

Supplementary Information for the article “Compositionally-Restricted Attention-Based Network for Materials Property Predictions”

Authors: Anthony Yu-Tung Wang, Steven K. Kauwe, Ryan J. Murdock, and Taylor D. Sparks

URL of CrabNet GitHub: <https://github.com/anthony-wang/CrabNet>

DOI of trained CrabNet model weights: <https://doi.org/10.5281/zenodo.4633866>

Supplementary Methods

As noted in the publication, we have taken additional steps to ensure that (1) no duplicate compositions were present in each of the train, validation, and test datasets, and that (2) if a composition exists in the train or validation dataset, all compounds with the same composition are removed from the validation and test datasets. Below are the details regarding the dataset processing steps.

First, all structural information (if any) was removed from the original datasets.

Then, the pre-split datasets for each material property were examined for any duplicate compositions in each of the train, validation and test splits. In our case, the duplicate compositions come from the fact that we have removed structural information from the datasets.

Thus, compounds which may exist with multiple crystal structures (with different properties) are now recorded in the dataset as the same composition, having different target values. We remove duplicate target values of the same composition by taking the mean of all available target values.

Differently, for the OQMD and MP datasets, we first sorted compounds which had duplicate entries by order of increasing formation enthalpy. Then, we took the property values associated with the lowest formation enthalpy value for each compound and used those as the target values for the compounds in each of the OQMD and MP datasets.

Second, we examine, for each of the material properties, the compositions present within the train, validation, and test splits. If a composition exists in the train dataset and is found in the validation or test dataset, we remove that composition in the validation or test dataset.

Likewise, we do the same for compositions in the validation dataset being found in the test dataset. This ensures that there is no leakage of chemical information between the dataset splits. Here we note that we use the normalized composition to detect matching composition, Na_2Cl_2 is treated as being the same composition as NaCl . This is comparable to the way how atomic fractional amounts are featurized by the CrabNet EDM featurization scheme (discussed in the publication).

We note additionally that, in order to remain comparable with other works which use the datasets including the structural information (such as the Automatminer publication), we have elected not to re-split the dataset splits into different train/validation/test datasets.

Instead, we have taken the original datasets from the publications and applied the data processing steps as mentioned above, to retain as much of the original dataset splits as possible.

CrabNet is composed of the following PyTorch modules. These are used to build the CrabNet architecture as specified in the files crabnet/{kingcrab|model}.py.

Model architecture: out_dims, d_model, N, heads
3, 512, 3, 4

```
CrabNet(
  (encoder): Encoder(
    (embed): Embedder(
      (fc_mat2vec): Linear(in_features=200, out_features=512, bias=True)
      (cbfv): Embedding(119, 200)
    )
    (pe): FractionalEncoder()
    (ple): FractionalEncoder()
    (transformer_encoder): TransformerEncoder(
      (layers): ModuleList(
        (0): TransformerEncoderLayer(
          (self_attn): MultiheadAttention(
            (out_proj): _LinearWithBias(in_features=512,
out_features=512, bias=True)
          )
          (linear1): Linear(in_features=512, out_features=2048,
bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
          (linear2): Linear(in_features=2048, out_features=512,
bias=True)
          (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
          (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
          (dropout1): Dropout(p=0.1, inplace=False)
          (dropout2): Dropout(p=0.1, inplace=False)
        )
        (1): TransformerEncoderLayer(
          (self_attn): MultiheadAttention(
            (out_proj): _LinearWithBias(in_features=512,
out_features=512, bias=True)
          )
          (linear1): Linear(in_features=512, out_features=2048,
bias=True)
          (dropout): Dropout(p=0.1, inplace=False)
          (linear2): Linear(in_features=2048, out_features=512,
bias=True)
          (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
          (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
          (dropout1): Dropout(p=0.1, inplace=False)
          (dropout2): Dropout(p=0.1, inplace=False)
        )
        (2): TransformerEncoderLayer(
          (self_attn): MultiheadAttention(
            (out_proj): _LinearWithBias(in_features=512,
out_features=512, bias=True)
```

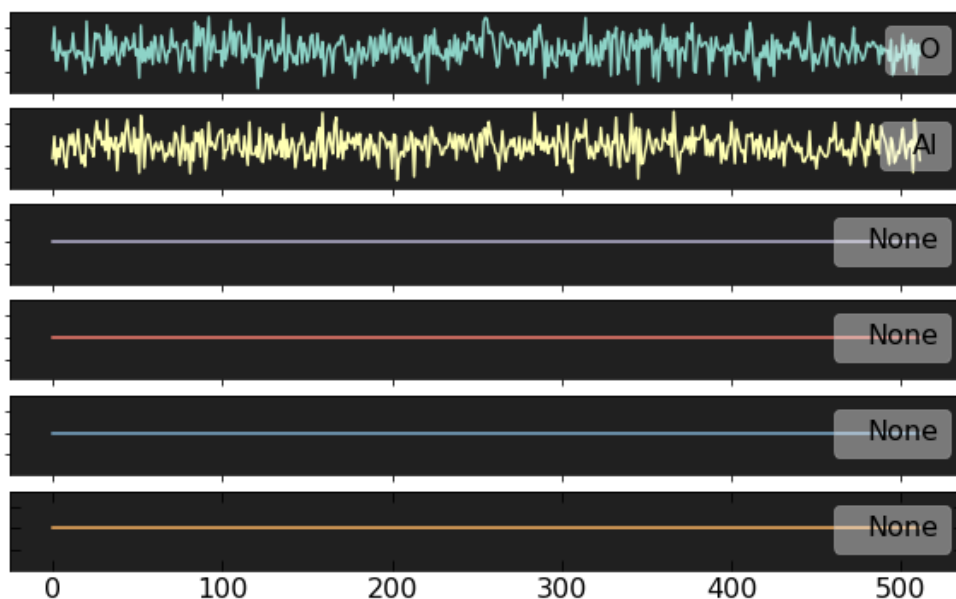
```

        )
        (linear1): Linear(in_features=512, out_features=2048,
bias=True)
        (dropout): Dropout(p=0.1, inplace=False)
        (linear2): Linear(in_features=2048, out_features=512,
bias=True)
        (norm1): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (norm2): LayerNorm((512,), eps=1e-05, elementwise_affine=True)
        (dropout1): Dropout(p=0.1, inplace=False)
        (dropout2): Dropout(p=0.1, inplace=False)
    )
)
)
)
(output_nn): ResidualNetwork
)

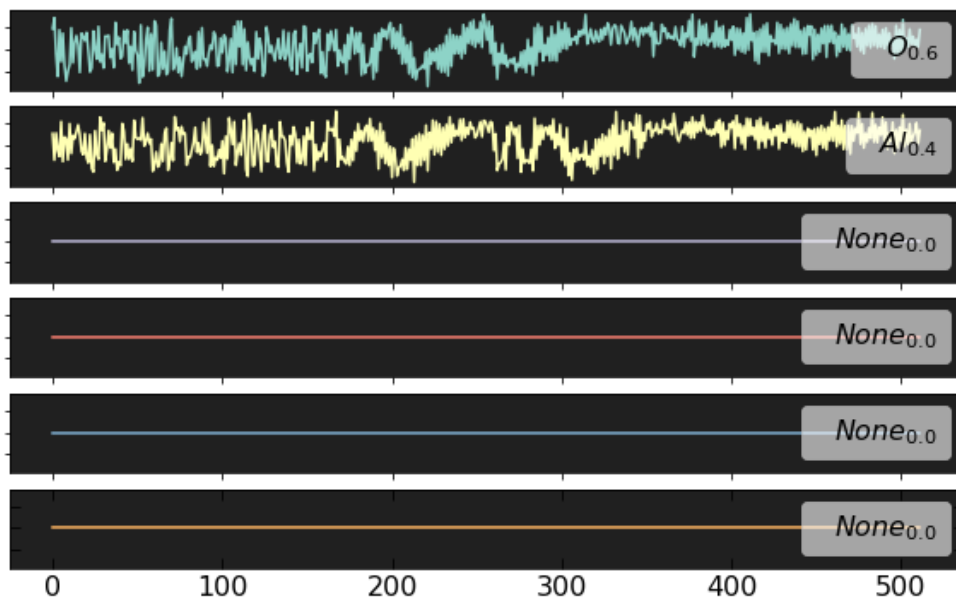
```

Number of params: 11987206

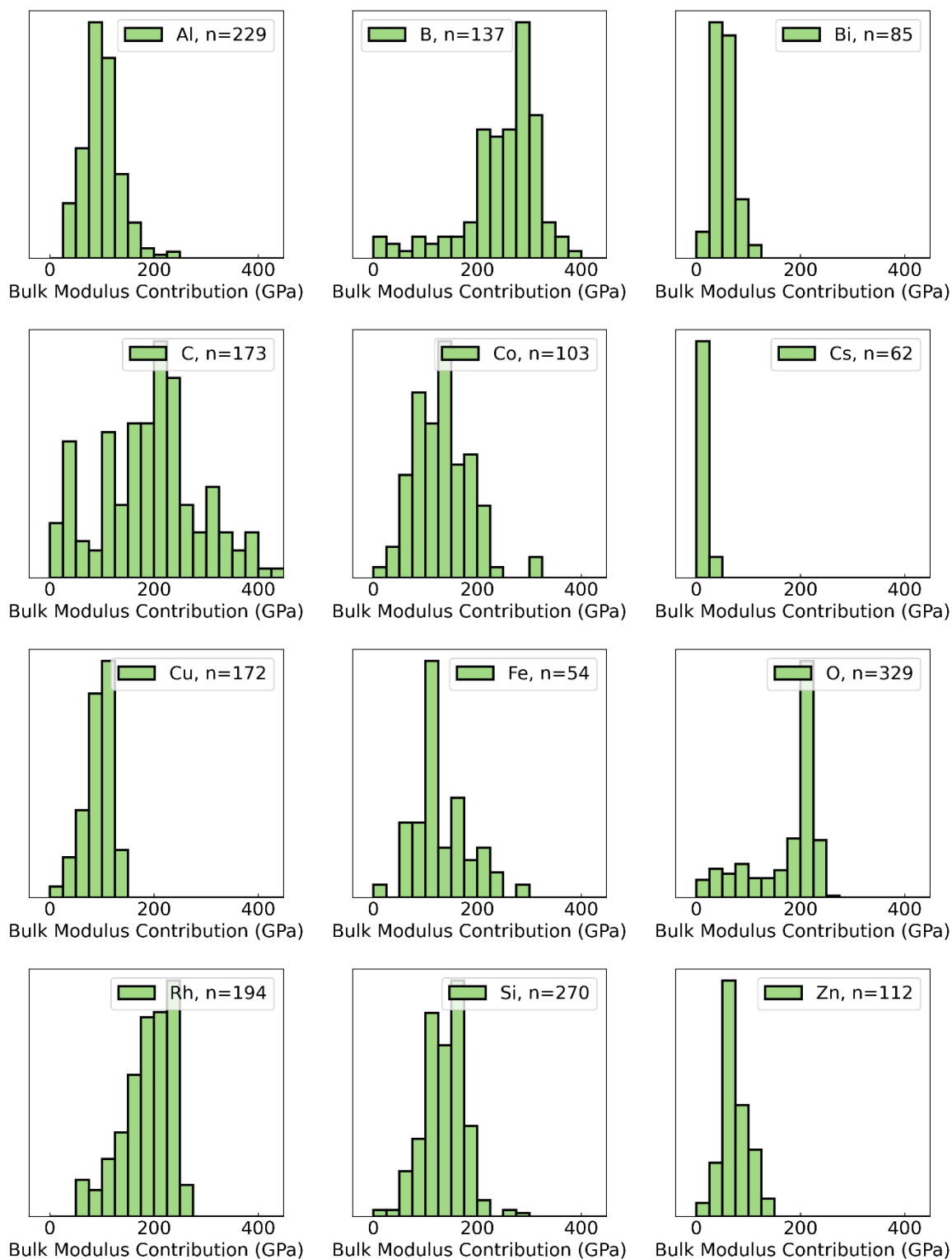
Supplementary Figures



Supplementary Figure 1. Visualization of an example EDM embedding of the compound Al_2O_3 . The elements shown were featurized using the “mat2vec” element properties. Note that the fractional embedding has not been added to the feature vectors of O and Al yet. “None” refers to no element present in the EDM.



Supplementary Figure 2. Visualization of an example EDM embedding of the compound Al_2O_3 , with the fractional embedding added element-wise to the element property embeddings. “None” refers to no element present in the EDM.



Supplementary Figure 3. Selected additional examples of the per-element contribution plots (Figure 3b in the publication) from the CrabNet model trained using the AFLOW bulk modulus dataset. See the directory `figures/contributions` in the CrabNet repository for all of the plots.