such a model.(Adapted from: Srivastava et al. (2015))

bedding (t-SNE) method (van der Maaten and Hinton, 2008), and variational Bayesian estimation of Gaussian Mixture Models (GMMs) for clustering and relabelling. The efficacy of relabelling the data is determined by means of a long short-term memory (LSTM) recurrent neural network (RNN) classifier.

3.1 Bidirectional LSTM autoencoder

The goal of NN autoencoders has been to learn feature representations that improve generalization in supervised learning problems: (Ammar et al., 2014; Collobert and Weston, 2008; Socher et al., 2010; Vincent et al., 2008). The autoencoder attempts to learn a function $h_{W,b}(x) \approx (x)$, where W and b are the weights and bias terms of the NN, and x is the input data. Deep learning techniques that are modified to better handle time series data yield better results than deep learning techniques treating the input as static data (Längkvist et al., 2014). Thus an LSTM autoencoder is employed to learn representations of the temporal data. The LSTM autoencoder consists of 2 RNNs, the encoder LSTM and the decoder LSTM. The input to the model is a sequence of vectors. After the encoder has read the last input element, the output state is copied to the decoder, which then outputs an estimation of the target sequence.

More specifically we implement a bidirectional LSTM autoencoder with the same architecture as described in (Graves et al., 2013). This results in one LSTM encoding the sequence from the start to the end, and a different LSTM encoding the sequence from the end to the start, as illustrated in Figure 2. This results in 2 hidden representations, one for each LSTM. Intuitively, this allows the temporal model to generate a more accurate representation of the signals. RNN autoencoders can be conditional by using the output of a previous time step as the input to the current time step during decoding. Our implementation is unconditional because the goal is to generate the most informative hidden representation, and reducing the aid provided to the decoder during the end-to-end training results in a more encapsulating latent representation.

In general, autoencoders have a decreasing number of hidden units up to the pinch point (final hidden layer of the encoder) in the model. Medical time series are low-dimensional, rendering a pinching autoencoder impractical. It was assumed that due to the temporal nature of the data, an increasing number of hidden units in the LSTM autoencoder could learn effective latent representations. Even if the number of hidden units is larger than the input dimensions, the autoencoder can learn useful representations if proper constraints are imposed on the network (Bengio et al., 2007). Our implementation made use of 15 hidden units in a single layer bidirectional LSTM for the autoencoder. This results in 30 hidden units in both the encoder and decoder.

In order to prevent the model from learning a trivial identity mapping, we regularise the model by implementing a denoising-autoencoder. The type of noise added to the input data depends on the dataset because, in addition to regularising the model, it determines the type of noise removed from new data samples processed by the autoencoder. The objective function J used for training the

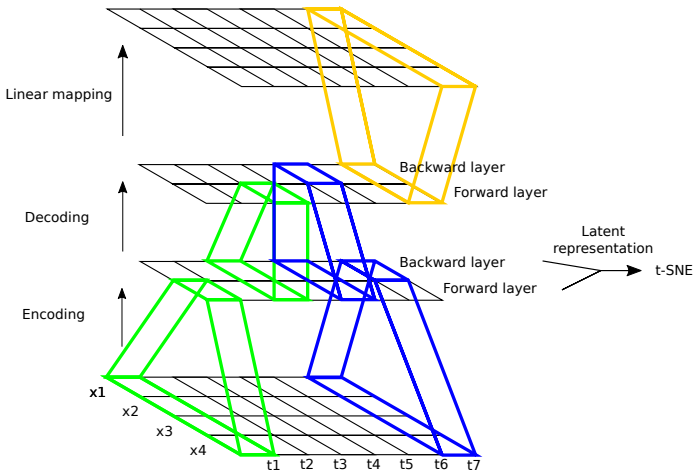


Figure 2: Bidirectional LSTM autoencoder architecture implemented in this study (number of elements in each layer are arbitrary examples). One LSTM encodes a hidden representation from the input data (bottom layer) in a forward fashion. Another LSTM encodes a second hidden representation in a backwards fashion. The *t*'s denote the time steps of the multi-variate time series x. The decoded backwards and forwards layers are linearly mapped to reconstruct a matrix with the same dimensions as the input matrix.

autoencoder is the squared-error defined as

$$J(W, b|x, \hat{x}) = \frac{1}{2}||h_{W,b}(x) - \hat{x}||^2 \tag{1}$$

with $\hat{x}$ being the reconstructed inputs. The model is trained using the Adam optimisation technique (Kingma and Ba, 2014) with a learning rate of 0.001 and a convergence criteria of a cost decrease less than 10^{-4} over 30 iterations. Hereafter the dataset is encoded and the dimensionality of the resulting latent representation is reduced using the t-Distributed Stochastic Neighbour Embedding discussed in the following section.

3.2 t-Distributed Stochastic Neighbour Embedding

The t-Distributed Stochastic Neighbour Embedding (t-SNE) method developed by van der Maaten and Hinton (2008) maps high-dimensional data into a low-dimensional (typically 2D or 3D) manifold and groups similar samples. This is achieved by minimising the divergence between a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding. Conventional methods for minimising the divergence scale quadratically with the number of objects, which limits the applicability to a few thousand data points. To mitigate the computational constraint, the Barnes-Hut approximation (Van Der Maaten, 2014) is implemented.

t-SNE requires specification of the perplexity parameter, which loosely guides the balance of attention between global and local aspects of the data. In a sense, the parameter is a guess of the number of close neighbours each point has (Wattenberg et al., 2016). Typical values of perplexity are between 5 and 50, and empirically 50 was found to be the best setting for our datasets. A weakness of t-SNE is that it is sensitive to data with high intrinsic dimensionality. A mitigation for this shortfall is to perform t-SNE on the nonlinear representation of the data produced by an autoencoder van der Maaten and Hinton (2008).

3.3 Variational Bayesian estimation of a Gaussian mixture

The main difficulty in clustering samples is knowing a priori how many clusters there are and what distribution each cluster has. To cluster the 2D representations in a non-parametric fashion, variational Bayesian estimation of Gaussian Mixture Models (GMMs) is used. A Dirichlet Process (DP) prior is used for the weights distribution. The DP inference algorithm is approximated and uses a truncated

distribution with a fixed number of components, called the Stick-breaking representation (Bishop, 2006). The number of components used almost always depends on the data, which is ideal for this application. The DP is a prior probability distribution on clusterings with an infinite unbounded number of partitions.

In this implementation, the upper bound hyperparameter on the number of components is specified as 1000, which is much larger than the number of different clusters expected in the datasets and chosen to simulate an infinite prior on the number of clusters. The weight concentration hyperparameter is chosen as 1, where a lower value leads to fewer clusters and vice versa.

3.4 Relabelling and classification

All the samples subjected to the clustering are labelled with the final outcome of the patient. The GMM inferred clusters (Section 3.3) are labelled according to the mode label of the samples within each cluster. Samples are then relabelled according to the label of their respective clusters. It is assumed that during the relabelled segment of a patient's stay, they experienced physiological changes indicative of the state represented by the cluster. For example, a patient that had a healthy final outcome might have had multiple segments in their time series that represented deterioration.

An LSTM classifier, with the same structure as that in (Graves, 2013), is then trained and tested separately on both the original dataset and the relabelled dataset. The LSTM has 128 units in one hidden layer, is regularised using 50% dropout, and trained using the Adam optimisation technique (Kingma and Ba, 2014) for 3000 epochs with a minibatch size of 512 samples. The datasets are split into a ratio of 50:10:40 for the training, validation, and testing sets. The best model from all the training epochs was determined by means of the validation set.

4 Data

The efficacy of the technique developed was evaluated with 2 datasets. The first dataset contained the first 48 hours of vital signs for 3 neonatal intensive care unit (NICU) patients. The signals used for analysis were electrocardiogram (ECG), blood pressure, and oxygen saturation. The data were segmented into samples with a length of 200 time steps at 60 Hz, resulting in a total of 134,812 samples from 3 different classes: normal, dying, and intraventricular haemorrhage.

The second dataset contained 48 half-hour excerpts of ECG recordings of 47 patients from the MIT-BIH arrhythmia database (Moody et al., 2001; Goldberger et al., 2000). Single heart beats were extracted using the Pan-Tompkins algorithm (Pan and Tompkins, 1985) which has a reported accuracy of 99.3% on this dataset. The result is 106,848 heart beat samples of 216 time steps at 360 Hz. The five heart beat classes selected from the database were: normal beat, right bundle branch block beat, left bundle branch block beat, paced beat, and premature ventricular fibrillation. Both datasets were normalised to have a zero mean and a range of [-1,1].

5 Results

Figure 3 illustrates that the autoencoder does not learn a trivial identity mapping of the data, and it is able to retain the important features of the original signal. Gaussian noise with a zero mean and 0.1 variance was added to 30% of the ECG data because the most likely type of noise inherent to ECG data was assumed to be Gaussian. For the NICU data, the most likely noise is missing values (replaced by zeros). Thus 30% of the NICU data was masked to zero as a means of adding noise to the data.

The dimensionality of the ECG encoded representations was reduced from $216 * 30$ to 2 values per sample using t-SNE. The 2D embedding after 10,000 iterations is illustrated in Figure 4a. t-SNE adapts its notion of distance to regional density variations in the dataset. Consequently, it naturally contracts sparse clusters and expands dense ones, evening out the clusters sizes. Thus the relative sizes of the clusters are not elucidated in t-SNE plots (Wattenberg et al., 2016). Moreover, the distance between clusters greatly depends on the chosen perplexity and is therefore not an informative metric. t-SNE has become popular because it's incredibly flexible and can often find structure were other dimensionality reduction techniques cannot. Unfortunately, the flexibility also makes the output difficult to interpret. The GMM clusters fitted to the 2D embedding by means of variational Bayes

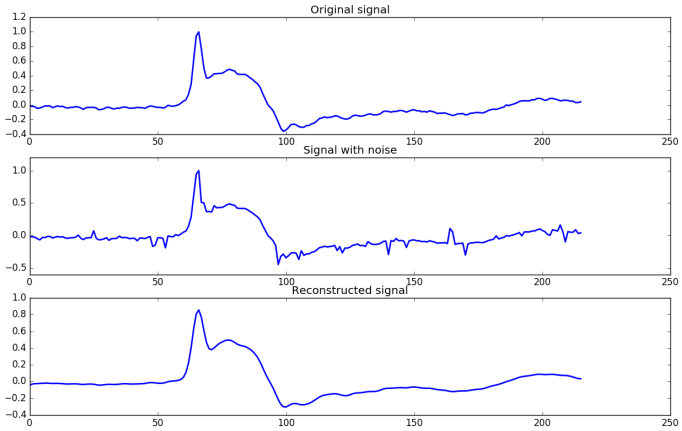
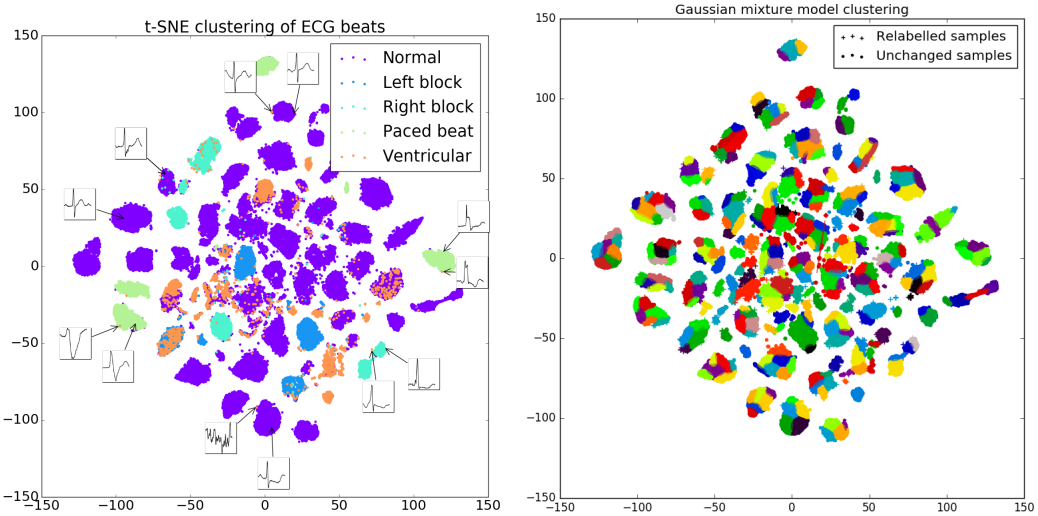


Figure 3: Example input and output of the autoencoder for ECG data. Gaussian noise with a zero mean and 0.1 variance was added (middle plot) to a randomly selected 30% of the original data (top plot). The autoencoder reconstruction (bottom plot) is not a precise replication of the original data, and the important features are retained.

are depicted in Figure 4b. The different colours represent different clusters. For relabelling, many small GMM clusters are ideal because this would lead to greater confidence in the label assigned to each cluster.

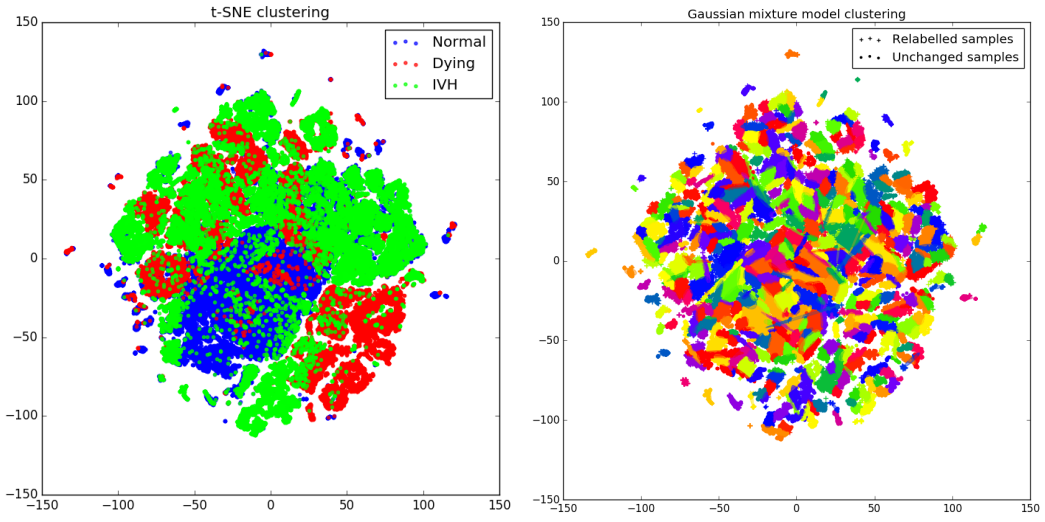


(a) (b)

Figure 4: Plots of the t-SNE 2D embedded output (4a) and the GMM relabelling (4b). Hand-picked signals are displayed in (4a), and the original classes are indicated by different colours. The hand-picked signals show how similar the outliers look to the rest of the cluster. In (4b) each color represents a different cluster with its own mode label.

For the NICU encoded representations, the dimensionality was reduced from $200 * 30$ to 2 using t-SNE. The result after 10,000 iterations is illustrated in Figure 5a. The GMM clusters fitted to the 2D embedding are shown in Figure 5b. Both figures elucidate the difficulty of clustering the representations, and that the t-SNE did not separate the samples adequately.

The classification accuracy found for the ECG dataset was 0.97 and 0.99 for the original and relabelled data respectively. For the NICU dataset, the model yielded an accuracy of 0.98 and 0.93 for the original and relabelled datasets respectively.



(a) (b)

Figure 5: Plots of the t-SNE 2D embedded output (5a) and the GMM relabelling (5b). In (5a) the original classes are indicated by different colours. In (5b) each color represents a different cluster with its own mode label.

6 Discussion

This study employed a denoising bidirectional LSTM autoencoder to infer latent representations of data. The representations were then mapped into a low-dimensional embedding using t-SNE before fitting GMM clusters to the 2D data using variational Bayesian estimation. The data was relabelled according to the inferred clusters. The classification accuracy was compared for the original and relabelled datasets using an LSTM classifier. It was found that the proposed approach for relabelling data to improve the accuracy and granularity of annotations resulted in an increase of 2% in classification performance for the ECG dataset and a decrease of 5% for the NICU dataset.

Intuitively with more accurate labels, more accurate classification is expected. However, the results of the NICU dataset indicate that what we believe to be more accurate labelling, does not improve classification accuracy. One reason could be that the t-SNE parameters need further refinement. Another possibility is that classification of the true medical states is harder than classifying the state of the whole signal (original labels). Larger datasets might mitigate this issue. Whether the proposed technique for labelling data improves classification accuracy remains inconclusive, but the semi-supervised approach has potential for further refinement and various other applications. Adversarial autoencoders (AAE) (Makhzani et al., 2015) allow unsupervised clustering and take a similar approach to the proposed technique, future work will investigate the efficacy of AAE in medicine.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., S. Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Ammar, W., Dyer, C., and Smith, N. A. (2014). Conditional Random Field Autoencoders for Unsupervised Structured Prediction. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3311–3319. Curran Associates, Inc.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., and others (2007). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19:153.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Choi, E., Bahadori, M. T., Schuetz, A., Stewart, W. F., Denny, J. C., Malin, B. A., and Sun, J. (2015). Doctor AI: Predicting Clinical Events via Recurrent Neural Networks. *arXiv:1511.05942 [cs]*.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., and Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet Components of a New Research Resource for Complex Physiologic Signals. *Circulation*, 101(23):e215–e220.

Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. *arXiv:1308.0850 [cs]*.

Graves, A., Jaitly, N., and r Mohamed, A. (2013). Hybrid speech recognition with Deep Bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507.

Jagannatha, A. N. and Yu, H. (2016). Bidirectional RNN for Medical Event Detection in Electronic Health Records. In *Proceedings of NAACL-HLT*, pages 473–482.

Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*.

Längkvist, M., Karlsson, L., and Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24.

Lipton, Z. C., Kale, D. C., Elkan, C., and Wetzell, R. (2015). Learning to Diagnose with LSTM Recurrent Neural Networks. *arXiv:1511.03677 [cs]*.

Lugovaya, T. S. (2005). Biometric human identification based on ECG. PhysioNet.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial Autoencoders. *arXiv:1511.05644 [cs]*.

Miotto, R., Li, L., Kidd, B. A., and Dudley, J. T. (2016). Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records. *Sci Rep*, 6.

Moody, G., Mark, R., and Goldberger, A. (May-June/2001). PhysioNet: A Web-based resource for the study of physiologic signals. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):70–75.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

Oresko, J. J., Jin, Z., Cheng, J., Huang, S., Sun, Y., Duschl, H., and Cheng, A. C. (2010). A Wearable Smartphone-Based Platform for Real-Time Cardiovascular Disease Detection Via Electrocardiogram Processing. *IEEE Transactions on Information Technology in Biomedicine*, 14(3):734–740.

Pan, J. and Tompkins, W. J. (1985). A Real-Time QRS Detection Algorithm. *IEEE Transactions on Biomedical Engineering*, BME-32(3):230–236.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2014). ImageNet Large Scale Visual Recognition Challenge. *arXiv:1409.0575 [cs]*.

Socher, R., Manning, C. D., and Ng, A. Y. (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pages 1–9.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. *CoRR, abs/1502.04681*, 2.

Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Van Der Maaten, L. (2014). Accelerating t-SNE using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245.

van der Maaten, L. and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.

Wattenberg, M., Viégas, F., and Johnson, I. (2016). How to Use t-SNE Effectively. `http://distill.pub/2016/misread-tsne/`.