SuperQ: A computationally efficient improvement of CNN based DNA function annotator DanQ

Elinor Löverli & Revant Gupta

***Abstract*—We have created a deep learning autoencoder of DNA sequences that is used to create data for classification by a random forest classifier. This is meant as a computationally efficient and less time consuming alternative to the CNN, RNN hybrid method DanQ, thus we have used the same dataset to train our model. Though we failed to meet our objectives we have provided the reasons for this, an alternative plan of action and the need for an alternative to DanQ in the discussion.**

I. Introduction

Interest in non-protein coding DNA motifs is increasing for the analysis of gene regulation. Motifs are short recurring sequence pattern in DNA with a biological function. They can be sequence-specific binding for nucleases and transcription factors, and regulatory elements involved in ribosome binding, transcription termination, or mRNA processing such as splicing, editing and polyadenylation.[1] Probabilistic methods using Position Specific Score Matrix (PSSM) are based on the principle that homology of the sequence can be used to annotate function.[2] These methods are easy to interpret and allow for genome-wide analysis, but other, more complex techniques can more accurately capture the sequence patterns.[3] More complex techniques are able to predict function directly from sequence, instead of from gene models and multiple species alignment, one such approach is through deep learning. Deep learning as an approach hopes to capture patterns in sequences that are relevant to its function. The advantage of using neural networks is that no explicit rules needs to created, rather the network will capture patterns that are not readily seen or interpreted by humans. Since over 98 % of the human genome is non-coding, this enables novel findings about these non-coding elements.[5] These elements are considerably harder to evaluate than genes.

A. Deep learning

Deep learning Deep learning algorithms has the capacity to use the large, high-dimensional dataset as training data and train the neural networks (DNNs) to recognise complex patterns. It has been shown that DNNs outperforms a wide variety of other information technology applications such as image and speech recognition.[4]

The algorithm consists of layers, the first layer is a representation of the raw data and with every layer the algorithm reevaluates the data into a multi-level, more abstract, representation of the data which is fed into a new one. These layers of features the algorithm learns itself without human impact.[4]

Two common classifications of deep learning networks are supervised or unsupervised learning, a classification based on how the input data is formatted. The most used method among these is the supervised learning, which is when the networks are trained on well annotated datasets as a means to evaluate the correctness of the model. The algorithm output is constrained to the same format as the label of the actual annotation and the difference in model output and the respective annotation is used to calculate an error value via an built in function which the machine uses to adjust the parameters in order to reduce the error. This is the principle of back-propagation which is fundamental to the learning process of supervised neural networks. The parameters are numbers and are often referred to as weights. Each input-output relationship has a weight score which determines the impact of a certain input in the layer. The more rare, unsupervised methods uses un-annotated data and uses methods such as clustering of similarity for the classification.[7]

In a deep learning network there are many more layers and thus more connections than in a conventional neural network.Convolutional neural networks (CNNs) are variants of DNNs that are highly suitable for pattern recognition.[5] Conventionally used to classify images, the method fundamentally is a technique that abstracts local patterns into higher level representations by convolving them. This is especially useful for sequence since DNA motifs also vary at a base level but functionally are a consequence of many different local variations. The ability of this method to predict higher level functions based on local differences is crucial. DeepSEA is a CNN developed for recognising sequence pattern in order to predict regulatory elements such as transcription factor binding, DeepSEA has been shown to outperform gkm-SVM, another machine learning algorithm predicts the function of the DNA sequence, but that instead of deep learning uses support vector machines (SVMs).

Another class of deep learning algorithms is the recurrent neural network (RNN). A RNN is implemented as a single layer though it itself contains internal layers in a manner that the a RNN feeds the processed inputs of a sequence as additional input when processing the next part of the sequence. The goal is to capture the nature of sequential information which has directionality.

A variant of RNNs are bi-directional long short-term memory networks (BLSTM), which combines the output of two RNNs, one that has processed sequential data from left to right, and one from right to left. LSTM has been a widely popular and successful in natural language processing and is commonly found in chatbots and speech recognition software.

B. DanQ

DanQ is a method that seek to exploit both these methods and combines them in one hybrid network. DanQs DNN starts with a convolutional layer that treats the DNA fed into it as an array with each of the 4 bases as channels. This is analogous to a hypothetical image that only has one line of pixels and the bases are analogous to RGB values. (Fig. 1.) These convolutions are then fed into a bidirectional LSTM

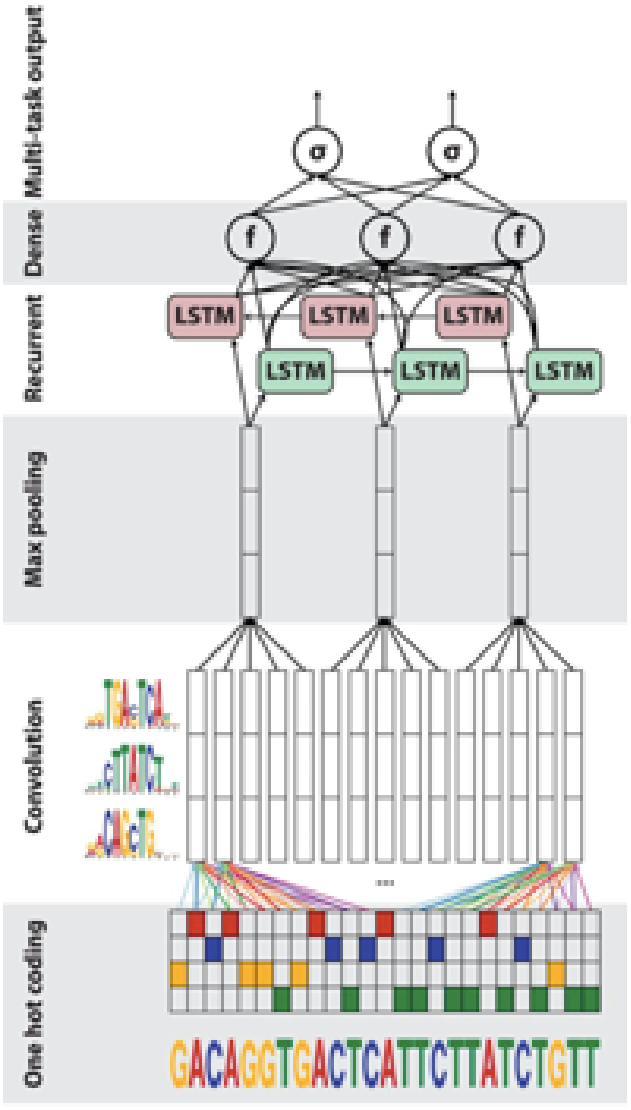

Fig. 1: Network architecture for DanQ

which process the now abstracted features as a sequence thus capturing directionality which is essential when processed motifs which are often regulators of downstream sequences. The final output of this network is a 919 element vector that has been subjected to sigmoid activation, thus the sum all the elements of this vector equal to 1. The value of each element can be interpreted as the likelihood of the class of DNA motifs that element corresponds to. Since our labels are not probabilities or likelihoods, this network trains to predict one particular class as the annotation. The challenge however is that training this network requires computational resources and time, both of which are expensive. DanQ was originally trained of a Titan V Nvidia GPU and took 6 hours per epoch, 360 hours for the full 60 epochs.

II. SuperQ

We proposed an alternate method that retains the powerful combination of convolutional and recurrent layers. (The version presented only had the convolutional layer but we added the recurrent layers since we didnt observe any major difference is training time per epoch after removing it. Also it is better that we capture the directional information using LSTM than by inverting the representations in the convolutional layer.) However instead of using this network for classification, we will broke the process into two parts.

The DNN is trained as an autoencoder and is meant to condense the sequence information captured by the convolutional and recurrent layers into a 500 feature encoding derived from an internal layer of the network. The encoded information is then used to train a random forest classifier. The reasoning for this approach is that a deep network is effective at capturing information in a dataset and the typical problem for such networks is overfitting of the data. A network like DanQ will give diminishing returns in terms of accuracy improvement as we increase the number of epochs i.e. complete passes over the data.

We seek to run the network for only 1/6th of DanQs training time and use a random forest classifier to use information captured by our autoencoder, even if it is accuracy is moderate (expectation 70-80%) to give as robust predictions as DanQ at 1/6th the time and hence increased computational efficiency.

A. Autoencoders

Autoencoders are supervised neural networks that are trained to predict its input as the output. Thus while the network architecture varies, the labels for the training data is the data itself. The value of these networks lies within the network layers, where intermediate layers can be used to generate outputs that can be interpreted as condensed versions of the same information.

In effect our autoencoder is a compression algorithm. Another important point to note is that by splitting the classifier away from the DNN, we can use the autoencoder even on unlabeled data and improve the encodings, thus even unannotated data for which need predictions can be used to improve the DNN allowing better classification.

III. Programming

A. Repository

https://github.com/aron0093/5MT003

B. Codes

1) DanQ_train_keras2.1.1.py: Since DanQ had been published in 2015 it was using an outdated version of the machine learning library keras. Keras is a wrapper around machine

learning libraries like theano, tensorflow or CNTK. We used the tensorflow backend and rewrote the code to be usable with the current version of keras.

2) SuperQ_train_keras2.1.1.py: We wrote SuperQ from scratch but we use the same data as DanQ for comparison. The version presented used just convolutional layers in the autoencoder but it has since then been updated to include BLSTM as well.

C. Resources

We used free credits ($300 worth) offered by Google on its cloud service to train our models.

We used two virtual machines each with 30 GB of HDD storage, 8 Intel haswell cores and 52 GB of RAM. We also attached a 11 GB K80 Nvidia GPU to each VM. Either machine was used to train the models parallely.

Data was obtained from the University of California at Irvins servers with full public access available.

IV. Results

A. DanQ_train_keras2.1.1.py

The code did not run for the full number of epochs. While the code does incorporate the ability to stop when the accuracy no longer improves over 5 epochs, in this case the process is being terminated due to unknown reasons. It does not appear to be a memory overflow. Despite this we can glean information from this. The most important thing to note is that DanQ performs exceptionally well on this dataset, reaching 97% percent accuracy on its first pass itself though returns seem to diminish henceforth. While this is good, 99.99% accuracy is both achievable and desirable. Another thing to note is that the validation accuracy is 98% which implies that this model does not overfit its training data however lacking an external set we cannot say whether this model would generalise to data from other sources.(Fig. 2.)

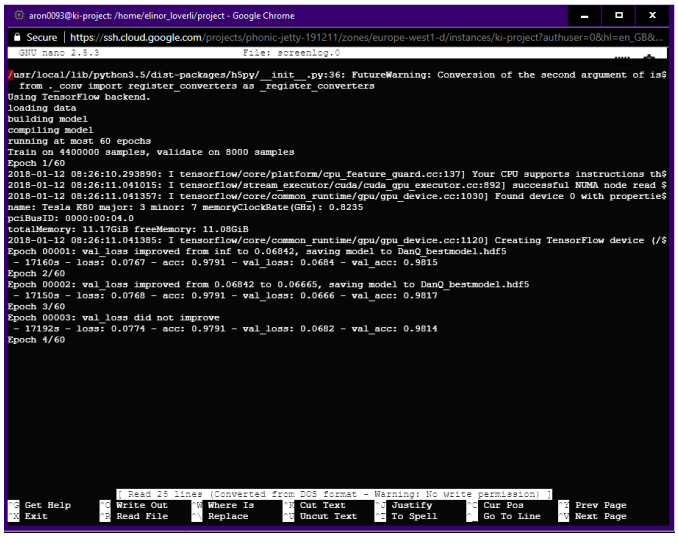

Fig. 2: Training logs for DanQ

B. SuperQ_train_keras2.1.1.py

We can see that the training time per epoch is lesser than DanQ though comparable. The accuracy does not improve over the iterations and 25% is the expected result with random chance (four bases for each position). Thus this network is not able to capture information from the input data. We got similar results with both versions.(Fig. 3.)

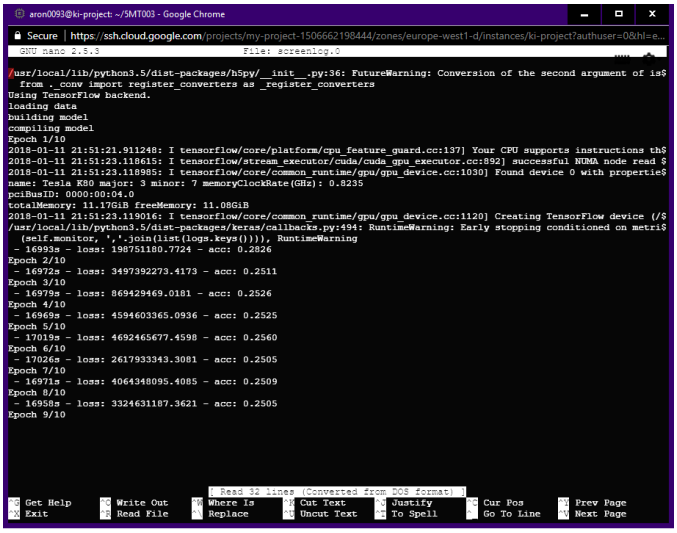

Fig. 3: Training logs for SuperQ

V. Discussion

In conclusion, our approach was a failure due to two reasons, DanQ performs well enough that it itself doesnt need to run for as many epochs as allowed for in the code severely limiting our ability to train a model in lesser amount of time.

Our network in this case was unable to capture information from the dataset. Our thinking in this regard is that unlike images and other biological sequences like proteins, a DNA sequence has only 4 bases, therefore its array representation is not as sparse.

For instance each position of the DNA seq is represented by 4 element vectors which are one hot encoded. Even of these 4 elements, more than one can be set to 1 since DNA sequences are often degenerated over multiple bases, compared to this a protein sequence represented by 21 amino acids would have a lot more sparsity.

In terms of compression, our data is not as compressible. The basic principle of compression of information is that the data must contain repeating patterns. In this case it refers to the representation of the sequence. Thus a sparse array that contains similar vectors which themselves are largely composed of zeros would be highly compressible with little loss. In our case the information is largely lost in the attempt to compress it. An alternative approach would be to perhaps represent the sequence as a 2D array with position and base as the two axes while unlike the current representation which only accounts for position and treats the bases as channels. This would increase the convolutions performed on the data while the size of a convolution would decrease. This would

dilute the information in the convolutional layer at relatively little difference in computational cost.

While we were constrained by the computational resources available to us, we feel that an autoencoder based approach confers greater generalisability than using a DNN as a classifier. While DanQ performs well on its validation data, the held out data is still from the same source and thus the classifier may not perform as well when data obtained from another source is used to test the model.

An autoencoder would be less impacted as accuracy deficiencies in the encodings can still be taken care of by the downstream classifier. Furthermore, for actual data for which predictions are needed, DanQ can only rely on previous labeled information while SuperQs autoencoder can be trained on the new data as well since it does not require labels thus improving the encodings and allowing better classification by the classifier. This point is of major importance as it allows our model to scale much better than a supervised DNN when faced with massive amount of real data that lacks annotation.

Further work would also replace accuracy with the F1 score since the classes in this data are not balanced at all, the accuracy could very well be due to the dominance of a single class. With 97% even a class representing 3 of 100 samples could suffer from misclassification therefore a deeper analysis would account for that possibility.

REFERENCES

[1] D'haeseleer P. What are DNA sequence motifs? Nature Biotechnology. 2006;24, 423425. doi:10.1038/nbt0406-423.

[2] Caldonazzo Garbelini J, Kashiwabara A, Sanches D. Sequence motif finder using memetic algorithm. BMC Bioinformatics. 2018;19(1):4. doi: 10.1186/s12859-017-2005-1.

[3] Alipanahi B, Delong A, Weirauch M, Frey B. Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. Nature Biotechnology. 2015(33);831838. doi: 10.1038/nbt.3300..

[4] LeCun Y, Bengio Y, Hinton G. Deep learning. 2015;521(7553):436-44. doi: 10.1038/nature14539.

[5] Quang D, Xie X. DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. Nucleic Acids Res. 2016;44(11):e107. doi: 10.1093/nar/gkw226.

[6] Batanlar Y, Ozuysal M. Introduction to machine learning. Methods Mol Biol. 2014;1107:105-28. doi: 10.1007/978-1-62703-748-8_7.

[7] Carpenter GA, Grossberg S. The ART of adaptive pattern recognition by a self-organizing nerual network. Computer. 1988;21:77-88. doi:10.1109/2.33.