DREAM CHALLENGE 2022
TEAM NAME: WZTR

Wuwei Zhang
University of Washington
wz86@cs.washington.edu

Xinming Tu
University of Washington
tuxm@cs.washington.edu

Wei Qiu
University of Washington
wqiu0528@cs.washington.edu

ABSTRACT

We build a sequence-to-expression CNN model to predict gene expression levels from genomic sequence. We first zero-pad all sequences to be the same length and use one-hot encoding to encode sequences into 4 by n matrices, where n is the length of the sequence. Then, we construct and append reverse complement[1] to each sequence as a form of data augmentation. Sequences are divided into the training and validation sets with a ratio of $99:1$. Our validation experiments inform model architecture, and our final submission.

1 Description of data usage

We zero-pad all input sequences to be the same length as the longest sequence in the dataset (k=142). We then use one-hot encoding, to encode sequences into 4 by 142 matrices, where 142 is the length of the longest sequence in the data. For missing inputs (i.e., N), we assign equal probability to all four possible nucleotides (i.e., a colum of 0.25 in the one-hot-encoded matrices). Then, we append reverse complement[1] to each input sequences. Thus, the resulting matrices have shapes 4 by 284. All input sequences are randomly divided into training set and validation set with a ratio of $99:1$.

2 Description of the model

We developed a deep neural network model with convolutional layers, including those with a range of dilation values [2], residual connection[3], and fully-connected layers.

Our model has the following architecture.

Input The input is a DNA sequences with appended reverse complement represented in one-hot encoding matrix. Input shape: (4, 284).

First two convolutional layers Our model begins with two convolutional layers, each with batch normalization and ReLU activation function:

- 1D convolution. (in-channel:4, out-channel:1024, kernel size:5, padding:same, dilation:1)
- Batch normalization
- ReLU
- 1D convolution. (in-channel:1024, out-channel:256, kernel size:3, padding:same, dilation:1)
- Batch normalization
- ReLU

Convolution blocks There are six convolution blocks in our model. Each convolution block is constructed in the following order:

- Hybrid convolution. (in-channel:256, out-channel:256, kernel size:3, padding:same, dilation list:[1,2,4,6])

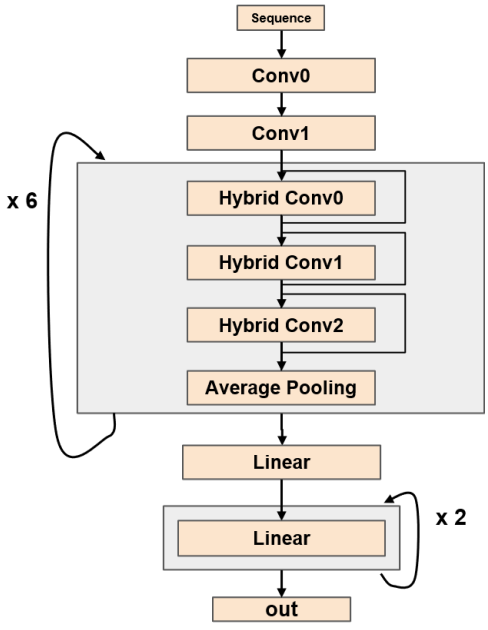
- Batch normalization
- ReLU
- Residual connection
- Hybrid convolution. (in-channel:256, out-channel:256, kernel size:3, padding:same, dilation list:[1,2,4,6])
- Batch normalization
- ReLU
- Residual connection
- Hybrid convolution. (in-channel:256, out-channel:256, kernel size:3, padding:same, dilation list:[1,2,4,6])
- Batch normalization
- ReLU
- Residual connection
- Average pooling (stride:2, kernel size:2)

Each hybrid convolution layer takes in a list of dilation values [1,2,4,6] and has 4 convolution layers (conv0, conv1, conv2, conv3) processing the input **in parallel**. All 4 convolution layers have the same in-channel, out-channel, kernel size, and padding (in-channel:256, out-channel:64, kernel size:3, padding:same). The difference is that the ith convolution layer has dilation: dilation_list[i]. This means that conv0 layer has dilaton=1, conv1 layer has dilation=2, conv2 layer has dilation=4, and conv3 layer has dilation=6. Then, we concatenate the output from all 4 convolution layers along the channel dimension. Thus, the out-channel of the hybrid convolution layer is $4 \times 64 = 256$.

Fully connected layers There are three fully connected layers in our model, which are constructed in the following order:

- Linear (input size: 256×5, output size:256)
- GeLU
- Linear (input size:256, output size:256)
- GeLU
- Dropout (0.2)
- Linear (input size:256, output size:256)
- GeLU
- Dropout (0.2)

Output Linear combination of 256 features extracted as a result of all the previous operations on the sequence to generate the predicted expression. Linear (input size: 256, output size: 1)



3 Training procedure

With 6,739,258 sequences, 99% sequences are used for training and 1% are used for validation. After one-hot encoding and data augmentation, each sequence is represented as a 4 by 284 matrix.

- Model is written in PyTorch and trained using PyTorch Lightning library.
- A batch size of 512 is used for training.
- MSE is used as loss function.
- Adam optimizer is used with a learning rate of 0.001.
- Validation loss and validation correlation are computed at the end of each epoch.
- Learning-rate scheduler is used to reduce learning rate by a factor of 0.1 when validation loss has stopped improving.
- The stopping criterion monitored is the validation loss and the model is allowed to train for 5 epochs without improvement before stopping training.

After training is done, we choose the checkpoint with the highest validation correlation to be used for prediction on test data. GPUs are used for both training and prediction.

4 Contributions and Acknowledgement

4.1 Contributions

Name	Affiliation	Email
Xinming Tu	University of Washington	tuxm@cs.washington.edu
Wuwei Zhang	University of Washington	wz86@cs.washington.edu
Wei Qiu	University of Washington	wqiu0528@cs.washington.edu

4.2 Acknowledgement

We thank University of Washington for GPU access. We thank our supervisor Sara Mostafavi for her supervision. We also thank Sheng Wang for discussion and advice.

References

[1] Hannah Zhou, Avanti Shrikumar, and Anshul Kundaje. Benchmarking reverse-complement strategies for deep learning models in genomics. *bioRxiv*, 2020.

[2] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions, 2015.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.