A Appendix

A.1 Proof of (7)

This statement follows closely the arguments by Dusson *et al.* (2022); Drautz (2020) and others.

$$\begin{aligned}\sum_{j_1,\dots,j_n}\sum_{\boldsymbol{k}} w_{\boldsymbol{k}}\prod_s \phi_{k_s}(x_{j_s}) &= \sum_{\boldsymbol{k}} w_{\boldsymbol{k}} \sum_{j_1,\dots,j_n}\prod_s \phi_{k_s}(x_{j_s}) \\ &= \sum_{\boldsymbol{k}} w_{\boldsymbol{k}} \prod_{s=1}^{n}\sum_j \phi_{k_s}(x_j) \\ &= \sum_{\boldsymbol{k}} w_{\boldsymbol{k}} \prod_{s=1}^{n} A_k \\ &= \sum_{\boldsymbol{k}} w_{\boldsymbol{k}} \boldsymbol{A}_{\boldsymbol{k}}.\end{aligned}$$

A.2 Custom notation and indexing

We briefly contrast our notation for Clebsch–Gordan coefficients (11) with the standard notation. By means of example, consider the group $SO(3)$ in which case the Clebsch–Gordan equations are written as

$$\sum_{m_1' m_2'} C^{LM}_{l_1 m_1' l_2 m_2'} \rho^{l_1}_{m_1' m_1}(g) \rho^{l_2}_{m_2' m_2}(g) = \sum_{M'} \rho^{L}_{M M'}(g) C^{L M'}_{l_1 m_1 l_2 m_2}. \tag{25}$$

In this setting, our index $\boldsymbol{\alpha}$ simply enumerates all possible such coefficients. One can often assign a natural meaning to this index, e.g., for the group $SO(3)$ it is given by the pair of angular quantum numbers (l_1, l_2). Specifically, in this case, we obtain

$$C^{\boldsymbol{\alpha},LM}_{l_1 m_1 l_2 m_2} = \begin{cases} C^{LM}_{l_1 m_1 l_2 m_2}, & \text{if } \boldsymbol{\alpha} = (l_1, l_2), \\ 0, & \text{otherwise,} \end{cases} \tag{26}$$

where $C^{LM}_{l_1 m_1 l_2 m_2}$ are the Clebsch–Gordan coefficients in the classical notation. Thus, the additional index $\boldsymbol{\alpha}$ is not really required in the case of $SO(3)$, nor our other main example, $SO(1,3)$. Our notation is still useful to organize the computations of equivariant models, especially when additional channels are present, which is usually the case. Moreover, it allows for easy generalization to other groups where such a simple identification is not possible (Steinberg, 1961).

A.3 Equivariance of G-cluster expansion

The equivariance of the G-cluster expansion is easily verified by applying a transformation g to the input,

$$\begin{aligned}\boldsymbol{B}^{K}_{\boldsymbol{\alpha}} \circ g &= \sum_{\boldsymbol{k}} \mathcal{C}^{\boldsymbol{\alpha},K}_{\boldsymbol{k}} \boldsymbol{A}_{\boldsymbol{k}} \circ g \\ &= \sum_{\boldsymbol{k}} \mathcal{C}^{\boldsymbol{\alpha},K}_{\boldsymbol{k}} \left(\sum_{\boldsymbol{k}'} \prod_t \rho_{k_t, k_t'}(g) \boldsymbol{A}_{\boldsymbol{k}'} \right) \\ &= \sum_{\boldsymbol{k}'} \left(\sum_{\boldsymbol{k}} \mathcal{C}^{\boldsymbol{\alpha},K}_{\boldsymbol{k}} \prod_t \rho_{k_t, k_t'}(g) \right) \boldsymbol{A}_{\boldsymbol{k}'} \\ &= \sum_{\boldsymbol{k}'} \left(\sum_{K'} \rho_{K K'}(g) \mathcal{C}^{\boldsymbol{\alpha},K'}_{\boldsymbol{k}'} \right) \boldsymbol{A}_{\boldsymbol{k}'} \\ &= \sum_{K'} \rho_{K K'}(g) \boldsymbol{B}^{K'}_{\boldsymbol{\alpha}}.\end{aligned} \tag{27}$$

A.4 Completeness of the basis and Universality of MACE

We explain in which sense the basis $\boldsymbol{B}^K_{\boldsymbol{\alpha}}$ is a complete basis, and briefly sketch how to prove this claim. The argument is contained almost entirely in (Dusson *et al.*, 2022) and only requires a single modification, namely Step 3 below, using a classical argument from representation theory. We will therefore give only a very brief summary and explain that necessary change.

We start with an arbitrary equivariant property Φ^V embedded in V where we have a representation, i.e. the actual target property is Φ is then given as a linear mapping from V to Z. For technical reasons, we require that only finitely many entries Φ^V_K may be non-zero, but this is consistent with common usage. For example, if $G = O(3)$ and if Φ is a scalar, then $\Phi^V_0 = \Phi$, while all other $\Phi^V_{LM} \equiv 0$. If Φ is a covariant vector, then Φ^V_{LM} is non-zero if and only if $L = 1$; and so forth. For other groups, the labeling may differ but the principle remains the same.

1. Convergence of the cluster expansion. The first step in our parameterisation is to approximate Φ^V in terms of a truncated many-body expansion (4). It is highly application-dependent on how fast this expansion converges. Rigorous results in this direction in the context of learning interatomic potentials can be found in (Bachmayr *et al.*, 2021; Thomas *et al.*, 2022). A generic statement can be made if the number of input particles is limited by an upper bound, in which case the expansion becomes exact for a finite N. This case leads to the uniform density result stated in Theorem 4.1. We adopt this setting for the time being and return to the pointwise convergence setting below.

In the uniform convergence setting we also require that the domain Ω is compact.

Throughout the remainder of this section we may therefore assume that an N can be chosen as well as smooth components $\varphi^{(n)}$ such that the resulting model $\Phi^{V,N}$ approximates Φ^V to within a target accuracy ϵ,

$$|\Phi^{V,N}_K(\boldsymbol{x}) - \Phi^V_K(\boldsymbol{x})| \le \epsilon \qquad \forall \boldsymbol{x} \in \mathrm{msets}(\Omega).$$

2. The density of the embedding. As already stated in the main text, if the components $\varphi^{(n)}_K$ are smooth, and the embedding $\{\phi_k\}_k$ is dense in the space of one-particle functions (5) then it follows that the $\varphi^{(n)}_K$ can be expanded in terms of the tensor product basis $\phi_{\boldsymbol{k}} := \otimes_{s=1}^n \phi_{k_s}$ to within arbitrary accuracy. The precise statement is the following standard result of approximation theory: if $\mathrm{span}\{\phi_k\}_k$ are dense in $C(\Omega)$, then $\mathrm{span}\{\phi_{\boldsymbol{k}}\}_{\boldsymbol{k}}$ are dense in $C(\Omega^n)$. That is, for any $\epsilon > 0$, there exist approximants $p^{(n)}_K$ such that

$$\|\varphi^{(n)}_K - p^{(n)}_K\|_\infty \le \epsilon.$$

3. The density of the symmetrized basis. The next and crucial step is to show that, if the $\varphi^{(n)}_K$ are equivariant, then the $p^{(n)}_K$ may be chosen equivariant as well without loss of accuracy. If the group G is compact then the representations ρ can be chosen unitary (Broecker, 1985). In that case, the argument from (Dusson *et al.*, 2022) can be used almost verbatim: let

$$\bar{p}^{(n)}(\boldsymbol{x}) := \int_G \rho(g)^{-1} p^{(n)}(g\boldsymbol{x})\, H(dg),$$

where H is the normalized Haar measure then $\bar{p}^{(n)}$ is equivariant by construction and

$$\begin{aligned}
&\left|\varphi^{(n)}(\boldsymbol{x}) - \bar{p}^{(n)}(\boldsymbol{x})\right| \\
&= \left| \int_G \rho(g)^{-1} \Big(\varphi^{(n)}(g\boldsymbol{x}) - p^{(n)}(g\boldsymbol{x}) \Big)\, H(dg) \right| \\
&\le \int_G \left| \varphi^{(n)}(g\boldsymbol{x}) - p^{(n)}(g\boldsymbol{x}) \right| H(dg) \\
&\le \int_G \|\varphi^{(n)} - p^{(n)}\|_\infty \, H(dg) \le \epsilon.
\end{aligned}$$

If the group is not compact, then one can apply "Weyl's Unitary Trick" (see (Bourbaki, 1989), Ch. 3): first, one complexifies the group (if it is real) and then constructs a maximal compact subgroup $K_{\mathbb{C}}$ of the complexification. This new group K will have the same representation as G and in virtue of being compact, that representation may again be chosen unitary. Therefore, symmetrizing $p^{(n)}$ with respect to $K_{\mathbb{C}}$ results in an approximant that is not only equivariant w.r.t. $K_{\mathbb{C}}$ but also equivariant w.r.t. G.

4. The density of the basis $\boldsymbol{B}^K_{\boldsymbol{\alpha}}$. As the last step one can readily observe that the symmetrization and cluster expansion steps can be exchanged. I.e. first symmetrizing and then employing the steps (7) result in the same model. Letting $\epsilon \to 0$ in the foregoing argument while fixing the number of particles $\#\boldsymbol{x}$ results in all errors vanishing. Note that this will in particular require taking $N \to \infty$.

5. Pointwise convergence. To obtain density in the sense of pointwise convergence we first introduce the *canonical cluster expansion* without self-interacting terms

$$\Phi_K(\boldsymbol{x}) = \sum_{n=0}^{\infty} \sum_{j_1 < \cdots < j_n} v_K^{(n)}(x_{j_1}, \ldots x_{j_n}).$$

The difference here is that the summation is only over genuine sub-clusters. Because of this restriction the series is finite for all multi-set inputs $\boldsymbol{x}$. In other words, it converges in the pointwise sense.

One can easily see that v_n can be chosen (explicitly) to make this expansion exact. After truncating the expansion at finite $n \leq N$ and then expanding the potentials $v_K^{(n)}$ one can exactly transform the canonical cluster expansion into the self-interacting cluster expansion. This procedure is detailed in (Dusson *et al.*, 2022; Drautz, 2020).

The arguments up to this point establish the claimed universality for the linear ACE model. The corresponding universality of the TRACE model follows immediately from (Darby *et al.*, 2022). Since a single layer of the MACE model is a TRACE model, this completes the proof of Theorem 4.1.

A.5 Product of groups

Let G_1 and G_2 be two reductive Lie groups, and form the direct product group $G_1 \times G_2$. and ρ_1 be a associated irreducible representations then $\rho_1 \otimes \rho_2$ is an irreducible representation of $G_1 \times G_2$. One can then generate any Clebsch–Gordan coefficient of the product group $G_1 \times G_2$ using the algorithm presented above. It is of particular interest in the case of the equivariant message passing networks on points clouds, where the group of interest is $G \times S_n$.

A.6 Symmetric Tensor products

The permutation group is an important concept in the context of tensor products. It can be useful to focus on a subset of the full tensor product space that exhibits certain permutation equivariance. For example, the spherical harmonics are defined as the permutation-invariant part of a tensor product.

The symmetric tensor product can be thought of as a change of basis, or projector, from the tensor product to the symmetric part of the tensor product. In the case of a tensor product of correlation order four we have,

$$S_\nu = B_{\nu;ijkl} x_i y_j z_k w_l \tag{28}$$

where B is the change of basis that satisfies:

$$B_{\nu;ijkl} = B_{\nu;\sigma(ijkl)} \forall \sigma \in S_4 \tag{29}$$

We propose in `lie-nn` a new algorithm used to calculate B. The Symmetric Tensor Product is calculated using a tree structure, starting at the leaves and progressing towards the trunk. The leaves are the basis of the individual indices, and they are combined and constrained at each step to impose symmetry.

A.7 Computing the irreps from input representations

For some groups, the computation of the generators X can become a very involved task. However in most applications, the data itself is already given in a form of a representation. One approach proposed by (Finzi *et al.*, 2021) is to not work in the space of irreps but the space of polynomials of the input representation. This approach has the advantage of requiring little previous knowledge of the group. However it is also much less efficient than using irreps. One alternative way is to consider polynomials of the input representation, that are reducible and then compute the block diagonalisation to project down to irreps subspace. One can then work directly as polynomials in this subspace and compute Clebsch–Gordan coefficients numerically. We provide routines in `lie-nn` to carry out these operations from any given input representation.

A.8 Details of numerical experiments

A.8.1 Jet Tagging

Dataset The dataset (Butter *et al.*, 2019) was generated using a `Pythia`, `Delphes`, and `FastJet` (using cuts for the jet's kinematics on $\Delta\eta = 2$, $R = 0.8$) to simulate the response of the ATLAS detector at the Large Hadron Collider (LHC). The dataset is released under the "Creative Commons Attribution 4.0" license. The entire dataset contains 2 millions jets with a 60/20/20 for training, validation, and testing balanced splits.

Model The model uses **3 layers** of the G-MACE architecture to generate the Lorentz group equivariant representation of each jet. For the 1 particle basis, we use a product of radial features on the Minkowski distances, and $SO(1,3)$ spherical harmonics. The radial features are computing by passing a logarithmic radial basis as in (Bogatskiy *et al.*, 2022) into a $[64, 64, 64, 512]$ MLP using SiLU nonlinearities on the outputs of the hidden layers. The internal representations used are $(0,0)$ and $(1,1)$. We use 72 channels for each representation. For the embedding, and readout out, we use similar achitectures to LorentzNet.

Training Models were trained on an NVIDIA A100 GPU in single GPU training. Typical training time for the dataset is up to 72 hours. Models were trained with AMSGrad variant of Adam, with default parameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We used a learning rate of 0.0035 and a batch size of 64. The model was trained for 80 epochs with 2 epochs of linear learning rate warmup and followed by a phase of cosine annealing LR scheduling.

A.8.2 3D shape recognition

Dataset ModelNet10 (Wu *et al.*, 2015) is a synthetic 3D object point clouds dataset containing 4,899 pre-aligned shapes from 10 categories. The dataset is split into 3,991 (80%) shapes for training and 908 (20%) shapes for testing. We were unable to find a license.

Model The model uses a three-layer encoder architecture following the PointNet++ one. We use an encoder of the full point cloud into sub-point clouds of sizes $[1024, 256, 128]$. Each PointNet layer maps a point cloud of size N^t to one of size N^{t+1}. We compute the node features as the sum of the PointNet output and the MACE output,

$$h^{(t+1)} = \text{PointNet}(xyz^{(t)}, h^{(t)}) + \text{MACE}(xyz^{(t)}, h^{(t)}) \tag{30}$$

Training Models were trained on an NVIDIA A100 GPU in single GPU training. The typical training time for the dataset is up to 12 hours. Models were trained with the AMSGrad variant of Adam, with default parameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$.

A.9 Limitations and Future Work

The spectrum of potential applications of the present method is very large. In this paper, we focus on a subset of applications that have known benchmarks and baselines. A broader range of groups is implemented in the lie-nn library. Future work should focus on applying this architecture to tasks with domain-specific knowledge.