# GWAS Statistics Imputation Using Stacked Bidirectional LSTMs

**Pol Rosello**
Dept. of Computer Science
Stanford University
prosello@stanford.edu

**Alex Tamkin**
Dept. of Computer Science
Stanford University
atamkin@stanford.com

**Alex Martinez**
Dept. of Computer Science
Stanford University
alexm711@stanford.edu

## Abstract

Genome-wide association (GWA) studies offer new avenues for medical insights. By aggregating summary association statistics across many individuals, they are largely free of the security and logistical concerns that burden the use of individual-level genome data, and provide a unique opportunity to uncover complex relationships between single nucleotide polymorphisms (SNPs) and phenotypic traits. However, this practice is complicated by different sets of SNPs being included in each study, leading to missing SNP statistics when analyzing studies in aggregate. To address this problem, we propose stacked bidirectional LSTM autoencoders with a custom loss function as a robust statistics imputation method. In our experiment involving the imputation of missing p-values across approximately one million SNPs and 11 traits, our method reduces the mean-squared logarithmic error on imputed p-values by 22.5% when compared to traditional statistical imputation techniques.

## 1 Introduction

A genome-wide association (GWA) study examines a set of single-nucleotide polymorphisms (SNPs) in a collection of individuals and correlates genetic variants with a specific trait. By aggregating many individuals' data, GWA studies avoid the privacy concerns and logistical hindrances of releasing individual genomes. GWA studies have produced vast databases of genetic variation featuring millions of individuals across hundreds of traits. Once a GWA study is compiled, practitioners can sequence a patient's genome to identify SNPs and improve their understanding of the patient's likelihood of developing conditions such as Alzheimer's or diabetes throughout their lifetime. This technique has already been used to reduce the risk of contracting life-threatening diseases such as breast cancer by performing preventative procedures on patients with the presence of risk-increasing SNPs. As the cost of sequencing techniques decreases, having a source of publicly-available, relatively noise-free GWA statistics for a comprehensive set of traits will become increasingly invaluable to medicine.

One complication of the use of GWA studies, however, is that not all studies analyze the same set of SNPs. Since GWA studies are often incomplete, the statistics for a specific SNP may be missing in a study. When aggregating findings from multiple GWA studies, solving this problem requires accurately imputing unobserved statistical data using linkage disequilibrium patterns found in more densely genotyped reference datasets. By leveraging interactions between nearby SNPs in the genome and inter-trait associations, a method that can accurately impute missing statistics would improve the robustness of genetics-based preventative medicine. In our work, we present several neural network-based architectures for imputing missing GWA statistics and show that our stacked bidirectional long short term memory (BLSTM) architecture consistently outperforms naive statistical imputation methods.

## 2 Related Work

In general, our problem can be thought of as a *sequence-to-sequence denoising problem*: we have an input sequence of values with some values missing and want to output a sequence with the imputed missing values. Recent work using multi-layer LSTMs to map sequences to sequences demonstrates their ability to better capture long-term dependencies across input data relative to traditional recurrent neural network units [1]. Regarding the denoising aspect of our problem, [2] introduces denoising autoencoders as a technique to codify meaningful, robust representations of inputs. Furthermore, [3] demonstrates the efficacy of stacked denoising autoencoders in learning useful, noise resistant representations of inputs, relative to alternative unsupervised learning models such as deep belief networks.

Recent work in the domain of genome statistics imputation evaluates a row averaging method, a weighted $k$-nearest neighbors method, and a singular-value decomposition-based method [4] as applied to DNA microarrays. Researchers found that the $k$-nearest neighbors and SVD-based methods surpassed the row-averaging method, with the former more robust and sensitive. To the best of our knowledge, the area of GWA statistics imputation is largely unexplored.

## 3 Dataset and Preprocessing

From a list of publicly-available summary association statistics featured in [5], we chose 11 studies for our work, shown in Table 1. Each study reports p-values for a specific trait for about a million SNPs each. The 11 studies we use mention 2,866,924 unique SNPs in total. 22.1% of the p-values for these SNPs are missing in the full data. To train our models, we need complete statistics to use as target labels. To that end, we use the subset of SNPs that have statistics for all 11 traits. 925,285 SNPs met this criterion.

We represent each SNP as an 11-dimensional vector of p-values and order SNPs by their position in the genome. When ordered this way, our dataset can be interpreted as a long sequence of SNP vectors. For most architectures, modeling this sequence at once is computationally unfeasible. We therefore split the sequence into shorter subsequences of fixed length with an overlapping stride between consecutive subsequences. We optimize these values on a per-model basis. We ensure that subsequences never cross chromosome boundaries and split the subsequences for each chromosome into training, validation, and test sets. Although subsequences may overlap within a set, we ensured that no SNPs were present in both the training and test sets.

Modeling p-values accurately becomes more important as they approach zero. P-values represent the probability that an observation (i.e. a particular SNP-trait pairing) occurs given the null hypothesis that they are uncorrelated. Therefore, we are not as interested in the difference between relatively large p-values (suggesting no significant correlation) as we are in the difference between small p-values. For example, the difference between a p-value of 0.1 and 0.0001 is much more important to capture than the difference between a p-value of 0.8 and 0.7, at which point both values fail to represent a strong correlation between a trait and the presence or absence of a SNP. We therefore modulate the raw p-values in the dataset by using log p-values instead. We add 1 to the p-values before taking the logarithm to simplify the evaluation of our loss function (see Section 4.1) and to prevent numerical stability for p-values near zero. We also mean-normalize the inputs, which helps with training and allows us to simulate missing statistics in the input by randomly setting p-values to zero.

## 4 Methodology

We explore two main architectures in our work and compare their performance to more naive baselines. Hyperparameters of each architecture, such as the subsequence length and the hidden layer sizes, were optimized with respect to the validation set. Section 5 includes a discussion on the results of our hyperparameter and architecture exploration.

Table 1: Publicly-available GWA studies used in our work. $N$ refers to the number of patients in each study.

| Trait | $N$ | Reference | URL |
|---|---|---|---|
| HbA1C | 46,368 | Soranzo et al. 2010 Diabetes | magicinvestigators.org/downloads/ |
| BMI | 122,033 | Speliotes et al. 2010 Nat Genet | broadinstitute.org/collaboration/giant/index.php/GIANT_consortium_data_files |
| Coronary artery disease | 77,210 | Schunkert et al. 2011 Nat Genet | cardiogramplusc4d.org/ |
| Cholesterol levels | 94,461 | Teslovich et al. 2010 Nature | broadinstitute.org/mpg/pubs/lipids2010/ |
| Crohn's disease | 20,883 | Jostins et al. 2012 Nature | ibdgenetics.org/downloads.html |
| Fasting glucose levels | 58,074 | Manning et al. 2012 Nat Genet | magicinvestigators.org/downloads/ |
| Fasting insulin levels | 51,750 | Manning et al. 2012 Nat Genet | magicinvestigators.org/downloads/ |
| Height | 131,547 | Lango Allen et al. 2010 Nature | broadinstitute.org/collaboration/giant/index.php/GIANT_consortium_data_files |
| Insulin sensitivity index | 16,753 | Walford GA et al. 2016 Diabetes | magicinvestigators.org/downloads/ |
| Triglyceride levels | 94,461 | Teslovich et al. 2010 Nature | broadinstitute.org/mpg/pubs/lipids2010/ |
| Ulcerative colitis | 27,432 | Jostins et al. 2012 Nature | ibdgenetics.org/downloads.html |

### 4.1 Objective

The framework for our problem is summarized in Figure 1. We corrupt the input SNP vectors by randomly setting some statistics to zero. Our models then attempt to impute the missing values. We evaluate all our models using the mean squared logarithmic error (MSLE) between the original, non-corrupted values and the imputed values. Given a vector of true statistics $s$ and a reconstructed vector $s^*$, the MSLE is defined as:

$$\text{MSLE}(s, s^*) = \frac{1}{|M(s)|} \sum_{i \in M(s)} (\log(s_i + 1) - \log(s_i^* + 1))^2$$

where $M(s)$ denotes the set of indices of $s$ where statistics were censored in the input to the model. When our dataset's p-values are preprocessed by taking their logarithm, the MSLE reduces to the mean-squared error (MSE). Computing the loss exclusively on the imputed values is equivalent to the loss function of the "emphasized denoising autoencoder" in [3] with $\alpha = 1$ and $\beta = 0$. Unlike in traditional denoising autoencoders, we do not care about the ability of our model to reconstruct values that are already present in the input: if the p-values are present in the input, there is no need to impute them.

In our neural network architectures, we directly optimize our objective by using the MSLE on the imputed values as the loss function. We use Adam [6] to perform gradient descent with per-parameter adaptive learning rates, using the default values of $\alpha = 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. We train our models on an NVIDIA Tesla K80 GPU and use the largest batch size for each architecture that can fit in memory. All our models were implemented using Keras [7] and TensorFlow [8].

### 4.2 Baseline Models

#### 4.2.1 Statistical baselines

To contextualize our results, we use several statistical methods to impute missing statistics. The simplest is a per-trait "global" average. In this method, we compute an average across all available p-values for each trait and then impute all missing values using these averages. We also use more sophisticated statistical methods to impute missing p-values. These include matrix completion by

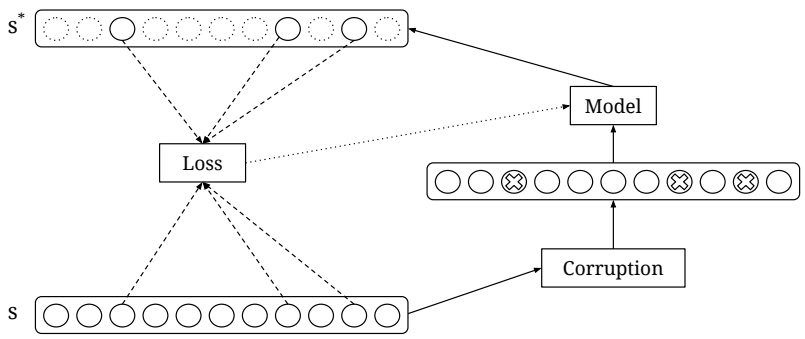


Figure 1: Imputation framework for all our models. The loss is computed only with respect to the corrupted inputs.

iterative soft thresholding of SVD decompositions [10], multiple imputation by chained equations [11], and matrix completion by iterative low-rank SVD decomposition [4]. We also attempted to use $k$-nearest neighbor imputation, but finding the Euclidean distance for every SNP pair was not computationally feasible. The issue with all of the methods we have mentioned so far is that they do not take the relative ordering of the SNP vectors into account. Rather, they assume they are independent samples drawn from the same distribution. However, the values of nearby SNPs in the genome for the same trait are likely to contain information that is useful in imputing the missing values.

To remedy this issue, we also use a "local" average variant to impute p-values. This method imputes each missing p-value by searching its left and right SNP vector neighbors for the nearest available p-value in each direction in the genome, and then uses the average of the two as the imputed value. In our experiments, all sophisticated statistical methods we tried performed worse than our local average variant, so we restrict our results to only the global and local averages.

#### 4.2.2 SNP-wise denoising autoencoder baseline

This baseline attempts to reconstruct censored individual SNP vectors one 11-dimensional vector at a time using a fully-connected network with a single hidden layer, in the style of denoising autoencoders [2]. The idea of this architecture is to exploit inter-trait dependencies to impute missing statistics. For example, if a SNP is correlated with low insulin levels, it may also be correlated with high levels of glucose, so if either p-value is missing, the model might be able to impute it relatively accurately. In this neural network architecture, as well as in the following two architectures, we choose the ReLU activation function for all layers except the output layer, where we use a linear activation.

### 4.3 Improved Architectures

#### 4.3.1 Architecture 1: Subsequence denoising autoencoder

This baseline attempts to impute summary statistics using a network similar to the one in Section 4.2.2, except that it takes in flattened sequences of adjacent SNP vectors and tries to reconstruct the censored values for the entire sequence. By treating the data as a sequence, the intuition is that the network may learn how to impute a summary statistic from the statistics of neighboring SNPs in the genome. Unlike all previous models, this architecture should be able to exploit inter-SNP dependencies for neighboring SNPs as well as inter-trait dependencies.

### 4.4 Architecture 2: Stacked bidirectional LSTM

A large drawback to the fully-connected sequence architecture is that long-distance interactions cannot be easily exploited. As the sequence length of the inputs is increased, the number of parameters in the architecture grows exponentially. To remedy this, we choose to treat our task as a sequence-to-

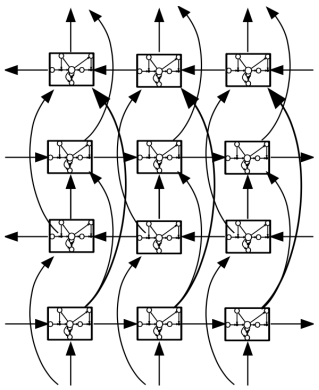

Figure 2: Structure of our stacked bidirectional LSTM architecture with two bidirectional layers. Illustration from [9].

sequence problem and turn our attention to recurrent architectures, where the input sequences can be of arbitrary length.

For our task, we consider stacked bidirectional LSTMs. In many domains, LSTMs have been more successful modeling sequences than plain recurrent neural networks. Using two LSTM units at each layer (one for each direction) allows the architecture to consider summary statistics from both directions in the genome. Additionally, stacking layers of bidirectional units enables our model to learn more complex patterns in the input. This architecture is summarized in Figure 2. We include a fully-connected layer with a linear activation just before the output to reduce the hidden dimensions to the correct size and to further enhance the expressiveness of our model.

Unlike sequence-to-sequence models that encode sequential inputs into a fixed-length representation before decoding that representation into a new sequence (as in translation models), this model outputs a vector for every position in the input sequence. This is preferred in our case because the input and target sequences are synchronized and equal in length, and we are not attempting to find a fixed-length encoding of the input sequence.

## 5 Results

### 5.1 Architecture exploration

Table 2 shows a subset of the results of our hyperparameter optimizations for each architecture. In all models with a sequence length greater than 1, we used a stride length of half of the sequence length. Overall, the stacked LSTM architecture with two bidirectional layers and a 100-dimensional hidden unit per LSTM outperformed all other models. In general, increasing the size of the hidden dimension improved the performance of the neural network-based models when holding sequence length constant, but we tended to experience diminishing returns.

Preprocessing the input by taking the logarithm of the raw p-values and mean-normalizing the result significantly improved the performance of all our models. As an example, the loss on the final BLSTM architecture with and without preprocessing the input is shown in Figure 3. When the input is preprocessed, the loss decays much faster from the very beginning. Without preprocessing, the models never perform quite as well even after many epochs of training.

We show the effect of increasing the length of the subsequences for the subsequence autoencoder and the BLSTM models in Figure 4. While longer sequence lengths improved the performance of the BLSTM without having to modify the architecture, the subsequence autoencoder model was much harder to train on longer sequences. This was despite our efforts to improve the expressiveness of the autoencoder by scaling up the size of the hidden dimension as the length of the input sequence was increased. Unlike the subsequence autoencoder, the stacked BLSTM was able to capture long-distance relationships in the input to provide more accurate imputations. Furthermore, the number of parameters in the subsequence autoencoder grows exponentially with the length of the input sequence, whereas in the BLSTM model, the number of parameters stays fixed. Even if we were able

Table 2: Architecture exploration results on validation set using 0.2 as the corruption probability. Models marked with * are the best-performing variants of each architecture, which we include in the rest of our results.

| Model | Seq length | Hidden dimension(s) | Trained parameters | LMSE |
|---|---|---|---|---|
| Global average* | - | - | 11 | 0.0395 |
| Local average* | - | - | - | 0.0388 |
| SNP-wise Autoencoder | 1 | 20 | 264 | 0.0402 |
| SNP-wise Autoencoder* | 1 | 500 | 2311 | 0.0401 |
| Subsequence Autoencoder* | 20 | 220 | 606,100 | 0.0309 |
| Subsequence Autoencoder | 500 | 5500 | 60,511,000 | 0.0395 |
| Subsequence Autoencoder | 500 | 5500, 5500 | 90,766,500 | 0.0397 |
| BLSTM | 500 | $50 \times 2$ | 25,911 | 0.0306 |
| BLSTM | 500 | $100 \times 2$ | 91,811 | 0.0301 |
| BLSTM | 500 | $50 \times 2$, $50 \times 2$ | 86,311 | 0.0301 |
| BLSTM* | 500 | $100 \times 2$, $100 \times 2$ | 332,611 | **0.0300** |

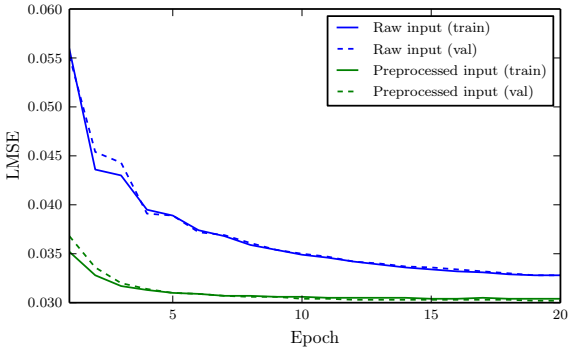


Figure 3: Loss on the final BLSTM architecture at the end of each epoch, illustrating the effects of preprocessing the input. The corruption ratio used was 0.2.

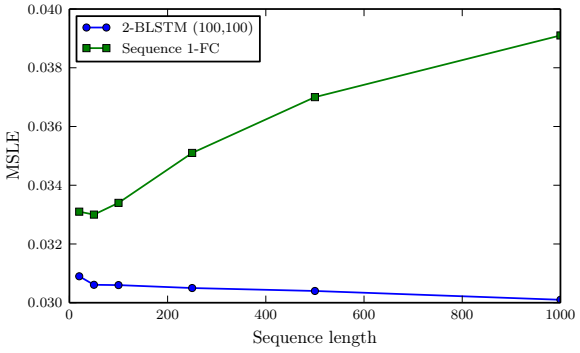


Figure 4: Loss of the BLSTM and subsequence autoencoder on the validation set after 20 epochs of training for different input sequence lengths. The corruption ratio used was 0.2. The hidden dimension of the subsequence autoencoder was set to be equal to the number values in the input (i.e. 11 times the number of SNP vectors in each input sequence).

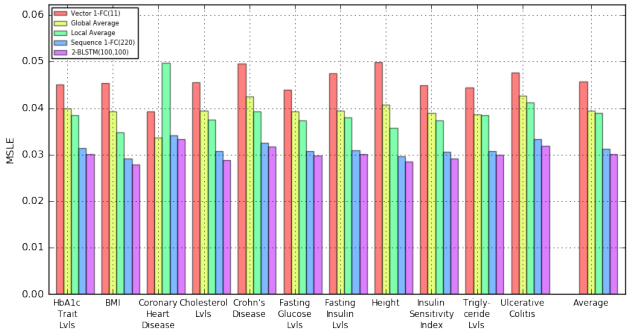


Figure 5: Loss per trait of each of our models on the test set for a 0.2 corruption ratio.

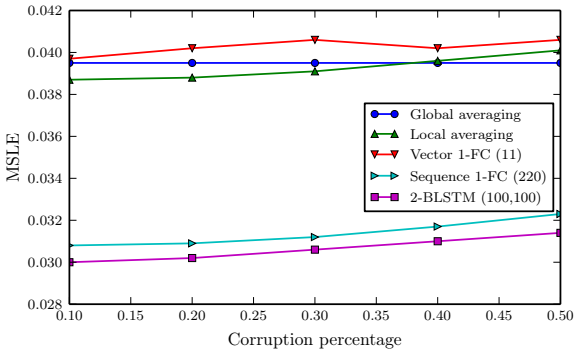


Figure 6: Loss of various architectures on the test set for different corruption percentages in the input.

to successfully train the subsequence autoencoder on longer sequences, it would only be able to use information from the nearest 1̃,000 SNPs to impute missing statistics before it would be impossible to train on a modern GPU.

In the next section, we restrict our discussion to the best-performing models for each architecture. Although the BLSTM model with 1,000 SNPs per sequence performed better, we use the model with 500 SNPs per sequence because it is slightly faster to train and performs comparably well. We train the stacked BLSTM model for 60 epochs, and all other trainable models for 100 epochs.

### 5.2 Test set evaluation

The final performance of our architectures on the test set is summarized in Figure 5. Both of our architectures beat the baseline models on every trait, and the BLSTM consistently outperforms the subsequence autoencoder model, although not by a large margin. The best results for both of our architectures are obtained on the BMI GWA study. The coronary artery disease GWA study is the hardest study for our architectures to model. We suspect this study is particularly noisy, since the local average baseline uncharacteristically performs much worse than the global average baseline. The SNP-wise autoencoder does not perform well overall, likely because inter-trait dependencies in the data are not enough to compute accurate statistical estimates. It may perform better when there is a greater number of GWA studies in the dataset and when the traits they explore are phenotypically related. On average, the BLSTM model reduces the MSLE by 22.5% compared to the best statistical model we found (the local average baseline) and by 25% compared to the global average baseline.

We illustrate the performance of our architectures on the test set for different corruption percentages in the input in Figure 6. As a reminder, the corruption ratio on the original dataset we collected (before restricting it to the subset of SNPs with full statistics) was 0.22. The mean-based baselines do not perform much worse as the percentage of withheld statistics increases. This is expected because the input values were corrupted uniformly at random, so the statistical properties of the data do not

change much. All neural network-based models, however, experience a greater loss as the corruption percentage is increased because there are fewer p-values in the input from which to draw information.

## 6 Conclusion and Future Work

We present two novel architectures for GWAS statistics imputation that consider interactions between neighboring SNPs in the genome. They provide better performance than both off-the-shelf matrix-naive algorithms, as well as global-averaging, local-averaging, and basic autoencoder baselines, achieving a 22.5% decrease in the mean-squared logarithmic error.

Moreover, one of these models, the stacked BLSTM, is computationally feasible to train as the size of the input grows, while the simpler subsequence autoencoder uses an exponential number of parameters under these conditions, takes longer to train, and often fails to converge to a better-than-baseline solution. The stacked BLSTM consistently outperforms subsequence autoencoders for every trait in our dataset, and has a fixed number of parameters that does not grow as the length of the input sequence is increased. Overall, our experiments show that treating GWA statistics imputation as a sequence-to-sequence problem is a powerful approach to exploit inter-SNP dependencies.

These findings provide several promising avenues for future work. For instance, there is a limit to the amount of information a model can glean from only summary statistics. To improve accuracy, future models could incorporate information about the nucleotides surrounding a given SNP, in addition to the distance between consecutive SNPs. Furthermore, exploring convolutional architectures, which may better model the effects of neighboring SNPs, could ultimately improve our model's accuracy and decrease the required training time through the use of shared filters.

In addition, our experiments only explore the case of imputing summary statistics deleted uniformly at random from our dataset. In practice, long consecutive sequences of summary statistics are missing for a specific trait. Future work should assess the accuracy of imputation models against these more common configurations of missing data and develop new architectures to impute them effectively.


### Acknowledgments

The authors would like to thank Ronjon Nag for discussions in the early stages of this work, James Zou for his guidance, and Microsoft Azure for providing us with much-needed computational resources.


## References


[1] Sutskever, I. Vinyals, O. & Le. Q. V. Sequence to sequence learning with neural networks. In Proc. Advances in Neural Information Processing Systems 27 3104–3112 (2014).

[2] P. Vincent, H. Larochelle, Y. Bengio and P. A. Manzagol. *Extracting and Composing Robust Features with Denoising Autoencoders.* Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08), 1096–1103, ACM, 2008.

[3] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. *Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion.* Journal of Machine Learning Research 11 (2010) 3371–3408.

[4] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, and R. B. Altman (2001). *Missing value estimation methods for DNA microarrays. Bioinformatics*, 17(6), 520–525.

[5] B. Pasaniuc and A. L. Price. *Dissecting the Genetics of Complex Traits Using Summary Association Statistics.* Preprint at *bioRxiv* http://dx.doi.org/10.1101/072934 (2016).

[6] D. Kingma, and J. Ba (2014). *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980.

[7] F. Chollet, *Keras* (2015). `https://github.com/fchollet/keras`.

[8] M. Abadi et al., *TensorFlow: Large-scale machine learning on heterogeneous systems* (2015). `http://tensorflow.org/`.

[9] A. Graves, N. Jaitly, and A. R. Mohamed, (Dec 2013). *Hybrid speech recognition with deep bidirectional LSTM*. In Automatic Speech Recognition and Understanding, 2013 IEEE Workshop 273–278.

[10] R. Mazumder, T. Hastie, and R. Tibshirani (2010). *Spectral Regularization Algorithms for Learning Large Incomplete Matrices*. Journal of Machine Learning Research, 11(Aug), 2287–2322.

[11] M. J. Azur, E. A. Stuart, C. Frangakis, and P. .J. Leaf (2011). *Multiple Imputation by Chained Equations: What is it and how does it work?* International Journal of Methods in Psychiatric Research, 20(1), 40—49.