A Appendix

A.1 Data and code availability

The model and code is available at `https://github.com/Thorben010/MTEncoder`.

A.2 Model architecture

Compound encoding
We use the encoding of compounds as proposed in [34]. Elements are captured by embeddings initialized using the Mat2Vec representations [31]. Fractions employ sinusoidal encoding from the foundational transformer design, and are bifurcated into logarithmic and unchanged fractions. This approach aids smaller elemental fractions of being recognised, akin to those in dopants [34, 32]. Finally the added representations of the fractions and elemental representations are fed into the transformer encoder blocks.

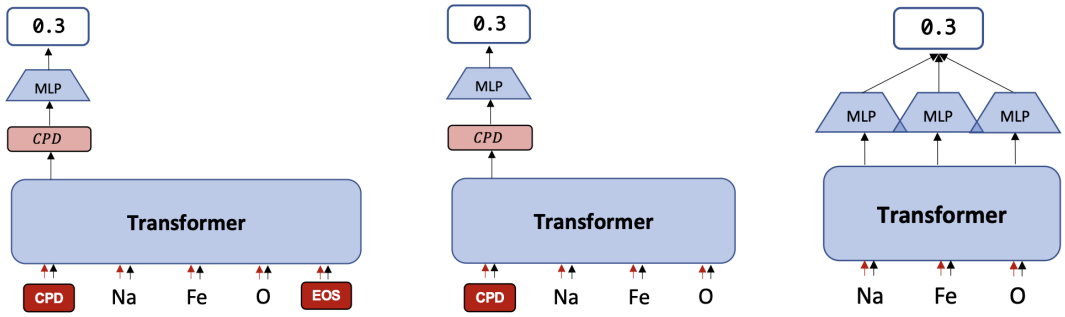

Figure 5: Schematic sketches of the different architectures used for the benchmark. From left to right: `CPD+EOS`, `CPD` and CrabNet.

Encoder
We use the transformer encoded as implemented in PyTorch [28]. In all experiments shown, except the result for 10 datasets shown in Fig. 3, four attention heads and three encoder blocks are used [32]. For the 10 dataset result a larger moodel with 6 encoder layers and 8 attention heads is used. Our architectural designs explore configurations that have not been explored by the materials science community. The `[CPD]` (compound) token is prepended as a dedicated start-of-sequence token, while the `EOS` (End-Of-Sequence) token is appended. Comparative performance metrics, as presented in Table 2, demonstrate the enhanced efficacy of our architecture relative to the established CrabNet baseline. CrabNet does not utilize special tokens and relies on predicting probabilities or properties on a per-element basis before aggregating the result to obtain predictions for entire elements.

Table 2: Comparing different model architectures, normalized to CrabNet. The `[CPD]` setting prepends a dedicated start-of-sequence token, while the `[EOS]` setting appends another dedicated end-of-sequence token. The CrabNet results have been obtained using the decoder architecture proposed in [34].

Pretraining Dataset	`CPD + EOS`	`CPD`	CrabNet (existing SOTA)
–	+5.0	+4.2	0
Bandgap	+7.8	**+8.2**	+4.5
Energy per Atom	+10.8	**+11.1**	+9.0
Formation Enthalpy	+11.2	**+12.1**	+8.8
Volume per Atom	+7.5	**+9.5**	+8.3
Average	+9.3	**+10.2**	+7.7

Interestingly, the use of the `[CPD]` only outperforms the setting where both `[CPD]` + `[EOS]` are used. Our hypothesis is that the brevity of the sequences obviates the need for an EOS token to delineate meaning across extensive sequences. Additionally, the incorporation of EOS tokens may increase the complexity of the model. The empirical findings underscore the criticality of judicious architectural selection. They also indicate promising avenues for future enhancements, particularly concerning the optimization of network depth within the framework of transfer learning.

MLP-heads
Task-specific MLPs are reinitialized randomly during the fine tuning experiments. For all tasks and experiments we use two forward layers with sizes fixed to 1024 and 512 neurons. For regression tasks no activation function is applied to the final neurons' output, classifications are handled with softmax.

Model training
To enable an unbiased comparison across strategies, the evaluation protocol on the 13 Matbench tasks was kept the same. An early-stopping strategy is used to prevent models from overfitting. Pretraining runs are performed for 150 epochs for results in Table 1. For single-task pretraining strategies, early stopping prompts a halt in training. In contrast, the joint training iterations continue until they conclude the 150 epochs. Training hardware were single NVIDIA A100 GPUs, training durations for the multi-dataset runs range between 12 to 24 hours.

A.3 Supervised pretraining

Supervised regression and classification based tasks represent the most accessible objective to incorporate chemical domain knowledge. Here all compositional data can be exploited by our model. We approach supervised property prediction by feeding the MTENCODER computed materials representation to task specific MLP-heads: $\text{MLP}_T(\texttt{[CPD]})$.

Dataset similarity in pretraining
Investigating the influence of dataset similarity, we adopt the approach established by Hargreaves, computing the Wasserstein distance between the pretraining OQMD and target Matbench datasets [10]. Results (shown in Fig. 2 (left)) highlight positive correlations between transfer effectiveness and dataset similarity. Notably, the pattern seems less apparent on smaller datasets, which could be explained by larger statistical deviations when using small data. We rarely see negative transfer with 11/13 tasks being improved.

A.4 Self-supervised pretraining

Leveraging unlabelled data through a self-supervised approach has been a prominent strategy in representation learning, spanning areas from NLP to multivariate timeseries analysis [38, 5].

Masked Element Modelling
Transferring these techniques to inorganic representation learning, we experiment with a similar masked element modelling task. Herein, each elemental token in the series of constituting elements in a compound is randomly picked by a certain probability. It is either replaced with a random elemental token, retained as it is, or substituted by a (`[MASK]`) token. Each of these events is sampled by equal probability. The modified input sequence is then contextualized by the transformer encoder and finally the chosen tokens' representaions are fed to a multi-class classification MLP head for element reconstruction.

Masking ratio
We conduct experiments to find suitable masking ratios (Fig. 2, right). Results indicate, that a replacement of 30% yields the best outcome. This is about the expectation value of masking one element per compound in the tested datasets. We believe these results reflects the high information density apparent to the very short sequences used during elemental property prediction, where sufficient remaining context of elements in the compounds proves crucial to retain the material classes's context.

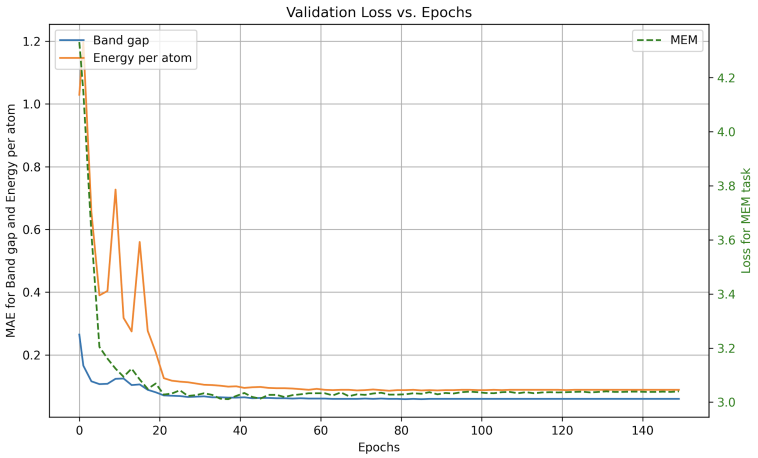

Figure 6: Validation loss for a joint training across three tasks by MTENCODER.

A.5 Performance

We report the performance as obtained by the 10 property pretrained model in Fig. 3. Note that the results have been obtained by a single model, without any task-specific hyperparameter optimisation for pretraining or finetuning. In comparison statistics reported on Matbench, are obtained by extensive hyperparameter tuning and model selection for each individual task [1].

Table 3: Table of tasks and their corresponding mean absolute error (MAE) values (rounded to four significant figures).

Task	MAE
castelli	0.1265
dielectric	0.3417
elasticity_log10(G_VRH)	0.08636
elasticity_log10(K_VRH)	0.06526
expt_gap	0.3106
expt_is_metal	0.08244
glass	0.14499
jdft2d	50.92
mp_e_form	0.08346
mp_gap	0.2576
mp_is_metal	0.09708
phonons	59.35
steels_yield	199.3

A.6 Datasets

We use the OQMD [30] datasets in v.01 with a 5-fold cross validation split. The reported performance equals an average across all five runs. The OQMD properties reported are split in a 70/15/15 ratio for training/test/validation [30]. We keep all compositions as listed in the dataset. For Fig. 2, right the *mp e hull* dataset from [35] was used.

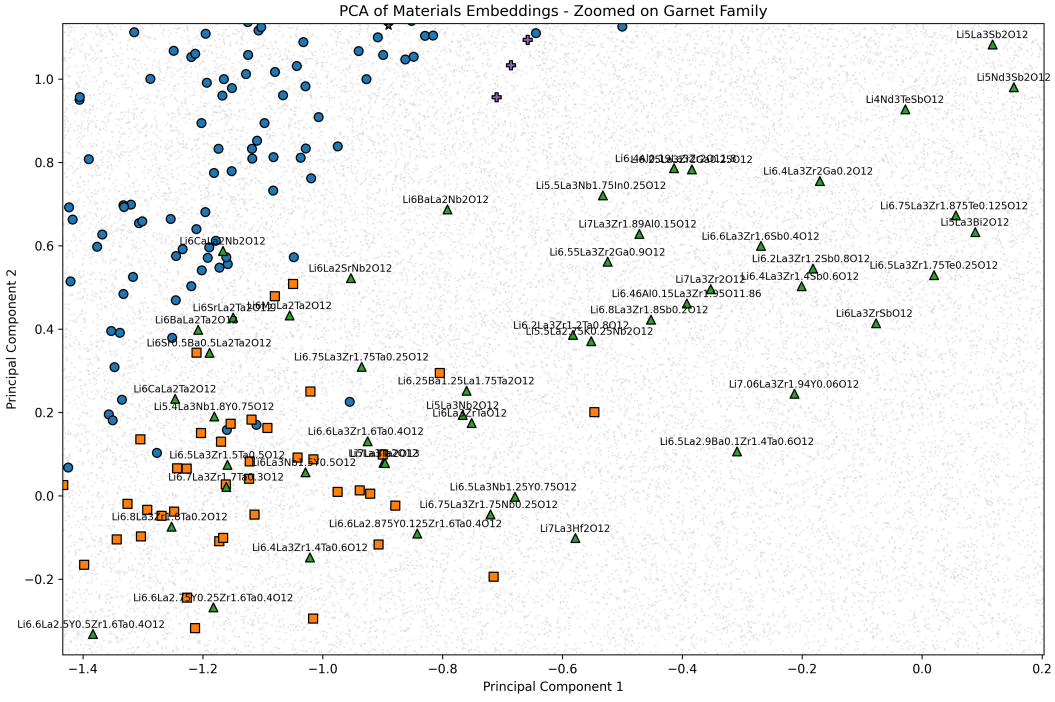

Figure 7: Magnified space of lithium garnet electrolytes, visualized by MTENCODER representations, dimensionally reduced via PCA.