Supplementary Material for
Biological Sequence Modeling with Convolutional Kernel Networks

Dexiong Chen∗†  Laurent Jacob‡  Julien Mairal∗

In the supplementary material, we present additional experiments mentioned in the paper.

1 Choice of model hyperparameters

We justify here the choice of the hyperparameters used in our experiments, including weight decay for CNNs, regularization parameter, bandwidth parameter in exponential kernel and perturbation intensity used in data-augmented CNN, CKN and hybrid model. We denote respectively by $k$ the filter size and $p$ the number of filters.

The scores for the following experiments are computed on a validation set, which is taken from one quarter of the training samples for each dataset and the models are trained on the rest of the training samples. For DeepBind’s datasets, we only perform validation on 100 randomly sampled datasets, which save a lot of computation time and should give similar results when using all datasets.

Weight decay for CNN. The choice of weight decay is validated on the validation set as shown in Figure 1.

Bandwidth parameter in exponential kernel. The choice of the bandwidth parameter is only validated for supervised CKN-seq and the same value is used for the unsupervised variant. Figure 2 shows the scores on the validation set when the other hyperparameters are fixed. The same choice as DeepBind’s dataset is applied to Zeng’s dataset.

Figure 1: Validation of weight decay in CNNs for DeepBind’s datasets (left) and SCOP 1.67 and its subsampled datasets (middle and right); $k = 12$ and 10 respectively for each task; $p = 128$ for both tasks.
Figure 2: Validation of the bandwidth parameter $\sigma$ for DeepBind’s datasets (left) and SCOP 1.67 (right). The regularization parameter is fixed to 1e-6 and 1.0 and $k = 12$ and 10 respectively for each task; $p = 128$ for both tasks.

Figure 3: Validation of the regularization $\lambda$ for DeepBind’s datasets (left) and SCOP 1.67 (right). The bandwith parameter is fixed to 0.3 and 0.6 and $k = 12$ and 10 respectively for each task; $p = 128$ for both tasks.

**Regularization parameter.** The choice of the regularization parameter is validated following the same protocol as the bandwidth parameter. Figure 3 shows the scores on the validation set.

**Perturbation intensity in data-augmented and hybrid model.** The perturbation amount used in the data-augmented CNN, CKN and the hybrid variant of CKN are also validated on the corresponding validation set. The scores are shown in Figure 5.

2 Hyperparameter study

We discuss here in more detail the effect of the number and size of convolutional filters and number of layers on CNN and CKN performances. We also present the discussions on the perturbation intensity in data-augmented and hybrid variants of CKN-seq.

For some of the following comparisons, we also include the oracle model, which represents the best performance achievable by choosing the optimal parameter in comparison for each dataset (whereas parameters used in our experiments are fixed across datasets). The experiment shows that a dataset-dependent parameter calibration step could possibly improve the performance, but that the potential gain would be relatively small.
Number of filters, filter size and number of layers. We show in Figure 6 that increasing the number of filters improved the performance for both supervised and unsupervised variants of CKN-seq. Furthermore, the improvement of prediction performance of the supervised one was saturated when more than 128 convolutional filters were deployed.

Both CNN and CKN-seq with one layer achieve better performance with a filter size of 12 for every fixed number of filters (Figure 7). Since this optimal value is only slightly larger than the typical length of the motifs for TFs, we deduce that the prediction mainly relies on a canonical motif while the nearby content has little contribution. However if one is interested in motif discovery only, running the algorithm with larger filter size may be of interest whenever one believes that some TF binding sites are explained by larger motifs.

Increasing the number of convolutional layers in CNNs has been shown to decrease its performance. By contrast, it does not affect the performance of CKN-seq when using a sufficient number of convolutional filters (Figure 8). Multilayer architectures allow to learn richer or more complex descriptors such as co-motifs, but may require a larger amount of data. They would also make the interpretation of the trained models more difficult. When training with 2-layer CKN models, we also notice that increasing the number of filters from 64 to 128 at the first layer or that from 16 to 64 at the second layer does not improve performance (Figure 9).

Perturbation intensity in data-augmented and hybrid CKN. We have shown that data augmentation improves both supervised and unsupervised CKN-seq. The hybrid approach has further improved data-augmented CKN-seq. We study here how the amount of perturbation used in augmenting training samples impacts performance. Specifically, we characterize the perturbation intensity by the percentage of changed characters in a sequence and show in Figure 11 the behavior of CKN-seq when increasing the amount of perturbation. By leveraging the best data-augmented unsupervised model on validation set, we train our hybrid variant and show its performance when increasing the amount of perturbation (Figure 10). We observe that the hybrid variant is more robust to larger amount of perturbation applied in the training samples than simply data-augmented one. Note that the results are consistent to those obtained on validation set (Section 1).

3 Effect of hyperparameter calibration in CNN

We study here how hyperparameter calibration as used in DeepBind could affect performance and training time for CNNs. For the calibrated variant of CNN, we used the same hyperparameter search scheme used in DeepBind for the CNN, with 30 randomly chosen calibration
Figure 5: Validation of the perturbation intensity for CKN on DeepBind’s small-scale datasets (left) and subsampled SCOP 1.67 (right); each line corresponds to data-augmented supervised (top), data-augmented unsupervised (middle) and hybrid (bottom) variants of CKN-seq. The bandwith parameter is fixed to 0.3 and 0.6, the regularization parameter is fixed to 1e-6 and 1.0, and $k = 12$ and 10 respectively for each task; $p = 128$ for both tasks.
Figure 6: Influence of the number of filters for supervised and unsupervised CKN-seq: left supervised variant with \( k = 12 \) on DeepBind’s datasets; right unsupervised variant with \( k = 10 \) on SCOP 1.67 datasets.

Figure 7: auROC scores on test datasets of DeepBind (left) and Zeng et al. (2016) (right) for single-layer CKN-seq and DeepBind-based CNNs with number of filters varying between 16, 64, 128 and filter size between 12, 18, 24; The pink and black line respectively represent mean and median.

Figure 8: Comparison between single-layer and 2-layer CKN-seq models; note that CKN64-16 has nearly the same number of parameters as CKN128.
Figure 9: Influence of the number of filters for 2-layer supervised CKN-seq on DeepBind’s datasets.

Figure 10: Effect of perturbation intensity on supervised and unsupervised CKN-seq: top: data-augmented supervised CKN-seq; bottom: data-augmented unsupervised CKN-seq; left: on DeepBind’s datasets; right: on SCOP 1.67. The number after + indicates the percentage of perturbation amount applied to the training samples.
settings and 6 training trials across the data sets.

The calibrated variant slightly outperformed hyperparameter-fixed CNN and showed similar performance to CKN-seq in the TF binding prediction task while it didn’t achieve better performance in the protein homology detection task (Figure 12).

On the other hand, training a calibrated CNN is much slower compared to hyperparameter-fixed CNN or CKN-seq. To make a fair comparison, we reimplemented and evaluated both DeepBind and CKN-seq in Pytorch. Our reimplemented model achieved almost identical performance to the original DeepBind (left panel of Figure 13) in DeepBind’s Datasets. In order to quantify the gain in training time for hyperparameter-fixed models, we measured the average training time on 50 different datasets for original DeepBind, our reimplemented DeepBind and CKN-seq on a Geeforce GTX Titan Black GPU. The right panel of Figure 13 shows that training a CKN-seq model is about 25 times faster than training the original DeepBind model and 5 times faster than our reimplemented version.

4 Influence of fully connected layer in CNN

The authors of DeepBind have used a fully connected layer in their model. However, we found that there was no significant gain with this supplementary layer in our experiments, as shown in Figure 14.
Figure 13: left: Comparison of reimplemented and original DeepBind, with the p-value of Wilcoxon unsigned-rank test; right: Average training time for DeepBind and CKN-seq on 50 datasets

Figure 14: Influence of the fully connected layer in CNN on DeepBind’s datasets: all models were trained with $p = 16$.
Figure 15: Pairwise comparison of CKN-seq and CNN on DeepBind, Zeng and SCOP 1.67 datasets. The metric is auROC for the two earlier datasets and auROC50 for the latter. The middle and bottom lines show performance of models trained on small-scale datasets.

5 Pairwise comparison of CKN and CNN

We include here some scatter plots to illustrate the pairwise comparison on each individual dataset of DeepBind and Zeng. The results are shown in 15.

6 Model interpretation and visualization

We perform the same experiments as in section 3.4 of the paper but on a larger datasets, with 9000 training samples and 1000 test samples. Motifs recovered by CKN-seq and CNN were aligned to the true motifs (Figure 16) while the logos given by CKN-seq are more informative and match better with the ground truth in terms of any distance measures (Table 1). The same conclusions can be drawn as in the small-scale case.
Figure 16: Motifs recovered by CKN-seq (middle row) and by CNN (bottom row) compared to the true motifs (top row)

Table 1: Tomtom motif p-value comparison of CKN-seq and CNN for different distance functions.

<table>
<thead>
<tr>
<th>Distance</th>
<th>FOXA1 CKN-seq</th>
<th>FOXA1 CNN</th>
<th>GATA1 CKN-seq</th>
<th>GATA1 CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>KL</td>
<td>9.77e-14</td>
<td>1.78e-08</td>
<td>3.61e-11</td>
<td>3.73e-08</td>
</tr>
<tr>
<td>Euclidean</td>
<td>1.10e-12</td>
<td>6.62e-10</td>
<td>6.49e-12</td>
<td>1.07e-07</td>
</tr>
<tr>
<td>SW</td>
<td>6.75e-11</td>
<td>4.65e-10</td>
<td>1.93e-11</td>
<td>2.71e-08</td>
</tr>
<tr>
<td>Pearson</td>
<td>2.63e-07</td>
<td>3.59e-09</td>
<td>1.72e-08</td>
<td>5.32e-07</td>
</tr>
</tbody>
</table>