DeepREG

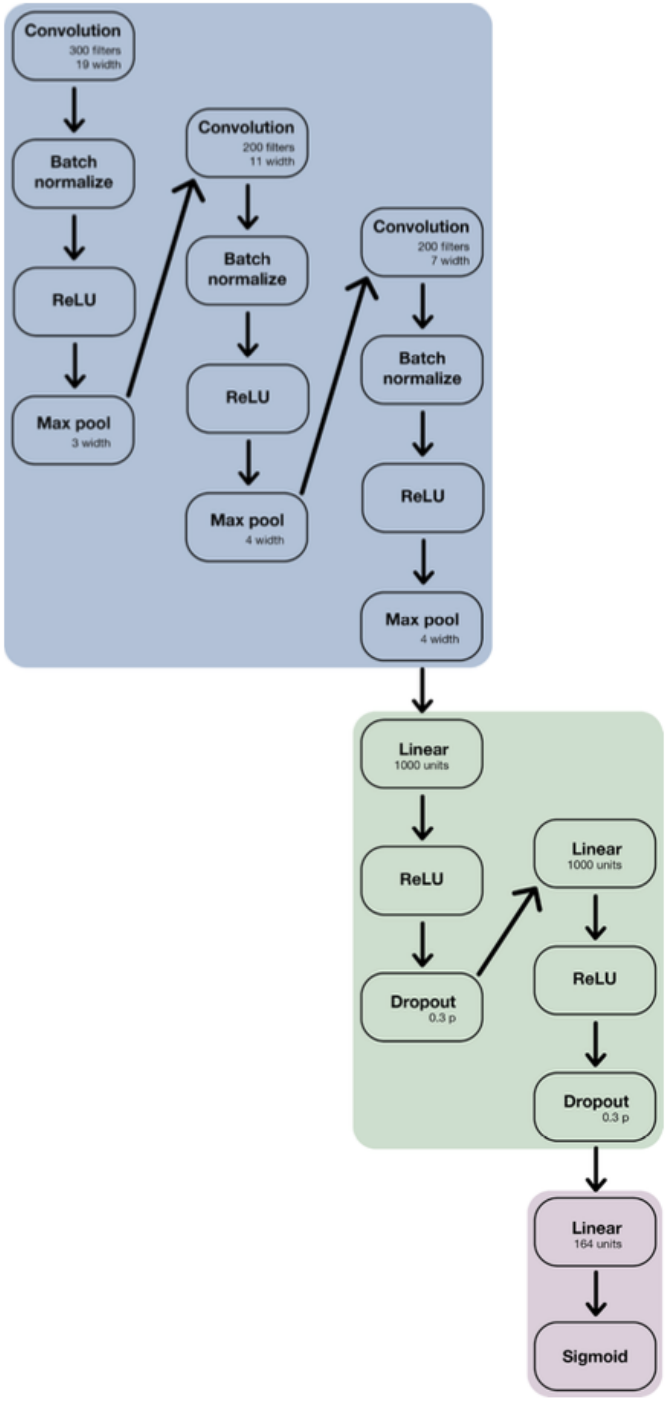
Boxiang Liu

June 5, 2017

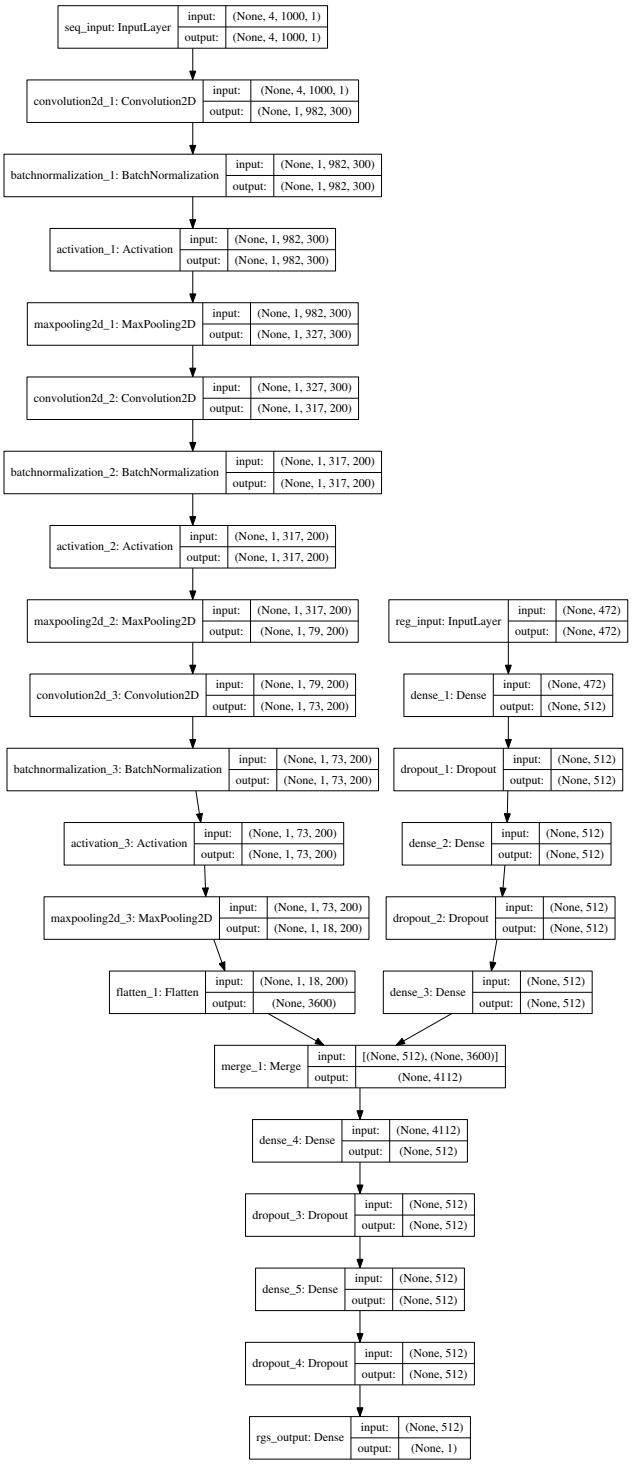
Contents

1 Basset

The Basset architecture represent the state-of-the-art for open chromatin predictions. The architecture is as follows:


Convolution
300 filters
19 width
Batch normalize
ReLU
Max pool
3 width
Convolution
200 filters
11 width
Batch normalize
ReLU
Max pool
4 width
Convolution
200 filters
7 width
Batch normalize
ReLU
Max pool
4 width
Linear
1000 units
ReLU
Dropout
0.3 p
Linear
1000 units
ReLU
Dropout
0.3 p
Linear
164 units
Sigmoid

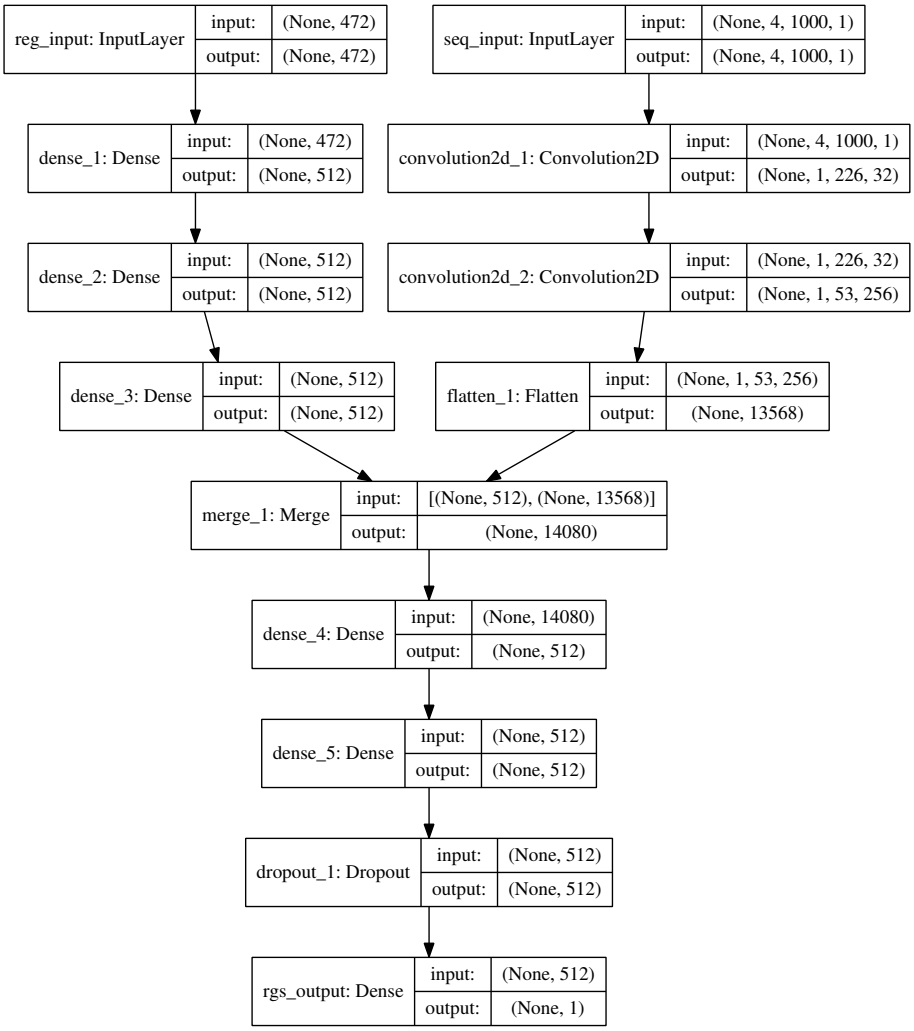
I used it for the CNN part of the network. The detailed network graph is below:


seq_input: InputLayer
input: (None, 4, 1000, 1)
output: (None, 4, 1000, 1)
convolution2d_1: Convolution2D
input: (None, 4, 1000, 1)
output: (None, 1, 982, 300)
batchnormalization_1: BatchNormalization
input: (None, 1, 982, 300)
output: (None, 1, 982, 300)
activation_1: Activation
input: (None, 1, 982, 300)
output: (None, 1, 982, 300)
maxpooling2d_1: MaxPooling2D
input: (None, 1, 982, 300)
output: (None, 1, 327, 300)
convolution2d_2: Convolution2D
input: (None, 1, 327, 300)
output: (None, 1, 317, 200)
batchnormalization_2: BatchNormalization
input: (None, 1, 317, 200)
output: (None, 1, 317, 200)
activation_2: Activation
input: (None, 1, 317, 200)
output: (None, 1, 317, 200)
maxpooling2d_2: MaxPooling2D
input: (None, 1, 317, 200)
output: (None, 1, 79, 200)
convolution2d_3: Convolution2D
input: (None, 1, 79, 200)
output: (None, 1, 73, 200)
batchnormalization_3: BatchNormalization
input: (None, 1, 73, 200)
output: (None, 1, 73, 200)
activation_3: Activation
input: (None, 1, 73, 200)
output: (None, 1, 73, 200)
maxpooling2d_3: MaxPooling2D
input: (None, 1, 73, 200)
output: (None, 1, 18, 200)
flatten_1: Flatten
input: (None, 1, 18, 200)
output: (None, 3600)
reg_input: InputLayer
input: (None, 472)
output: (None, 472)
dense_1: Dense
input: (None, 472)
output: (None, 512)
dropout_1: Dropout
input: (None, 512)
output: (None, 512)
dense_2: Dense
input: (None, 512)
output: (None, 512)
dropout_2: Dropout
input: (None, 512)
output: (None, 512)
dense_3: Dense
input: (None, 512)
output: (None, 512)
merge_1: Merge
input: [(None, 512), (None, 3600)]
output: (None, 4112)
dense_4: Dense
input: (None, 4112)
output: (None, 512)
dropout_3: Dropout
input: (None, 512)
output: (None, 512)
dense_5: Dense
input: (None, 512)
output: (None, 512)
dropout_4: Dropout
input: (None, 512)
output: (None, 512)
rgs_output: Dense
input: (None, 512)
output: (None, 1)

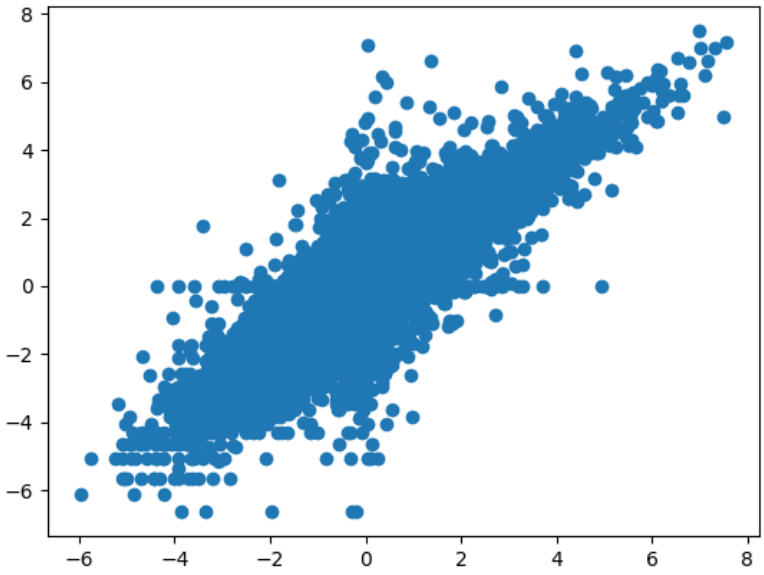
However, the network did not train properly, likely due to too many layers.

2 Reducing CNN layers

Given that 3 layers won't train, I removed one layer (in directory keras1).



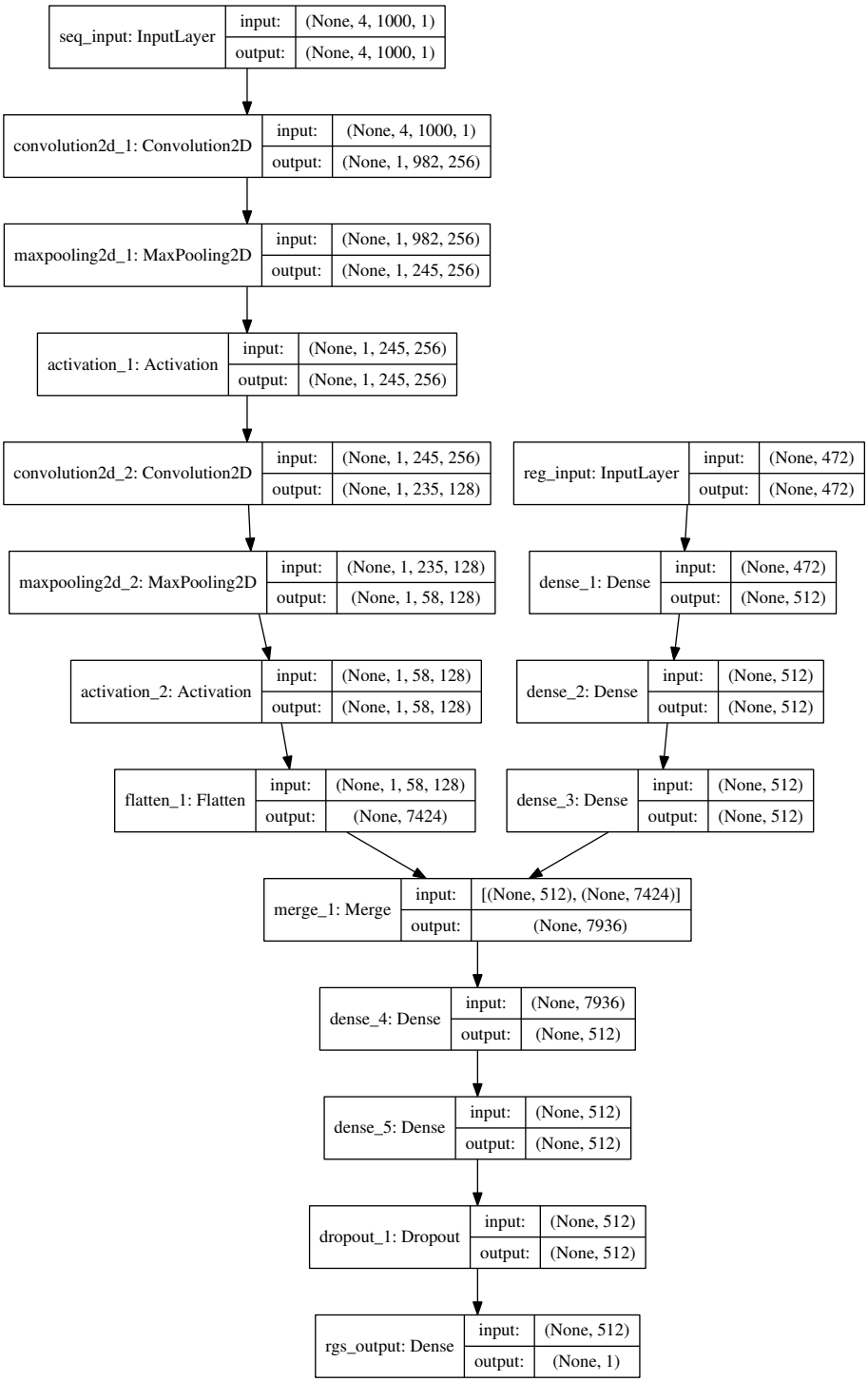
The result is quite promising.



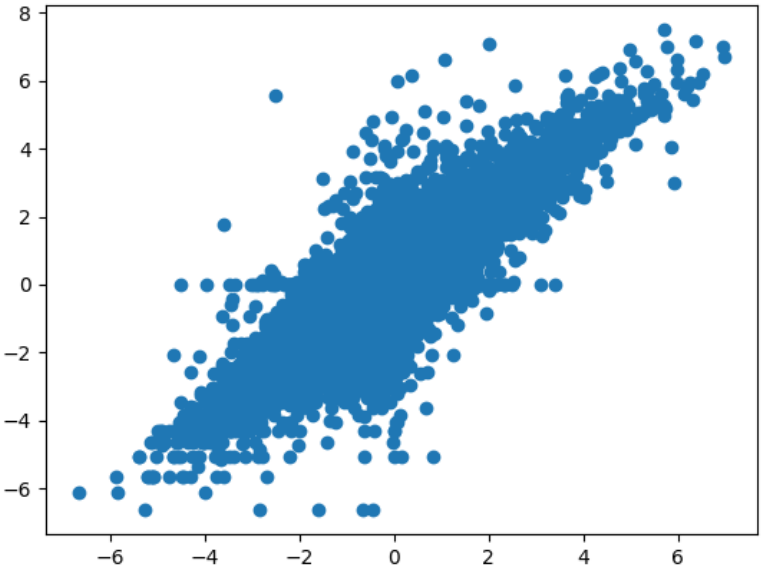
The first layer used filter width of 100, which is quite large.

3 Small filter

Since most motifs are less than 20 bps, I used a filter witdh of 19 bps.

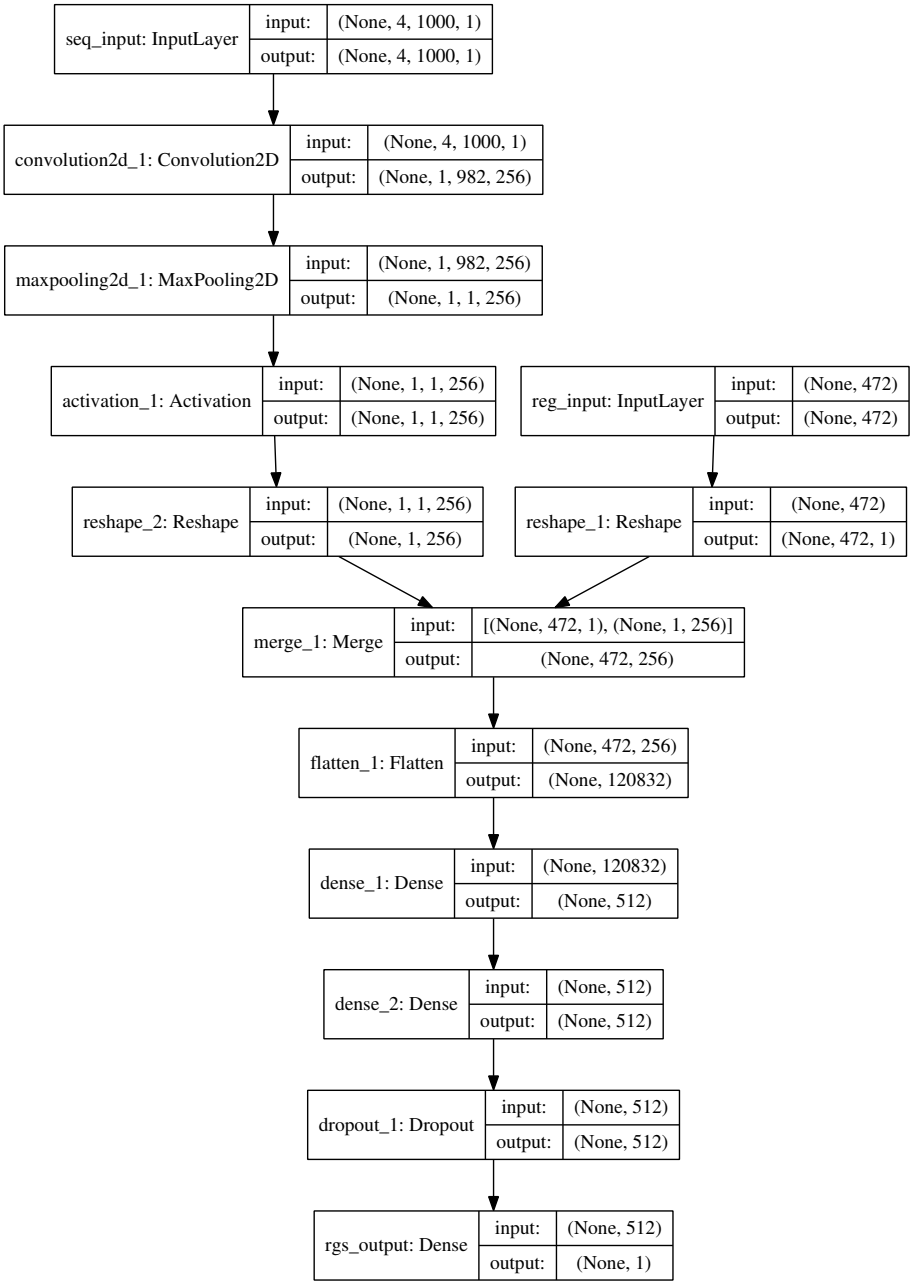

seq_input: InputLayer
input: (None, 4, 1000, 1)
output: (None, 4, 1000, 1)
convolution2d_1: Convolution2D
input: (None, 4, 1000, 1)
output: (None, 1, 982, 256)
maxpooling2d_1: MaxPooling2D
input: (None, 1, 982, 256)
output: (None, 1, 245, 256)
activation_1: Activation
input: (None, 1, 245, 256)
output: (None, 1, 245, 256)
convolution2d_2: Convolution2D
input: (None, 1, 245, 256)
output: (None, 1, 235, 128)
maxpooling2d_2: MaxPooling2D
input: (None, 1, 235, 128)
output: (None, 1, 58, 128)
activation_2: Activation
input: (None, 1, 58, 128)
output: (None, 1, 58, 128)
flatten_1: Flatten
input: (None, 1, 58, 128)
output: (None, 7424)
reg_input: InputLayer
input: (None, 472)
output: (None, 472)
dense_1: Dense
input: (None, 472)
output: (None, 512)
dense_2: Dense
input: (None, 512)
output: (None, 512)
dense_3: Dense
input: (None, 512)
output: (None, 512)
merge_1: Merge
input: [(None, 512), (None, 7424)]
output: (None, 7936)
dense_4: Dense
input: (None, 7936)
output: (None, 512)
dense_5: Dense
input: (None, 512)
output: (None, 512)
dropout_1: Dropout
input: (None, 512)
output: (None, 512)
rgs_output: Dense
input: (None, 512)
output: (None, 1)

The result is as good as using 100 width filters.

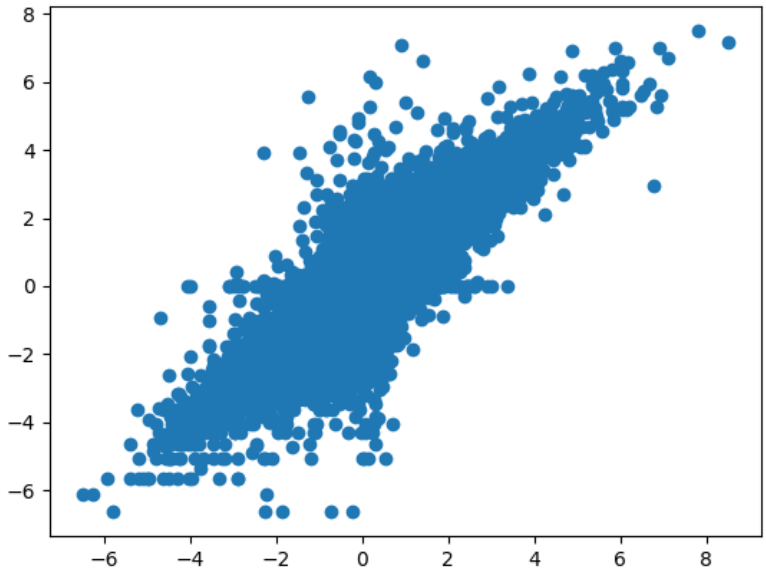


4 Single layer

When using more than one layer, either for the seq or the regulator network, the interpretability is lost. I therefore tried using one conv layer (as motif scanner) for the seq network, and no dense layer for the reg network.

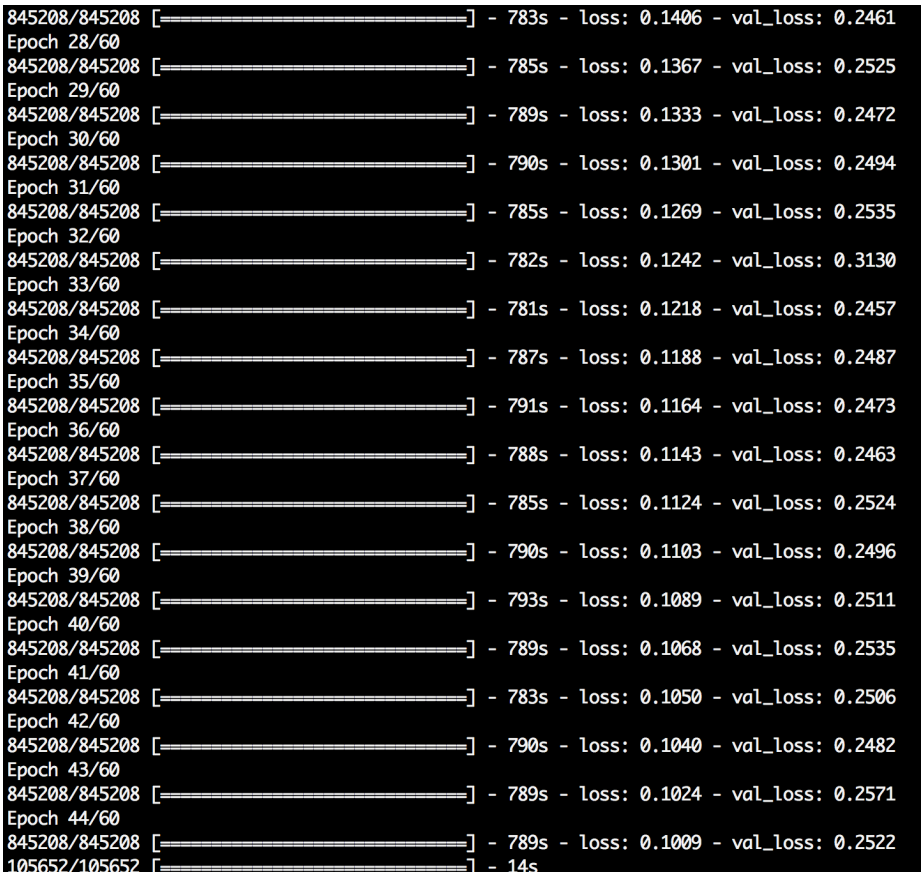


The model worked really well. The training loss dropped to almost zero after 30 epochs, and does not show any sign of plateau (compared to)small filter). However the test loss does not decrease as much indicating overfitting.


8
6
4
2
0
−2
−4
−6
−6
−4
−2
0
2
4
6
8

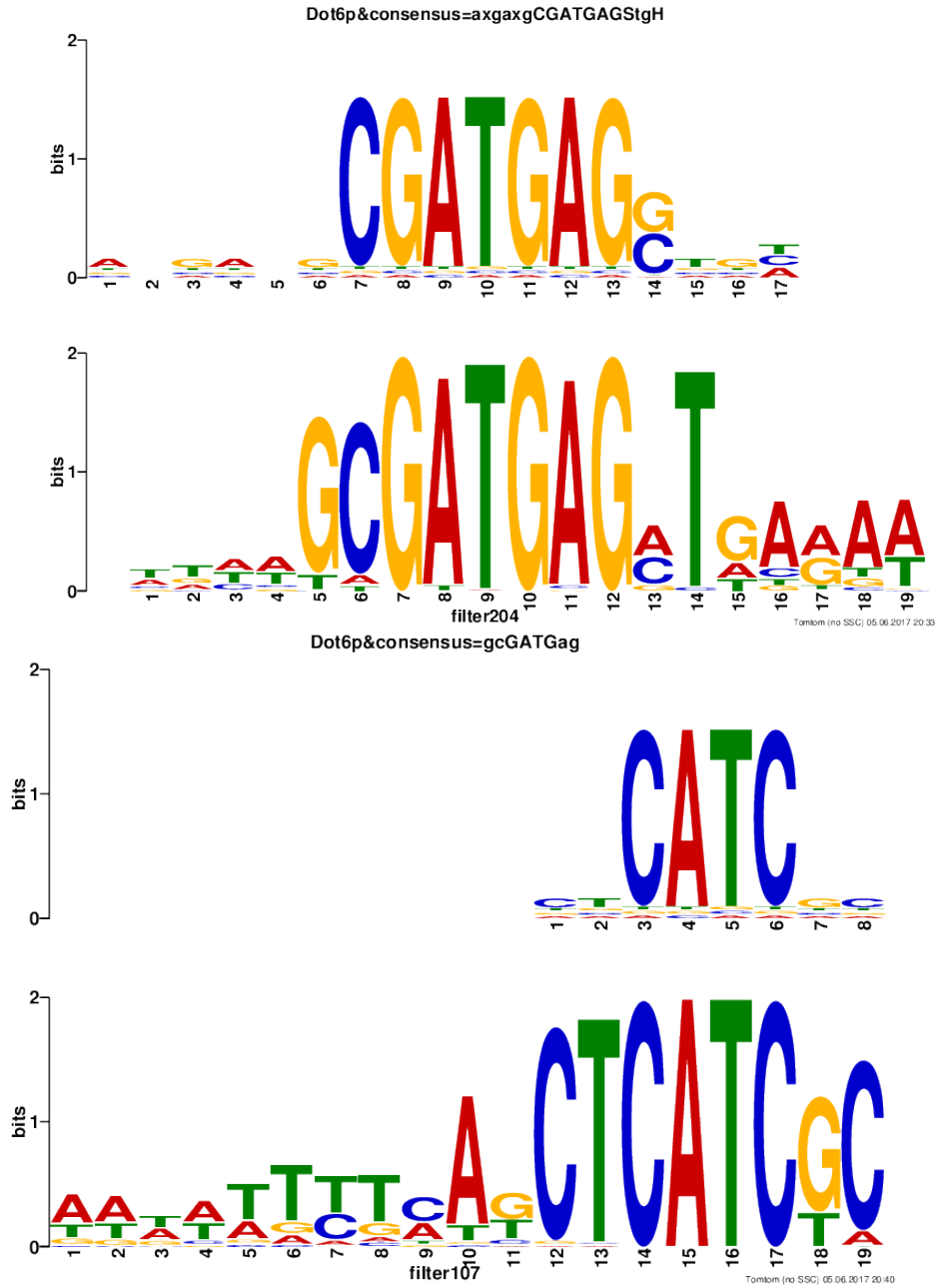
```
Epoch 10/30
845208/845208 [==============================] - 505s - loss: 0.1892 - val_loss: 0.2426
Epoch 11/30
845208/845208 [==============================] - 501s - loss: 0.1679 - val_loss: 0.2351
Epoch 12/30
845208/845208 [==============================] - 503s - loss: 0.1470 - val_loss: 0.2500
Epoch 13/30
845208/845208 [==============================] - 499s - loss: 0.1279 - val_loss: 0.2880
Epoch 14/30
845208/845208 [==============================] - 497s - loss: 0.1111 - val_loss: 0.2439
Epoch 15/30
845208/845208 [==============================] - 498s - loss: 0.0966 - val_loss: 0.2307
Epoch 16/30
845208/845208 [==============================] - 502s - loss: 0.0848 - val_loss: 0.2442
Epoch 17/30
845208/845208 [==============================] - 500s - loss: 0.0749 - val_loss: 0.2221
Epoch 18/30
845208/845208 [==============================] - 505s - loss: 0.0667 - val_loss: 0.2182
Epoch 19/30
845208/845208 [==============================] - 504s - loss: 0.0600 - val_loss: 0.2152
Epoch 20/30
845208/845208 [==============================] - 490s - loss: 0.0536 - val_loss: 0.2418
Epoch 21/30
845208/845208 [==============================] - 499s - loss: 0.0491 - val_loss: 0.2178
Epoch 22/30
845208/845208 [==============================] - 498s - loss: 0.0450 - val_loss: 0.2156
Epoch 23/30
845208/845208 [==============================] - 497s - loss: 0.0417 - val_loss: 0.2163
Epoch 24/30
845208/845208 [==============================] - 499s - loss: 0.0388 - val_loss: 0.2151
Epoch 25/30
845208/845208 [==============================] - 497s - loss: 0.0361 - val_loss: 0.2129
Epoch 26/30
845208/845208 [==============================] - 489s - loss: 0.0340 - val_loss: 0.2110
Epoch 27/30
845208/845208 [==============================] - 501s - loss: 0.0321 - val_loss: 0.2430
Epoch 28/30
845208/845208 [==============================] - 497s - loss: 0.0304 - val_loss: 0.2117
Epoch 29/30
845208/845208 [==============================] - 495s - loss: 0.0287 - val_loss: 0.2128
Epoch 30/30
845208/845208 [==============================] - 498s - loss: 0.0274 - val_loss: 0.2093
```

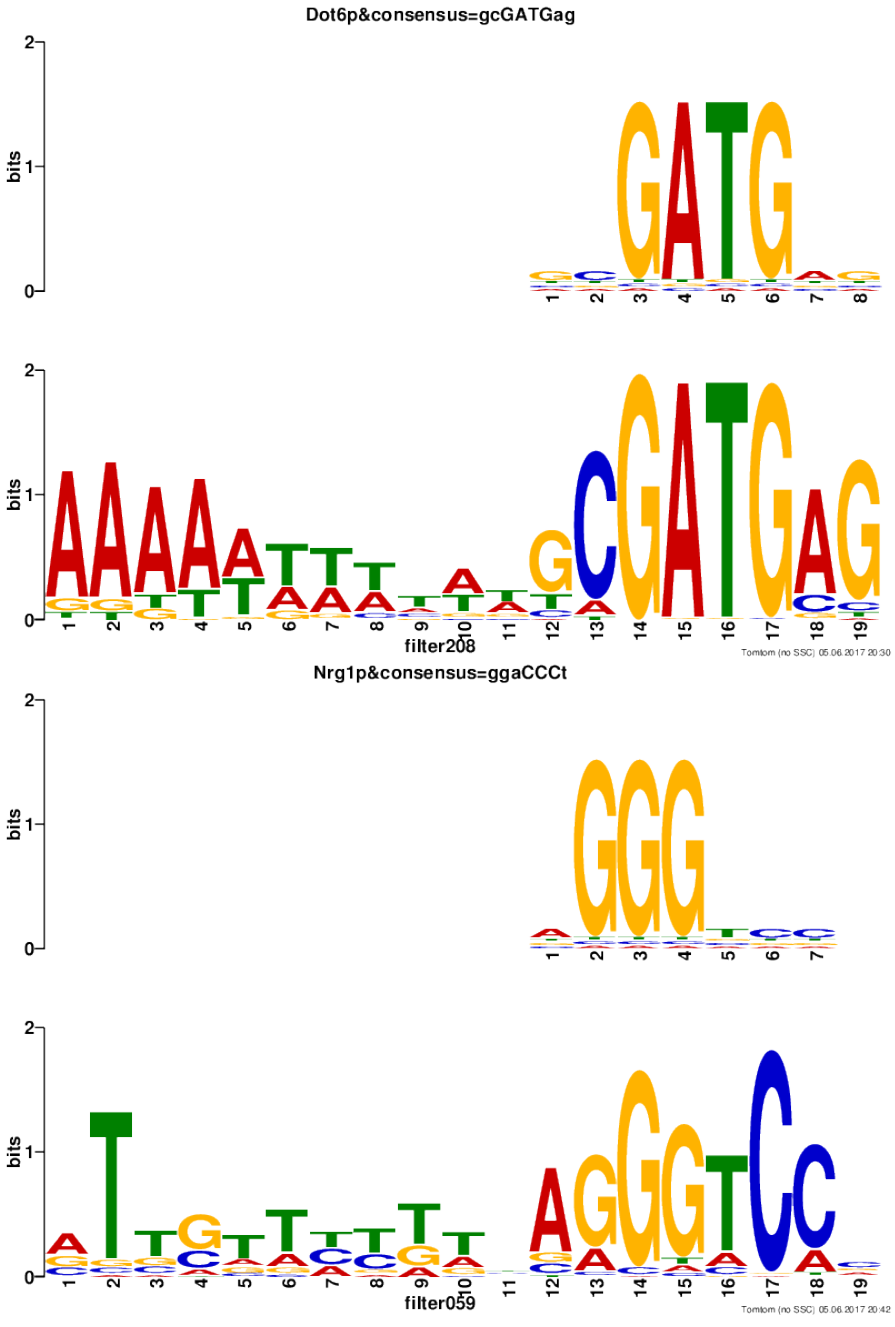
Therefore I created a new model with l1 (=1e-7) and l2 (=1e-7) on all weights regularization. Although this model prevented overfitting on the training set, the test set performance actually worsened.

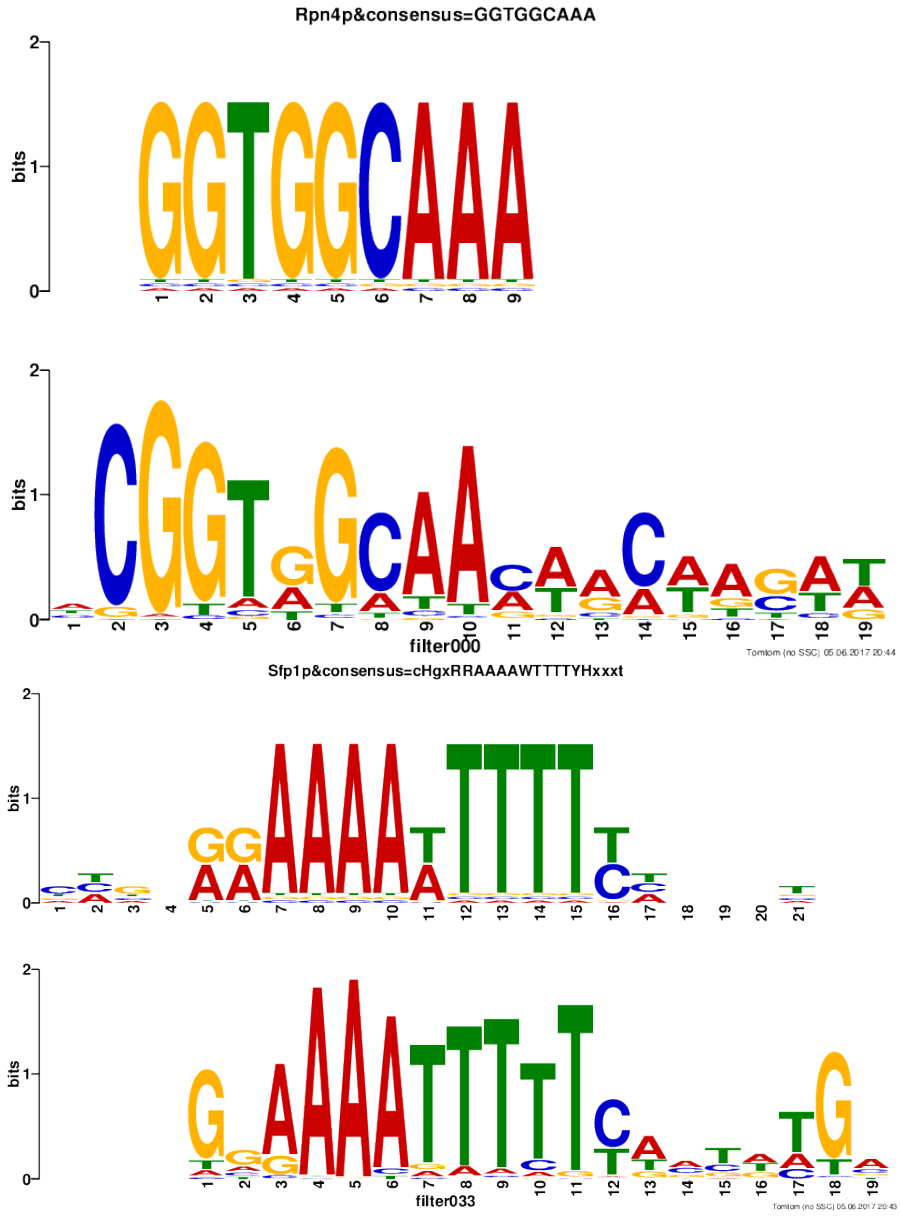


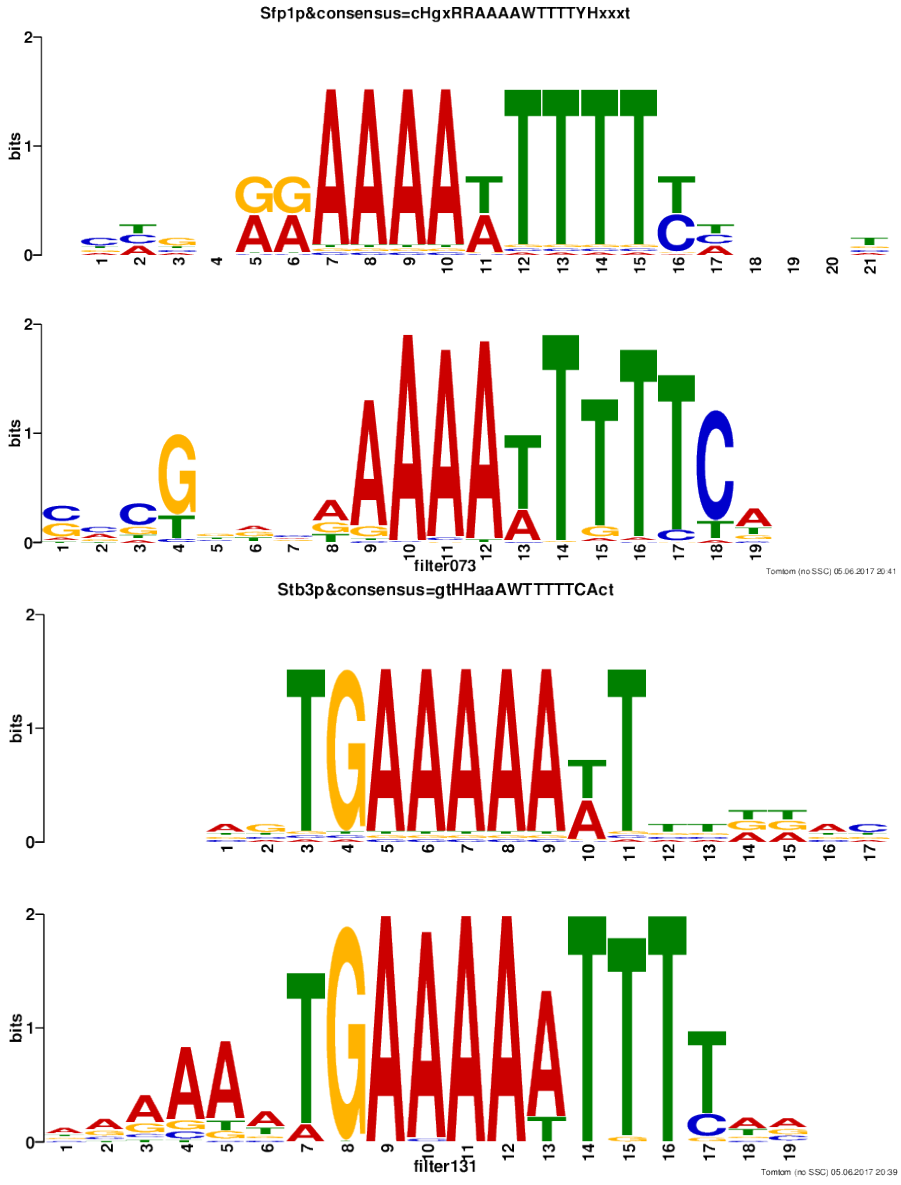
4.1 Motif discovery

Is the network find known motifs? I took top 100 sequences with largest activation for each filter and use TomTom to match them to known motifs.


Dot6p&consensus=axgaxgCGATGAGStgH
bits
filter204
Tomtom (no SSC) 05.06.2017 20:33
Dot6p&consensus=gcGATGag
bits
bits
filter107
Tomtom (no SSC) 05.06.2017 20:40


Dot6p&consensus=gcGATGag
bits
filter208
Tomtom (no SSC) 05.06.2017 20:30
Nrg1p&consensus=ggaCCCt
bits
filter059
Tomtom (no SSC) 05.06.2017 20:42


Rpn4p&consensus=GGTGGCAAA
bits
filter000
Tomtom (no SSC) 05.06.2017 20:44
Sfp1p&consensus=cHgxRRAAAAWTTTTYHxxxt
bits
bits
filter033
Tomtom (no SSC) 05.06.2017 20:43


Sfp1p&consensus=cHgxRRAAAAWTTTTYHxxxt
bits
filter073
Tomtom (no SSC) 05.06.2017 20:41
Stb3p&consensus=gtHHaaAWTTTTTCAct
bits
filter131
Tomtom (no SSC) 05.06.2017 20:39