Neural Networks Approaches to DNA Sequence Denoising

Project Report
HealthCare

Christine Tataru, Abhishek Roushan, Clara McCreery
{ctataru5,aroushan,mccreery}@stanford.edu

***Abstract*—Noise introduced to DNA sequences from technological sequencing error is a significant problem in the field of microbiome research. Some techniques already exist for sequence denoising, but they don't scale well. We investigate the potential for neural networks to convert noisy DNA sequences to their denoised counterparts and present the results of two architectures: a convolutional neural network (CNN) and a long-short-term memory cell (LSTM) network. Shallow versions of both network types effectively reduced the noise by more than a factor of 3 after only a few epochs of training.**

I. Introduction

RECENT scientific discoveries suggest that the human gut microbiome has a significant effect on human health, even beyond the gastrointestinal tract [1]. The gut microbiome may be implicated in obesity, diabetes, allergies, anxiety, depression, schizophrenia, autism and more [2]. A whole new frontier of medicine thus depends upon our ability to reliably sequence and analyze the gut microbiome. However, high-throughput DNA sequence data is error-prone, and the DNA sequences of bacterial species in the gut sometimes differ by as little as one nucleotide in the analyzed region. Therefore it is hard to distinguish between noise introduced by sequencing error and natural variation due to evolution.

Current denoising protocols are predominantly alignment-based, and thus struggle to resolve nucleotide differences without clear reference sequences, of which there are an insufficient number. We propose a neural network approach to resolve noisy microbiome DNA sequences without reliance on reference sequences. The assumption that we are making (which experts in the field support) is that certain types of nucleotide variation are more characteristic of sequencing error, while others are more characteristic of evolution. For example, G to A substitution is known as a common error made by sequencing methods [3] and certain patterns such as CGGT increase the likelihood of sequencing errors as well [4].

To observe the abundance of bacterial species in the human gut, scientists sequence the 16S rRNA gene, which evolves at an estimated rate of 1% per 50 million years and can function as a taxonomic tag [5]. A few nucleotide differences between 16S sequences can signify thousands of years of evolutionary change, making it imperative to deconvolve evolutionary variation from sequencing errors. We aim to build a neural network capable of distinguishing between biological and technological sequence variation, finding the most likely original DNA sequences from which a set of noisy sequences were derived.

II. Related Work

In the past, there have been a few algorithms proposed to solve the denoising problem. An older technique like image stacking was used for removal of white noise [6]. Bioinformatics developers have also developed "consensus" methods [7], which use pairwise alignment for insertion and deletion handling. Due to the poor scaling of such algorithms we choose to apply deep learning to generate non-noisy sequences. When choosing a network architecture, we treat this problem like image denoising on one hand, which uses deep CNNs [8] and like other sequential data on the other hand, that leverages Recurrent Networks (RNN's).

III. DataSet Overview

We are using data specifically created for benchmarking methods to improve microbial profiling. A community of 57 bacteria of about equal abundances, is sequenced twice, once to the standard 20X coverage and once to a financially unsustainable but more accurate 500X coverage. Coverage means each segment of DNA is sequenced on average 20 or 500 times and can act as a proxy for confidence in nucleotide calls. This provides us with data for a supervised learning problem where in input is a sequence from the 20X batch and the output should be it's corresponding 500X sequence. Data came from the European Nucleotide Archive [9].

The column scheme is organized as follows: query identifier, bacterium species, raw noisy sequence (input), denoised sequence (output), e-value. The lower the e-value, the closer the association between the noisy and the denoised sequence. These data consist of approximately 500,000 sequences, out of which approximately 12% have at least one error. The noisy and denoised sequences have been aligned such that insertions and deletions in the noisy sequence do not create off-by-one errors for the remainder of the sequence. In order to align them, a "-" has been inserted in either the noisy or the denoised sequence. An example noisy sequence is shown in Figure 1.

A. Data Preparation

The objective of sequence to sequence learning can be accomplished efficiently by character level modeling of the input data [10]. The nucleotides are not believed to have

TABLE: Example Dataset Sequence

Noisy Seq ID	Reference Taxa ID	Noisy Sequence	Reference Sequence	e-value
ERR777695.8	Desulfovibrio_piger_ATCC_29098	ACGGAGGGTGCAAGCGTTAATCGGAATCACTG GGCGTAAAGCGCACGTAGGCTGTTTGTAAGT CAGGGGTGAAAGCCCGCGGCTCAACCGCGGAA TTGCCTTTGATACTGC CGAGCTAGATCCGGG AGAGGGTAGTGGAATTCCAGGTGTAGGAGTGA AATCCGTAGAGATCTGGAGGAACATCAGTGGC GAAGGCGACTACCTGGACCGGTACTGACGCTG AGGTGCGAAAGCGTGGGGAGCAAACAG	ACGGAGGGTGCAAGCGTTAATCGGAATCACTGGGCG TAAAGCGCACGTAGGCTGTTATGTAAGTCAGGGGTG AAAGCCCACGGCTCAACCGTGGAACTGCCTTGATA CTGCACGA CTCGAATCCGGGAGAGGGTAGCGGAAT TCCAGGTGTAGGAGTGAAATCCGTAGATATCTGGAG GAACATCAGTGGCGAAGGCGACCACCTGGACCGGTA TTGACGCTGAGGTGCGAAAGCGTGGGGAGCAAACAG	5.64e-103

Figure 1: Noisy and Exact Sequence example

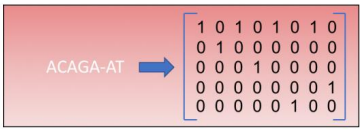


Figure 2: One-hot representation for DNA data

a hidden feature breakdown (which helps a great deal in embeddings like GloVe, Word2Vec). Thus a one-hot encoding representation of the input data makes most sense. The input to the models discussed in the following sections is a 5-dimensional one-hot vector of classes A, C, G, T, and - (see Figure 2).

IV. Methodology

As mentioned in the previous section, we treat our problem as the supervised learning of an output sequence from an input sequence. This constricted our model space search in the domain of sequence prediction. We therefore, propose two very different model architectures: a **Convolutional** model and a **Recurrent** model.

Both models are trained on a higher ratio of noisy samples than is naturally occurring in order to speed up training and steer the network away from predicting the input as output. Specifically, the training set contains 112,089 sequences, of which 55,906 are noisy (contain at least one incorrect nucleotide). The dev set contains 12,269 sequences of which 1,450 are noisy and the test set contains 12,107 sequences and 1,507 are noisy. Both networks are trained and tested on the exact same train/val/test sequences in order to maintain a fair comparison (see Figure 3 for distribution representation).

A. *Convolutional Neural Network Model*

The first architecture is a convolutional neural network (CNN) that treats the data like a noisy image, using the nucleotides on both sides of an error [11] to recognize and correct them (See Fig 4).

The current network is five layers deep and includes regularization at each layer. We use ReLU [20] except in the final layer, which uses softmax activation to force the entries of each one-hot vector to sum to 1. Kernel sizes are not square, which allows the network to rely on a greater number of nucleotides before and after the noise. For example, a kernel size of (5×7) was experimentally found to perform much better than a kernel size of (5×5) for the first layer. The network was trained for 12 epochs, though we speculate it might improve slightly with more training time.

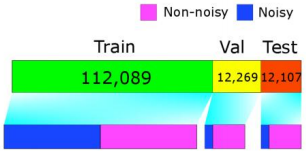


Figure 3: Data distribution

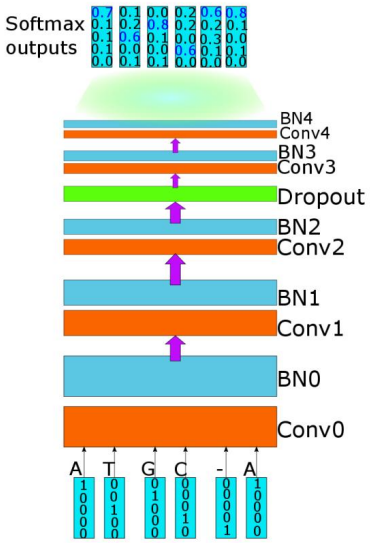


Figure 4: Conv Net model architecture

Table I: CNN Model Architecture

Attributes	Details
Number of layers	5
Activation	ReLU, (exc. softmax final layer)
Dropout	4^{th} layer dropout p=0.3
Convolution	2D "same" padding
Kernel type	Non-square
Optimizer	Adam
Loss	Mean squared error
Metric of success	Accuracy

B. *Recurrent Neural Net model: LSTM*

Whereas the convolutional model mimics image denoising, we hypothesized that an architecture modeling DNA sequences as a time series, such as those used in machine translation of language, may perform better.

We therefore develop several LSTM-based models [12] that encode and use the previous nucleotides in the sequence to predict the next nucleotide. In this model, we treat DNA as a natural language with vocabulary of five words: Adenine, Cytosine, Guanine, Thymine, and "-" for nucleotide deletion.

For each of our three architecture experiments, we use about 7,000 training examples, and decide based on the keras evaluation metric ($\hat{Y}$= predicted one-hot vectors, Y^*= true labels):

$$M = \frac{\hat{Y} - Y^*}{onehot\_size * numSeqs * max\_seqlength}$$

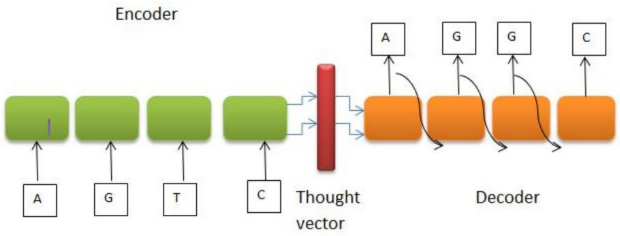


Figure 5: LSTM encoder-decoder architecture

whether to tune that architecture's hyperparameters. We then train the chosen architecture with chosen hyperparameters on the whole train set and present more intuitive evaluation on test set performance. The model descriptions are below:

*1) **Single layer Encoder-Decoder LSTM model**:* Inspired by the autoencoder model [13], we create a four hidden layer LSTM architecture following the encoder-decoder architecture (see Figure 5) where the information is passed via a thought vector.

Table II: Encoder decoder LSTM Model architecture

Attributes	Details
Number of layers	1
Latent dimension	100
Optimizer	Adam
Loss	Categorical cross-entropy
Prediction	Decoder Inference greedy sampling
Metric of success	Accuracy

The details of the LSTM model are provided in Table II. We use "Adam" to optimize the "categorical cross-entropy" loss between the network output one-hot encoded sequences and the expected denoised sequences. Because of our design choice of relatively few hidden layers with limited latent dimensions, the model requires very few(10s) of epochs to train. From our observation, the training model stagnates at $\boxed{75\%}$ training and validation accuracy. For experimentation purpose, we trained it on noisy examples only. Validation and training accuracy were very close to one another.

*2) **Neural Machine Translation model**:* Neural machine translation (NMT) has proven to be quite successful in a sequence to sequence translation and prediction. Recent advancements in NMT like Beam search decoding and Attention based models [14] have been great in correctly predicting the "next" word in the sequence.

We utilized some of the expertise of the NMT model and developed an architecture with character level tensorflow embedding. The dataset is preprocessed to be given input as a **space separated** sequence of A,C,G,T,- (Adenine, Cytosine, Guanine, Thymine, and no nucleotide respectively) forwarded to both the source and target predictions [15] (refer Fig 6). The design has a single embedding layer, two LSTM hidden layers and an output projection layer. For the purpose of experimentation, we trained the model on around 50,000 input sequences split into train/dev/test set as $\boxed{90}\boxed{5}\boxed{5}$. The model is trained for "Softmax Cross Entropy" loss with "Adam" optimizer. As the model is trying to learn the input sequences

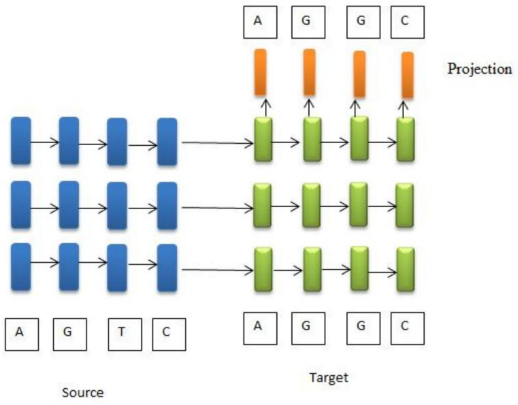


Figure 6: NMT architecture

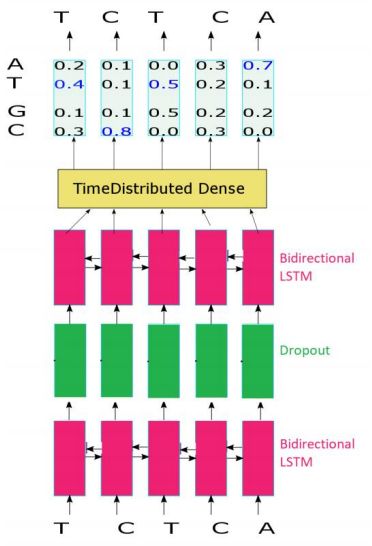


Figure 7: Many-to-many LSTM architecture

of variable length, we expect the predictions of NMT to be of variable length. We were able to achieve a Bleu score [17] around $\boxed{4.0}$, which suggests an area of improvement for future.

*3) **Many-to-many multi-layered LSTM Architecture**:* After observing the performance of the previous architectures, we thought to combine the best elements of our architectures so far, namely the CNN's ability to condense information from the entire sequence and the machine translation LSTM model's ability to more heavily weight local environment information. We created a many-to-many [16] model architecture with two bidirectional LSTM layers of hidden dimension 100 and a Dropout layer (p=0.5) between them for regularization (see Figure 7). Ultimately LSTM output is put through a TimeDistributed Dense layer with softmax activation to obtain a probability of each A-T-C-G nucleotide. The predicted sequence in the maximum probability nucleotide at each position. This is the model for which **we provide final results** because it outperformed the others.

V. Experiments and Results

Both the convolutional network and the recurrent network are able to learn to output the correct nucleotide rather than the input noisy one most of the time, often with very high confidence. See Figure 8 for an example of such an output.

```
Red: Should have stayed the same but didnt (bad)[87 98]
Green: Should have changed and did (good)[70 76 82 86]
Blue: Should have changed but didnt (ok)[74 95]
T ATCGAAAACTGCCGGTCGGAAGGAACACCTGTGGCGAAAGCGG
I ATCGAAAACTGCCGGTCTGGAGGAATACCGGTGGCGAAGGCGG
P ATCGAAAACTGCCGGTCCGAAGGAACACCTATGGCGAAGGCAG
```

Figure 8: Segment of text output color-coded by output type. T = Target sequence, I = Input sequence, P = Predicted sequence

After selecting our top performing models based on validation set accuracy, we test our models on the test data and report performance on a few metrics:

- Accuracy per sequence:

$$APS = \frac{\#correct\_nucleotides}{\#sequences}$$

(indicates number of correct nucleotides expected in a sequence). Vast majority of sequences are 251 ± 3 nucleotides

- Accuracy per base:

$$APB = \frac{\#correct\_nucleotides}{total\#nucleotides}$$

(probability of a given nucleotide being correct)

- Total set accuracy:

$$TSA = \frac{\#completely\_correct\_sequences}{total\_\#of\_sequences}$$

A. *Convolutional Model*

- **Train curves** We observe a plateaued behavior of average nucleotides changed per sequence as the training progresses. Figure 9 displays the pattern of both the mistakes introduced and mistakes corrected as the model trains, which corroborates the desired behavior of substituting/inserting nucleotides.
- **Accuracy of predictions**
  The current convolutional model has surpassed $\boxed{96.5\%}$ accuracy on the test set, when measuring accuracy as total set accuracy. As a point of reference, accuracy of the input test sequences (total set accuracy) was $\boxed{87.6\%}$, so we have eliminated about 72% of the noisy sequences completely.
- **Prediction patterns in the model**
  We visualize the results both as an overall accuracy and by separating out the different types of nucleotide errors (Fig 10). In these charts, "Baseline Predictions" illustrates the frequency of changes that a perfect network would make, and the other three charts show the frequency of changes that this network made. The convolutional network learns to correct the two most common errors (T to C and G to A) very well, but often fails to change C to T and A to G, which are also common mistakes. Furthermore, we see that on average it is changing about 0.602 nucleotides per sequence, with on average 0.595 of those being correct changes and 0.007 of those being bad changes (introduction of new noise to previously correct sequences). Because the good changes are two orders of magnitude greater than the bad, the data set overall becomes much less noisy.

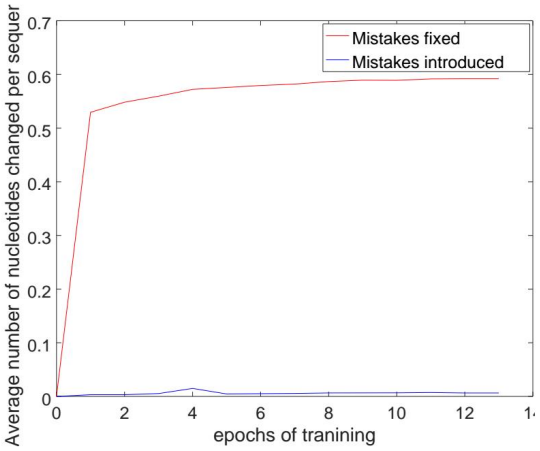
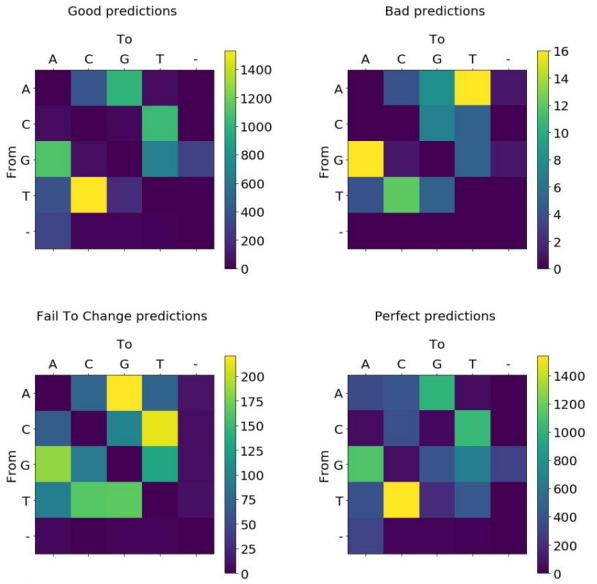


Figure 9: CNN: correct and incorrect nucleotide changes




```
Baseline error per sequence: 0.72883
Prediction error per sequence): 0.14124
Average good changes per sequence : 0.59461
Average bad changes per sequence: 0.00702
Average failure to change per sequence: 0.13422
```

Figure 10: Convolutional Network performance- Frequency of switching from y-axis nucleotide to x-axis.

B. *Recurrent Model: Many-to-many bidirectional LSTM*

- **Train curves** We see probability of predicting a correct nucleotide base increase as the recurrent model trains (see Figure 11). The validation set contains a much

higher proportion of naturally non-noisy sequences (input matches output to begin with) than does the train set, and for this reason the validation accuracy starts and remains much higher.

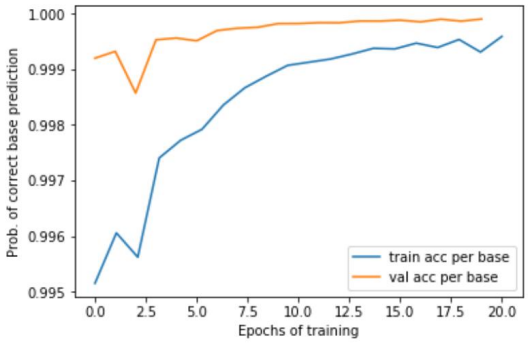


Figure 11: Training curves uses "accuracy per base". By 20 epochs, the improvement to training accuracy plateaus

- **Accuracy of predictions**
  The current recurrent model surpassed $\boxed{99.0\%}$ total set accuracy on the test set. Again, accuracy of the input test sequences (total set accuracy) was $\boxed{87.6\%}$, so we have eliminated about 92% of the noisy sequences completely.
- **Correct base predictions** The recurrent model conforms with the behavior of increase in probability of correct base prediction (see Figure 11). The validation probabilities offset train probabilities being more close to 1 as the validation set has way more exact sequences than the training set (refer Fig 3).
- **Patterns in noisy nucleotide calling**
  We observe that the perfect predictions matrix must switch very frequently from T to C, and somewhat frequently from G to A, A to G, and C to T. Other nucleotide base transfers also occur at lower frequencies. The distribution of correct changes made by the many-to-many LSTM model very closely matches the perfect predictions pattern (Figure 12), which suggests that the network is appropriately learning to distinguish between technological and biological noise. The distribution of transitions that the network fails to recognize is fairly even- while the network was able to learn the most common types of noise, the less common patterns are not detected. The bad predictions distribution closely matches the good predictions distribution, so the network is occasionally over-applying the patterns it has learned as technological noise. This could also be a sign of mislabeled truth data and should be investigated further. The network was able to take a dataset starting with 0.73 noisy incorrect nucleotides per sequence and output one with only .03 incorrect nucleotides per sequence, a 24-fold improvement. The dataset began with 12.5% of the sequences containing some noise, and the model was able to reduce this to .095% of the sequences containing some noise.

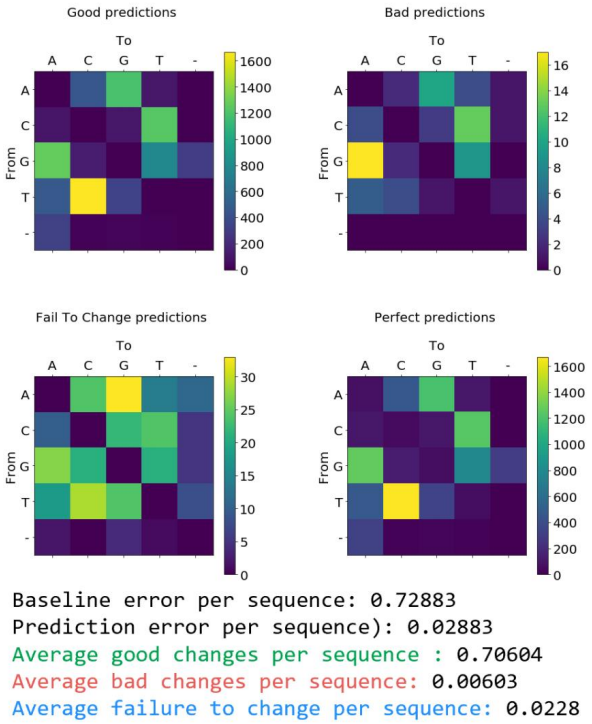


Figure 12: Frequency of switching from y-axis nucleotide to x-axis for sequential model

VI. Conclusions/Future Work

We have demonstrated the ability of neural networks to remove errors from noisy DNA sequences. The CNN gave us promising results right away, without much hyperparameter tuning. The LSTM model required much more fine-tuning, but with the right architecture has started to outperform the shallow CNN. We hypothesize that the ultimate difference in performance can be attributed to the LSTM model containing over 300,000 trainable parameters, while the CNN has just over 100,000. Both networks decreased the noise in the synthetic data set by more than a factor of 3, and if combined, may be able to do better than either individually could.

In the future, we hope to accomplish the following:

- Incorporate whether or not the network correctly distinguishes a noisy sequence from a correct one into the loss.
- Tune the depth of each network (number of layers).
- Test on different data sets to see if current weights transfer at all and if not, whether including them in the training data increases transferability.
- Combine the two networks to decrease noise even further, as they seem to have picked up on different types of errors.
- Move away from a completely supervised learning problem. Our goal in this project was to assign potentially noisy sequences to the closest reference sequence available in the microbial 16S database. However, some of the nucleotides we currently label as noise may in reality be currently undocumented biological variation. More advanced benchmarking techniques will make this method ground-breaking in microbiology research spheres.

VII. Contributions

- **Christine Tataru** Mined for datasets and did the processing for appropriate data set; worked on LSTM encoder decoder model and many-to-many LSTM model; implemented confusion matrix evaluation metric.
- **Clara McCreery** Performed data formatting for Keras platform; developed and trained the CNN architecture; implemented percent noisy sequence evaluation metric.
- **Abhishek Roushan** Literature research on seq2seq model; data formatting for NMT; Worked on Neural Machine Translation model, many-to-many LSTM model, and noisy/non-noisy sequence classification (incl. Future work); formatting presentational contents.

References

[1] Clemente JC, Ursell LK, Parfrey LW, Knight R. *The Impact of the Gut Microbiota on Human Health: An Integrative View*. Cell. 2012;148(6):1258-1270. doi:10.1016/j.cell.2012.01.035.

[2] Bull, M. and N. Plummer *Part 1: The Human Gut Microbiome in Health and Disease*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4566439/

[3] Zaranek, Alexander et al. *A Survey of Genomic Traces Reveals a Common Sequencing Error, RNA Editing, and DNA Editing*. http://journals.plos.org/plosgenetics/article?id=10.1371 /journal.pgen.1000954

[4] Allhoff, Manuel et al. *Discovering Motifs that Induce Sequencing Errors*. https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-14-S5-S1

[5] Ochman H, Elwyn S, Moran NA *Calibrating bacterial evolution.* Proceedings of the National Academy of Sciences of the United States of America. 1999;96(22):12638-12643.

[6] *Shift-and-add.* Wikipedia. https://en.wikipedia.org/wiki/Shift-and-add

[7] Waterman MS, Jones R *Consensus methods for DNA and protein sequence alignment*. Methods Enzymol. 1990;183:221-37.PMID: 2314276

[8] Zhang, Kai et al. *Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising*. https://arxiv.org/pdf/1608.03981.pdf

[9] *Study PRJEB6244*. http://www.ebi.ac.uk/ena/data/view/PRJEB6244

[10] Woolf, Max. *Pretrained Character Embeddings for Deep Learning and Automatic Text Generation.* http://minimaxir.com/2017/04/char-embeddings/

[11] Anonymous. *Convolutional Sequence Modeling Revisited.* Paper Under Review for ICLR 2018. https://openreview.net/pdf?id=rk8wKk-R-

[12] *Understanding LSTM Networks.* colah's blog. August 27 2015. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[13] *Introduction to Autoencoders* https://www.doc.ic.ac.uk/ js4416/163/website /autoencoders/index.html

[14] Genthial, Guillaume. *Seq2Seq with Attention and Beam Search* https://guillaumegenthial.github.io/sequence-to-sequence.html

[15] *Neural Machine Translation (seq2seq) Tutorial* https://www.tensorflow.org/tutorials/seq2seq

[16] Karpathy, Andrej. *The Unreasonable Effectiveness of Recurrent Neural Networks* http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[17] *BLEU (bilingual evaluation understudy)* https://en.wikipedia.org/wiki/BLEU

[18] *Sequence learning in Keras* https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html

[19] *Sequence Modeling with Neural Networks: Attention Models* https://indico.io/blog/sequence-modeling-neural-networks-part2-attention-models/

[20] *Rectified Linear Unit* https://en.wikipedia.org/wiki/Rectifier_(neural_networks)