Rotation-Invariant Random Features Provide a Strong Baseline for Machine Learning on 3D Point Clouds

Anonymous authors
Paper under double-blind review

Abstract

Rotational invariance is a popular inductive bias used by many fields in machine learning, such as computer vision and machine learning for quantum chemistry. Rotation-invariant machine learning methods set the state of the art for many tasks, including molecular property prediction and 3D shape classification. These methods generally either rely on task-specific rotation-invariant features, or they use general-purpose deep neural networks which are complicated to design and train. We suggest a simple and general-purpose method for learning rotation-invariant functions of three-dimensional point cloud data using a random features approach. Specifically, we extend the random features method of Rahimi & Recht (2007) by deriving a version that is invariant to three-dimensional rotations and showing that it is fast to evaluate on point cloud data. We show through experiments that our method matches or outperforms the performance of general-purpose invariant architectures on standard molecular property prediction benchmark datasets QM7 and QM9. We also show that our method is competitive with other general-purpose invariant architectures on the ModelNet40 shape classification task. Finally, we show that our method is an order of magnitude faster at prediction time than competing kernel methods.

1 Introduction

Many common prediction tasks where the inputs are three-dimensional physical objects are known to be rotation-invariant; the ground-truth label does not change when the object is rotated. Building rotation invariance into machine learning models is an important inductive bias for such problems. The common intuition is that restricting the learning process to rotation-invariant models will remove any possibility of poor generalization performance due to rotations of test samples, and may improve sample efficiency by reducing the effective complexity of learned models. These ideas have inspired a line of research begun by Kondor et al. (2018); Cohen et al. (2018a) into building general-purpose deep neural network architectures that are invariant to rotations of their input. This line of work complements methods that use task-specific rotation-invariant features for problems at the interface of computational chemistry and machine learning (e.g., Montavon et al. (2012); Rupp et al. (2012)).

In this paper, we consider prediction problems that are *rotation-invariant*, that is, the ground-truth response $f^*(x) = y$ does not change when an arbitrary rotation is applied to the input x. We choose to model the input data x as a 3D point cloud, an unordered set of points in $\mathbb{R}^3$ possibly with accompanying labels. Common examples of rotation-invariant prediction problems on 3D point clouds include molecular property prediction, where the point cloud consists of the positions of a molecule's constituent atoms, and 3D shape classification, where the points are sampled from the surface of an object. The method we suggest and the methods we review in this paper are all rotation-invariant.

We propose a new method for learning rotation-invariant functions of point cloud data, which we call rotation-invariant random features. We extend the random features method of Rahimi & Recht (2007) by deriving a version that is invariant to three-dimensional rotations and showing that it is fast to evaluate on point cloud data. The rotation invariance property of our features is clear from their definition, and computing the features only requires two simple results from the representation theory of $SO(3)$. Despite its

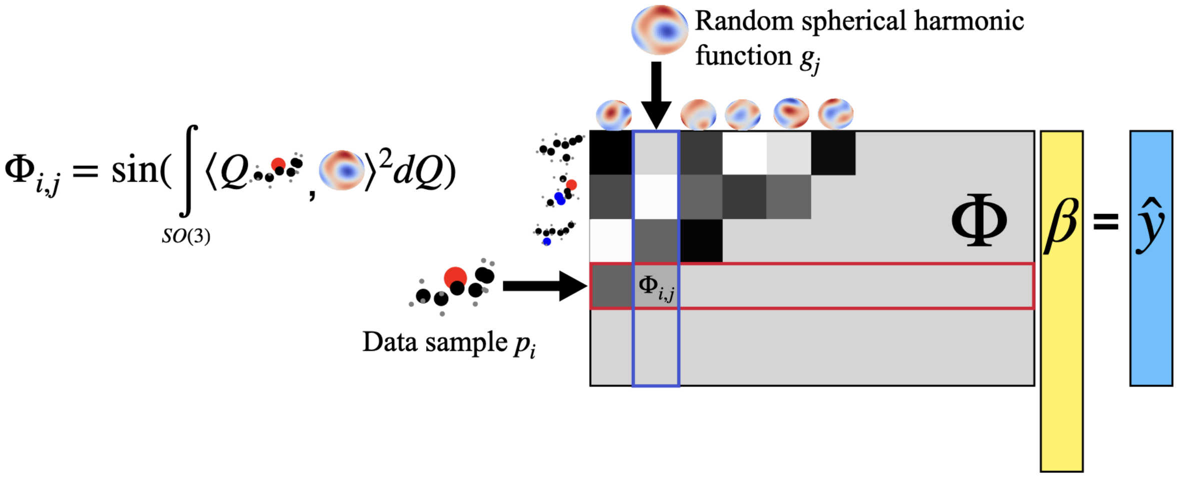

Random spherical harmonic function g_j
$\Phi_{i,j} = \sin(\int_{SO(3)} \langle Q \cdot , \cdot \rangle^2 dQ)$
Data sample p_i
Φ
$\Phi_{i,j}$
β
=
$\hat{y}$

Figure 1: An overview of the rotation-invariant random features method. First we construct a feature matrix Φ. The rows of Φ correspond to p_i, different samples in our training set, and the columns of Φ correspond to g_j, different random sums of spherical harmonic functions, which are defined in Equation (9). The entries of the feature matrix are rotation-invariant random features. To compute the random features, we analytically evaluate the integral over all possible rotations of the data sample p_i. We describe our method for evaluating the random features in Section 4.1. After constructing the feature matrix, we fit a set of linear weights β and make predictions by evaluating the linear model $\Phi\beta = \hat{y}$.

simplicity, our method achieves performance near state of the art on small-molecule energy regression and 3D shape classification. The strong performance on both tasks gives evidence that our method is general-purpose. An overview of our method can be found in Figure 1.

Other general-purpose methods for learning rotation-invariant functions use special deep neural networks. Methods in this line of work leverage results in the representation theory of $SO(3)$ to enforce rotational invariance (or equivariance) at each intermediate layer of the network, and examples include Cohen et al. (2018a); Kondor et al. (2018); Anderson et al. (2019); Thomas et al. (2018); Weiler et al. (2018); Poulenard et al. (2019). Throughout this paper, we refer to these methods as general-purpose invariant architectures. These methods are attractive because they do not rely on expert domain-specific knowledge to extract features, and they can learn richer classes of functions than methods specifying a fixed set of rotation-invariant features. However, the design of these methods requires expert knowledge in the representation theory of $SO(3)$. At a practical level, a significant amount of knowledge and engineering is required to implement and successfully train these networks. In comparison, training our model requires fitting a linear model, for which there are optimized solvers in most common scientific computing libraries. Finally, deep learning methods often under-perform classical methods such as kernels or linear models in small-data regimes when the number of training samples is limited.

Another approach to learning a rotation-invariant predictor from point cloud data is to extract a set of rotation-invariant features and feed those features into an expressive machine learning model to make predictions. A prototypical example of this approach comes from molecular energy regression and uses the Coulomb matrix as a set of rotation-invariant features (Rupp et al., 2012). The Coulomb matrix models the forces generated by electrostatic repulsion between a molecule's atoms. As the Coulomb matrix only depends on atomic charges and interatomic distances, it is clearly invariant to rotations of the original molecule. This general approach is powerful because any machine learning model can be used while retaining provable rotational invariance. Often, finding informative features to extract from the data is highly task-specific and requires expert background knowledge.

A third approach (Xiao et al., 2020) to learning rotation-invariant functions of three-dimensional point clouds is to preprocess the data by rotating it into a canonical alignment, defined by the principal components of the matrix of Euclidean coordinates of the point cloud. After the data samples are aligned, one can use any method and enjoy rotation-invariant predictions. This alignment preprocessing step has been shown to be very effective for shape classification tasks with densely-sampled point clouds, but to our knowledge has been untested in molecular property prediction.

Computational cost at prediction time is a growing concern for many machine learning methods. Using machine learning predictions for real-time system control places hard requirements on prediction latencies. For example, trigger algorithms in the ATLAS experiment (Collaboration, 2008) at the CERN Large Hadron Collider requires prediction latencies ranging from $2\mu s$ to $40ms$ at different stages of the event filtering hierarchy. Methods which extract a fixed set of rotation-invariant features are often able to quickly compute the features of a new sample, but the best-performing examples of these methods are kernels (Christensen et al., 2020) which have prediction-time complexity linear in the number of training samples. The deep neural network approaches which learn rotation-invariant features have rigid architectures that often involve many layers computed in serial, Fourier transforms, or the computation and storage of high-dimensional geometric tensors. We show that our method is an order of magnitude faster than state-of-the-art methods when predicting on new samples of molecular energy regression tasks. Due to the simplicity and generality of our design, we consider our method as a baseline against which one can measure the benefit of using expert domain knowledge for a given learning task.

1.1 Contributions

- We derive a rotation-invariant version of the standard random features approach presented in Rahimi & Recht (2007). By using simple ideas from the representation theory of $SO(3)$, our rotation-invariant random feature method is easy to describe and implement, and requires minimal expert knowledge in its design. (Section 4)
- We show with experiments that our rotation-invariant random feature method outperforms or matches the performance of general-purpose equivariant architectures optimized for small-molecule energy regression tasks. In particular, we outperform or match the performance of Spherical CNNs (Cohen et al., 2018a) on the QM7 dataset and Cormorant (Anderson et al., 2019) on the QM9 dataset. We compare our test errors with state-of-the-art methods on the QM9 dataset to quantify the benefit of designing task-specific rotation invariant architectures. (Section 5.1)
- We show the rotation-invariant random feature method is general-purpose by achieving low test errors using the same method on a different task, shape classification on the ModelNet40 benchmark dataset. (Section 5.2)
- We show that our method makes predictions an order of magnitude faster than kernel methods at test time. (Section 5.1.1)

2 Related Work

The organization of this section reflects the taxonomy of approaches to producing rotation-invariant models described above. First, we describe methods that extract rotation-invariant features using expert chemistry knowledge. We then describe general-purpose invariant architectures. Finally, we briefly survey the random features work that inspire our method.

Descriptors of atomic environments In the computational chemistry community, significant effort has been invested in using physical knowledge to generate "descriptors of atomic environments," or rotation-invariant feature embeddings of molecular configurations. One such feature embedding is the Coulomb matrix, used by Montavon et al. (2012). In Rupp et al. (2012), the sorted eigenvalues of the Coulomb matrix are used as input features. The sorted eigenvalues are invariant to rotations of the molecule and give a more compact representation than the full Coulomb matrix. Noticing that the Coulomb matrix can relatively faithfully represent a molecule with only pairwise atomic interactions has been an important idea in this community. Many methods have been developed to generate descriptions of atomic environments by decomposing the representation into contributions from all atomic pairs, and other higher-order approximations consider triplets and quadruplets of atoms and beyond (Christensen et al., 2020; Bartók et al., 2013; Kovács et al., 2021; Drautz, 2019).[1] The FCHL19 (Christensen et al., 2020) method creates a featurization that relies on chemical knowledge and is optimized for learning the energy of small organic molecules. The FCHL19 feature embedding depends on 2-body and 3-body terms. There are multiple different approaches, such as Smooth Overlap of Atomic Potentials (SOAP) (Bartók et al., 2013) and Atomic Cluster Expansion (ACE) (Drautz, 2019; Kovács et al., 2021) which first compute a rotation-invariant basis expansion of a chemical system, and then learn models in this basis expansion. In Appendix B, we show that our proposed method can be interpreted as using a highly-simplified set of ACE basis functions to learn a randomized nonlinear model.

Deep learning is an effective tool in computational chemistry as well. Task-specific deep neural network architectures for learning molecular properties of small organic molecules outperform other approaches on common energy learning benchmarks. Multiple methods (Schütt et al., 2018; Unke & Meuwly, 2019) design neural networks to take atomic charges and interatomic distances as input. These methods use highly-optimized neural network architectures, including specially-designed atomic interaction blocks and self-attention layers. OrbNet (Qiao et al., 2020), another neural network method, takes the output of a low-cost density functional theory calculation as rotation-invariant input features, and uses a message-passing graph neural network architecture.

General-purpose invariant architectures While many general-purpose invariant architectures operate directly on point clouds, the first examples of such architectures, Spherical CNNs (Cohen et al., 2018a) and Fully-Fourier Spherical CNNs (Kondor et al., 2018) operate on “spherical images”, or functions defined on the unit sphere. These networks define spherical convolutions between a spherical image and a filter. These convolutions are performed in Fourier space, and the input spherical image is transformed using a fast Fourier transform on the unit sphere. Much like a traditional convolutional neural network, these networks are comprised of multiple convolutional layers in series with nonlinearities and pooling layers in between. To accommodate multiple different types of 3D data, including 3D point clouds, these methods take a preprocessing step to render the information of a 3D object into a spherical image.

Other methods operate on point clouds directly. SPHNet (Poulenard et al., 2019) computes the spherical harmonic power spectrum of the input point cloud, which is a set of rotation-invariant features, and passes this through a series of point convolutional layers. Cormorant (Anderson et al., 2019) and Tensor Field Networks (Thomas et al., 2018) both design neural networks that at the first layer have activations corresponding to each point in the point cloud. All activations of these networks are so-called “spherical tensors,” which means they are equivariant to rotations of the input point cloud. To combine the spherical tensors at subsequent layers, Clebsch-Gordan products are used. Many other deep neural network architectures have been designed to operate on point clouds, but the majority have not been designed to respect rotational invariance. A survey of this literature can be found in Guo et al. (2021).

There are many ways to categorize the broad field of group-invariant deep neural network architectures, and we have chosen to form categories based on the input data type. Another helpful categorization is the distinction between *regular* group-CNNs and *steerable* group-CNNs introduced by Cohen et al. (2018b). In this distinction, *regular* group-CNNs form intermediate representations that are scalar functions of the sphere or the rotation group, and examples of this category are Cohen et al. (2018a); Kondor et al. (2018); Poulenard et al. (2019). *Steerable* group-CNNs form intermediate representations that are comprised of “spherical tensors”, and examples of this category are Anderson et al. (2019); Weiler et al. (2018); Thomas et al. (2018).

Random features Random Fourier features (Rahimi & Recht, 2007) are an efficient, randomized method of approximating common kernels. For kernel methods, prediction time scales linearly with the size of the dataset. Random Fourier feature methods require time linear in d, an approximation parameter. The approximation works by drawing a random vector ξ and defining a low-dimensional feature embedding $\varphi(x; \xi)$, called a random Fourier feature. The random Fourier features approximate a shift-invariant kernel

---

[1] Many general-purpose invariant methods follow this design as well. Cormorant (Anderson et al., 2019) has internal nodes which correspond to all pairs of atoms in the molecule, and Spherical CNNs (Cohen et al., 2018a) render spherical images from molecular point clouds using ideas from pairwise electrostatic (Coulombic) repulsion.

$k(x, x') = k(x - x')$ in the sense that the inner product of two random Fourier feature evaluations is an unbiased estimate of the kernel:

$$\mathbb{E}_{\xi}\left[\langle \varphi(x;\xi), \varphi(x;\xi)\rangle\right] = k(x, x')$$

This approximation holds when the random vectors ξ are drawn from the Fourier transform of the kernel k and the random Fourier features have the following functional form:

$$\varphi(x;\xi) = [\cos(\langle x, \xi\rangle), \quad \sin(\langle x, \xi\rangle)]^T$$

To reduce the variance of this approximation, one can draw d such random vectors $\{\xi_1, ..., \xi_d\}$ and concatenate the resulting random features. As a practical method, Rahimi & Recht (2007) proposes building a feature matrix $\Phi \in \mathbb{R}^{n\times d}$ by drawing d different random Fourier features $\varphi(\,\cdot\,;\xi_j)$ and evaluating the random Fourier features at each of the n samples x_i in the training dataset:

$$\Phi_{i,j} = \varphi(x_i; \xi_j)$$

Once the feature matrix is formed, a linear model is trained to fit a response vector y using regularized linear regression, such as ridge regression:

$$\arg\min_{\beta} \|\Phi\beta - y\|_2^2 + \lambda\|\beta\|_2^2 \tag{1}$$

Experiments in Rahimi & Recht (2007) show this is an effective method for fitting data, even when $d << n$. When $d << n$, the model can be evaluated much faster than the original kernel.

Follow-up work, including Rahimi & Recht (2008), suggested that interpreting random feature methods as kernel approximators is not necessary. This work uses random nonlinear features with varying functional forms, including random decision stumps and randomly-initialized sigmoid neurons $\sigma(\langle \xi, x\rangle)$. These random nonlinear features form effective models for diverse types of data. In Section 4, we introduce random features similar to those in Rahimi & Recht (2007). Our method does not approximate any explicit kernel, and it is designed to be invariant to any rotation of the input data, so we call our method rotation-invariant random features.

Finally, the notion of group-invariant random feature models also appears in Mei et al. (2021) as a technical tool to understand the sample complexity benefit of enforcing group invariances in overparameterized models. In this work, the invariances considered are transformations on one-dimensional signals that arise from cyclic and translation groups.

3 Rotational invariance and spherical harmonics

In this section, we will introduce rotational invariance and the spherical harmonics. In our method, we use the definitions presented in this section and only two simple facts about the spherical harmonics and rotations, which we list at the end of this section. We say a function f mapping a point cloud p to an element of the vector space $\mathcal{Y}$ is rotation-equivariant if there exists some group action on the vector space $\mathcal{Y}$ such that

$$f(Q \circ p) = Q \circ f(p) \quad \forall Q \in SO(3)$$

Intuitively, this means that when an input to f is rotated, f preserves the group structure and the output changes predictably. We say that a function f is rotation-invariant if the group action on the output vector space $\mathcal{Y}$ is the identity:

$$f(p) = f(Q \circ p) = f(\{Qx_1, Qx_2, \ldots, Qx_N\}) \quad \forall Q \in SO(3) \tag{2}$$

Finally, our goal is to learn a function over point clouds that does not change when any rotation is applied to the point cloud. Formally, given some distribution $\mathcal{D}$ generating pairs of data (p, y) our learning goal is to find

$$f^* = \arg\min_{f} \mathbb{E}_{(p,y)\sim\mathcal{D}}\left[l(f(p), y)\right]$$

where l is a classification or regression loss depending on the task and the minimization is over all functions that satisfy a rotational invariance constraint (2).

3.1 Spherical harmonics

When decomposing a periodic function $f : [0, 2\pi] \to \mathbb{R}$, a natural choice of basis functions for the decomposition is the complex exponentials e^{imx}. In this basis, $f(x)$ may be expressed as $f(x) = \sum_m a_m e^{imx}$ where $a_m = \langle f(x), e^{imx} \rangle$ When decomposing a function on the unit sphere S^2, a similar set of basis functions emerges, and they are called the spherical harmonics. The complex exponentials are indexed by a single parameter m, and because the spherical harmonics span a more complicated space of functions, they require two indices, ℓ and m. By convention, the indices are restricted to $\ell \geq 0$ and $-\ell \leq m \leq \ell$. A single spherical harmonic function $Y_m^{(\ell)}$ maps from the unit sphere S^2 to the complex plane $\mathbb{C}$. For a function $f(x) : S^2 \to \mathbb{R}$ we can compute an expansion in the spherical harmonic basis:

$$f(x) = \sum_{m,\ell} a_m^{(\ell)} Y_m^{(\ell)}(x)$$

$$a_m^{(\ell)} = \langle f(x), Y_m^{(\ell)}(x) \rangle$$

In designing our method, we use two simple facts about spherical harmonics:

- If one has evaluated the spherical harmonics of some original point $x \in S^2$ and wants to evaluate those spherical harmonics at a new rotated point Qx for some rotation matrix Q, the rotated evaluation is a linear combination of the un-rotated spherical harmonics at the same index ℓ:

  $$Y_m^{(\ell)}(Qx) = \sum_{m'=-\ell}^{\ell} Y_{m'}^{(\ell)}(x) D^{(\ell)}(Q)_{m,m'} \tag{3}$$

  Here $D^\ell(Q)$ is a Wigner-D matrix; it is a square matrix of size $(2\ell + 1 \times 2\ell + 1)$.

- Evaluating the inner product between two elements of the Wigner D-matrices is simple. In particular, when dQ is the uniform measure over $SO(3)$, we have the following expression:

  $$\int_{SO(3)} D^{(\ell_1)}(Q)_{m_1,k_1} D^{(\ell_2)}(Q)_{m_2,k_2} dQ = (-1)^{m_1 - k_1} \frac{8\pi^2}{2\ell + 1} \delta_{-m_1,m_2} \delta_{-k_1,k_2} \delta_{\ell_1,\ell_2} \tag{4}$$

  This is a corollary of Schur's lemma from group representation theory, and we discuss this fact in Appendix C.

4 Rotation-invariant random features

In our setting, an individual data sample is a point cloud $p_i = \{x_{i,1}, \ldots, x_{i,N_i}\}$ with points $x_{i,j} \in \mathbb{R}^3$. We model the data as a function: $p_i(x) = \sum_{j=1}^{N_i} \delta(x - x_{i,j})$ where $\delta(\cdot)$ is a delta function centered at 0. We can then use a functional version of the random feature method where our data p_i is a function, g is a random function drawn from some pre-specified distribution and $\langle \cdot, \cdot \rangle$ is the L^2 inner product:

$$\varphi(p_i; g) = \sin\left(\langle p_i, g \rangle\right) \tag{5}$$

We want the random feature to remain unchanged after rotating the point cloud, but (5) does not satisfy this. For general functions g, $\langle Q \circ p_i, g \rangle \neq \langle p_i, g \rangle$. We achieve rotational invariance by defining the following rotation-invariant random feature:

$$\varphi(p_i; g) = \sin\left(\int_{SO(3)} \langle Q \circ p_i, g \rangle^2 dQ \right) \tag{6}$$

We "symmetrize" the inner product by integrating over all possible orientations of the point cloud p_i, eliminating any dependence of $\phi(\cdot; w_j)$ on the data's initial orientation. Because $\phi(\cdot; w_j)$ does not depend on the initial orientation of the data, it will not change if this initial orientation changes, i.e., if p_i is rotated by Q. To build a regression model, we construct a feature matrix

$$\Phi_{i,j} = \varphi(p_i; g_j)$$

and fit a linear model to this feature matrix using ridge regression, as in (1).

4.1 Evaluating the random features

In this section, we describe how to efficiently evaluate the integral in (6). $SO(3)$ is a three-dimensional manifold and the integral has a nonlinear dependence on the data, so methods of evaluating this integral are not immediately clear. However, representation theory of $SO(3)$ renders this integral analytically tractable and efficiently computable. The main ideas used to evaluate this integral are exploiting the linearity of inner products and integration, and choosing a particular distribution of random functions g. First, we observe that because the rotated data function $Q \circ p_i$ is the sum of a few delta functions, we can expand the inner product as a sum of evaluations of the random function g:

$$\langle Q \circ p_i, g \rangle = \sum_{j=1}^{N_i} g(Qx_{i,j}) \tag{7}$$

Expanding the inner product and the quadratic that appear in (6) and using the linearity of the integral, we are left with a sum of simpler integrals:

$$\int_{SO(3)} \langle Q \circ p_i, g \rangle^2 dQ = \sum_{j_1, j_2=1}^{N_i} \int_{SO(3)} g(Qx_{i,j_1}) g(Qx_{i,j_2}) dQ \tag{8}$$

Next, we choose a particular distribution for g which allows for easy integration over $SO(3)$. We choose to decompose g as a sum of randomly-weighted spherical harmonics with maximum order L and K fixed radial functions:

$$g(x) = \sum_{k=1}^{K} \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} w_{m,k}^{(\ell)} Y_m^{(\ell)}(\hat{x}) R_k(\|x\|) \tag{9}$$

where $w_{m,k}^{(\ell)}$ are random weights, $\hat{x} = \frac{x}{\|x\|}$, and $R_k : \mathbb{R}_+ \to \mathbb{R}$ is a radial function. Two examples of such radial functions are shown in Figure 2.

By repeated application of the linearity of the integral and the two facts about the spherical harmonics mentioned in Section 3.1, we are able to write down a closed-form expression for the integral on the left-hand side of (8). We include these algebraic details in Appendix A. Computing this integral has relatively low complexity; it requires evaluating a table of spherical harmonics for each point $x_{i,j}$ up to maximum order L, and then performing a particular tensor contraction between the array of spherical harmonic evaluations and the array of random weights. This tensor contraction has complexity $O(N^2 L^3 K^2)$.

5 Experiments

We conduct multiple experiments comparing our method with other landmark rotation-invariant machine learning methods. We find that for predicting the atomization energy of small molecules, our method outperforms neural networks in the small dataset setting, and we have competitive test errors in the large dataset setting. Our method is an order of magnitude faster than competing kernel methods at test time. We also find that our method performs competitively on a completely different task, 3D shape classification. We perform these experiments to show that our method can be used as a fast, simple, and flexible baseline for rotation-invariant prediction problems on 3D point cloud data.

5.1 Small-Molecule Energy Regression

A common target for machine learning in chemistry is the prediction of a potential energy surface, which maps from 3D atomic configurations to the atomization energy of a molecule. Large standardized datasets such as QM7 Blum & Reymond (2009); Rupp et al. (2012) and QM9 Ruddigkeit et al. (2012); Ramakrishnan et al. (2014) offer a convenient way to test the performance of these machine learning models across a wide chemical space of small organic molecules. Both datasets contain the 3D atomic coordinates at equilibrium and corresponding internal energies. The 3D coordinates and molecular properties are obtained by costly quantum mechanical calculations, so there may be settings which require high-throughput screening where an approximate but fast machine learning model may provide an advantageous alternative to classical methods. To adapt our method to this task, we make two small changes.

Element-Type Encoding The rotation-invariant random feature method defined above works for general, unlabeled point clouds. However, in the chemistry datasets mentioned above, we are given more information than just the 3D coordinates of particles in the molecule: the particles are individual atoms, and we know their element type. Incorporating the element type of individual atoms in a machine learning method is crucial for accurate prediction. Different elements interact in quantitatively different ways, as predicted by the laws of gravitation and electrostatic repulsion, and they interact in qualitatively different ways due to their electron configurations.

We use an element-type encoding method inspired by FCHL19 Christensen et al. (2020) and ACE Kovács et al. (2021). To construct this element-type encoding, we look at the local view of a molecule created by centering the atomic coordinates at a given atom, separate the atoms into different point clouds for each element type, and compute one random feature per element type. We then repeat this procedure for all elements of a given type and sum their feature vectors.

Expressed mathematically, we are given a sample $p_i = \{x_{i,1}, \dots, x_{i,N}\}$ with charges $\{c_{i,1}, \dots, c_{i,N}\}$. Let $p_i^{(c_j)}$ be the collection of atoms with charge c_j, and let $p_i^{(c_j)} - x_{i,h}$ denote the point cloud of atoms in p_i with charge c_j centered at point $x_{i,h}$. In this notation, $p_i^{(c_j)} - x_{i,h}$ specifies a point cloud, which we can treat as unlabeled because all they all have the same charge. As before, a random feature is denoted $\varphi(\cdot; g_j)$. Then for all possible element-type pairs (c_1, c_2), we compute individual entries in our feature matrix

$$\Phi_{i,j'} = \sum_{h:\ c_{i_h} = c_1} \varphi\left(p_i^{(c_2)} - x_{i,h}; g_j\right)$$

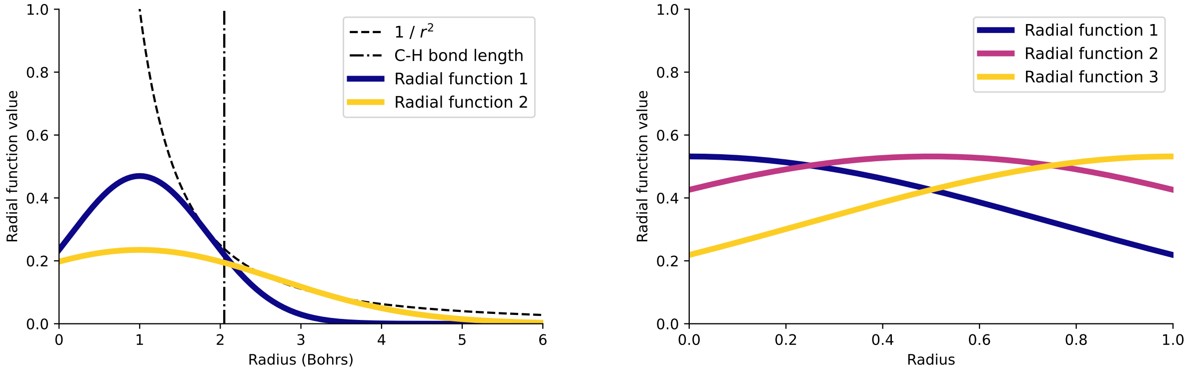
The column index j' depends on j, c_1, and c_2.

Radial functions We choose to parameterize our set of random functions as randomly-weighted spherical harmonics with fixed radial functions. The choice of radial functions appears as a design decision in many rotation-invariant machine learning methods. For example, radial functions appear as explicit design choices in Kovács et al. (2021); Christensen et al. (2020); Poulenard et al. (2019) and implicitly in the design of Cohen et al. (2018a). This design choice is often paramount to the empirical success of these methods. FCHL19 (Christensen et al., 2020) optimize their radial functions over multiple hyperparameters and carefully balance multiplicative terms including log-normal radial functions, polynomial decay, and soft cut-offs. The ACE models in Kovács et al. (2021) use a set of orthogonal polynomials defined over a carefully-chosen subset of the real line, a physically-motivated spatial transformation, and use extra ad-hoc radial functions that control the behavior of extremely nearby points. For our radial functions, we use two Gaussians, both centered at 1, with width parameters chosen so the full widths at half maximum are 2 and 4 respectively. Our radial functions are shown in Figure 2.

5.1.1 QM7 atomization energy regression

The QM7 dataset contains 7,165 small molecules with element types H, C, N, O, S and a maximum of 7 heavy elements. We compare with two lightweight methods, FCHL19 and Spherical CNNs, that show learning results on the QM7 dataset. We reproduce their original training methods and compare test errors in Table 1. Notably, our rotation-invariant random feature method has average errors half of those of Spherical CNNs while being faster to train. Our best-performing model uses 2,000 random features. With the element-type encoding described above, this corresponds to 50,000 trainable parameters.

In this experiment, we use a training dataset of 5,732 samples and a test set of size 1,433. For our method and FCHL19, we use 90% of the training set to train the models and the remaining 10% as a validation set for hyperparameter optimization. To train the FCHL19 models, we use the validation set to optimize over Gaussian kernel widths and L^2 regularization parameters. In our method, we search over the number of random features and L^2 regularization parameters. We solve our ridge regression problem by taking



(a) Radial functions for small-molecule energy regression

(b) Radial functions for 3D shape classification

Figure 2: Our rotation-invariant random feature method requires simple user-defined radial functions. Figure 2a shows the radial functions used in our small-molecule energy regression experiments. We include $\frac{1}{r^2}$ and the average Carbon-Hydrogen bond length to give context to the horizontal axis. Figure 2b shows the radial functions used in our 3D shape classification experiments. The 3D shapes in the benchmark dataset are normalized to fit inside the unit sphere.

Method	**Mean Absolute Error (eV)**	**Train Time (s)**	**Train Device**
Spherical CNNs	0.1405	512	single GPU
Random Features (Ours)	0.0641 ± 6.99e-05	121	24 CPU cores
FCHL19	0.0510	111	24 CPU cores

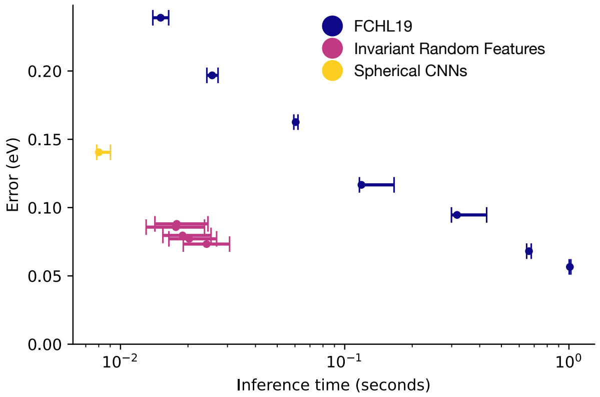
Table 1: Test error and training time on the QM7 dataset. Methods above the break are general-purpose, and FCHL19 is optimized for molecular systems. We report the mean absolute error on the test set for all methods and the standard error of the mean for our method.

an singular value decomposition of the random feature matrix, and then constructing a solution for each regularization level.

The space of small organic drug-like molecules is extremely large. The GDB-13 dataset Blum & Reymond (2009) contains nearly one billion reference molecules, even after filtering for molecule size and synthesis prospects. Most of the machine learning methods approximating potential energy surfaces are considered fast approximate replacements for costly quantum mechanical calculations. To screen the extremely large space of small organic drug-like molecules, these methods require high prediction throughput, which can be achieved by exploiting parallelism in the models' computational graphs and readily-available multicore CPU architectures. When parallelism inside the model is not available, the set of candidate molecules can be divided and data parallelism can be used to increase screening throughput. However, we consider another setting where experimental samples need to be analyzed in real time to make decisions about ongoing dynamic experiments. In this setting, prediction latency is paramount.

In the low-latency setting, our method can gracefully trade off prediction latency and prediction error by tuning the number of random features used in a given model. Importantly, all training samples are used to train the model. FCHL19 is a kernel method, so the time required to predict a new data point scales linearly with the number of training samples used. Thus, FCHL19 can only improve test latency at the cost of using fewer training samples and incurring higher test errors.

To explore the tradeoff between prediction latency and prediction error, we train random feature models and FCHL19 models of different sizes. We measure their prediction latencies and plot the results in Figure 3. We note that at similar error levels, FCHL19 models show prediction latencies that are an order of magnitude



Method	**Model Size**	**Inference time (s)**	**Error (eV)**
FCHL19	100	0.0151	0.2390
FCHL19	200	0.0256	0.1968
FCHL19	400	0.0604	0.1626
FCHL19	800	0.1188	0.1166
FCHL19	1600	0.3170	0.0946
FCHL19	3200	0.6635	0.0681
FCHL19	5000	1.0096	0.0566
Invariant Random Features	250	0.0177	0.0894
Invariant Random Features	500	0.0178	0.0861
Invariant Random Features	750	0.0189	0.0775
Invariant Random Features	1000	0.0203	0.0758
Invariant Random Features	2000	0.0242	0.0718
Spherical CNNs	N/A	0.0080	0.1405

Figure 3: Invariant Random Features are faster at prediction time than FCHL19 and more accurate than Spherical CNNs. For each method, we measure the latency of predicting each molecule on a held-out test set 5 times. In the figure, we plot the median prediction latencies with error bars spanning the 25th and 75th percentile measurements. The vertical axis of the figure is mean absolute error measured in eV on a held-out test set. The FCHL19 method is evaluated on 24 CPU cores using kernels with sample sizes 100, 200, 400, 800, 1600, 3200, and 5000. The rotation-invariant random features method is evaluated on 24 CPU cores with model sizes of 250, 500, 750, 1000, and 2000 random features. The Spherical CNNs method is evaluated on a single GPU.

slower than ours. Spherical CNNs exhibit low test latency because their method is implemented for a GPU architecture, but their test errors are high, and the neural network approach does not admit obvious ways to achieve a tradeoff between latency and test error. The prediction latencies of FCHL19 and our random feature models depend on the number of atoms in the molecule, and we show this dependence in Figure 4.

5.1.2 QM9 atomization energy regression

The QM9 dataset contains 133,885 molecules with up to 9 heavy atoms C, O, N, and F, as well as Hydrogen atoms. This is a larger and more complex dataset than QM7, and because there is more training data, neural network methods perform well on this benchmark. For this dataset, we follow Anderson et al. (2019); Gilmer et al. (2017) by constructing atomization energy as the difference between the internal energy at $0K$ and the thermochemical energy of a molecule's constituent atoms. We train a large-scale random feature model by taking 100,000 training samples from QM9 and generating 10,000 random features. We select an L^2 regularization parameter *a priori* by observing optimal regularization parameters from smaller scale experiments on a proper subset of the QM9 training data.

We compare the performance of our model with the reported errors of FCHL19 and other neural network methods. All methods considered in this comparison enforce rotational invariance in their predictions. The results of this comparison are shown in Table 2. Our rotation-invariant random feature method matches the performance of Cormorant, and it provides a very strong baseline against which we can quantify the effect of expert chemistry knowledge or neural network design. OrbNet uses a graph neural network architecture, and their inputs are carefully-chosen from the results of density functional theory calculations in a rotation-invariant basis. PhysNet uses an architecture heavily inspired by that of SCHNet, and this iteration in model design resulted in almost a 50% reduction in test error.

Training models on large subsets of the QM9 dataset is difficult. FCHL19 requires 27 hours to construct a complete kernel matrix of size (133,855 × 133,855) on a compute node with 24 processors. The 27 hours does not include the time required to find the model's linear weights. The neural network methods require long training sequences on GPUs. Both Cormorant and PhysNet report training for 48 hours on a GPU.

Method	**Model Type**	**Mean Absolute Error (eV)**	**Input Features**
Cormorant	General-purpose rotation-invariant architecture	0.022	• Charges • Atomic locations
Random Features (Ours)	Ridge Regression	0.022 ± 2.42e-06	• Charges • Rotation-invariant random features
FCHL19	Kernel Ridge Regression	0.011	• Charges • Interatomic distances • 3-body angles
SCHNet	Neural Network with atomic interaction blocks	0.014	• Charges • Interatomic distances
PhysNet	Neural Network with attention layers	0.008	• Charges • Interatomic distances
OrbNet	Message-Passing GNN	0.005	• Mean-field DFT calculations

Table 2: Comparing large-scale models trained on QM9. FCHL19 (Christensen et al., 2020) was trained on 75,000 samples from QM9 and the other methods were trained on 100,000 samples. Methods in the first group are general-purpose invariant architectures, and methods in the second group are specialized to molecular property prediction. We report the mean absolute error on the test set for all methods and the standard error of the mean for our method.

When using a large sample size, our method is similarly difficult to train. We are able to construct a matrix of random features of size (100,000 × 250,000) in 27 minutes on a machine with 24 processors, but the matrix is highly ill-conditioned, and using an iterative method to approximately solve the ridge regression problem requires 76.5 hours. Using an iterative method introduces approximation error which we did not encounter when performing experiments on the QM7 dataset. We attribute some of the performance gap between our method and FCHL19 to this approximation error. We attempted to improve the training time of our method by using a principal components regression approach, where we approximated the solution to the ridge regression problem using the top singular vectors of the random feature matrix. Surprisingly, this led to poor performance. We believe the conditioning of our problem is caused by the choice of element-type encoding, and we leave finding a new element-type encoding that produces well-conditioned feature matrices for future work.

5.2 Shape Classification

To show that our method is general-purpose, we test our same method on a completely different task, 3D shape classification. In particular, we consider multiclass classification on the ModelNet40 benchmark dataset. The ModelNet40 benchmark dataset Zhirong Wu et al. (2015) is set of computer-generated 3D models of common shapes, such as mugs, tables, and airplanes. There are 9,843 training examples and 2,468 test examples spread across 40 different classes. The shape objects are specified by 3D triangular meshes, which define a (possibly disconnected) object surface. To generate a point cloud from individual objects in this dataset, one must choose a sampling strategy and sample points from the surface of the object. We use the dataset generated by Qi et al. (2017), which samples 1,024 points on the mesh faces uniformly at random. The point clouds are then centered at the origin and scaled to fit inside the unit sphere.

We solve the multiclass classification problem with multinomial logistic regression. More specifically, we optimize the binary cross entropy loss with L2 regularization to learn a set of linear weights for each of the 40 classes in the dataset. At test time, we evaluate the 40 different linear models and predict by choosing the class with the highest prediction score. We also slightly change the definition of our data function; for a point cloud $S_i = \{x_{i,1}, ..., x_{i,N_i}\}$, we use a normalized data function $S_i(x) = \frac{1}{N_i} \sum_{j=1}^{N_i} \delta(x - x_{i,j})$ to eliminate any

Method	**Test Accuracy**	**Train Time (s)**	**Train Device**
SPHNet	0.863	6,485	single GPU
Ours	0.653	9,664	24 CPU cores

Table 3: Training and test results on the ModelNet40 shape classification benchmark task.

dependence on the number of points sampled from the surface of the shape objects. For this setting, we use radial functions that are three Gaussian bumps with centers at $0, 0.5$, and 1, with width $\sigma = 0.75$.

We compare our method with another rotation-invariant method, SPHNet Poulenard et al. (2019). SPHNet is a multilayer point convolutional neural network that enforces rotational invariance by computing the spherical harmonic power spectrum of the input point cloud. SPHNet's rotation-invariant method also requires a choice of radial functions, and they use two Gaussians with different centers and vary the width of the Gaussians at different layers of their network. The results of our comparison are in Table 3. Our method does not achieve near state of the art results, but we conclude it provides a strong baseline on this challenging multiclass problem.

6 Discussion and future work

Our method is a simple, flexible, and competitive baseline for rotationally-invariant prediction problems on 3D point clouds. Rotation-invariant random features are simple to explain and implement, and using them in varied prediction tasks requires a minimal amount of design choices. Our method does not achieve state-of-the-art accuracy on any prediction task, but it provides a competitive baseline that allows us to begin to quantify the effect of neural network models and expert chemistry knowledge in methods that enforce rotational invariance.

Our method shows promise in settings where low prediction latency is desired. In high-throughput experiments, such as the ATLAS experiment at the CERN Large Hadron Collider Collaboration (2008), data is generated at such a high velocity that real-time decisions must be made whether to save or discard individual samples. For this type of initial screening task, we imagine our low-latency and flexible prediction method will be an attractive candidate. Once trained, our method has a shallow computational graph which includes only simple algebraic functions, evaluations of trigonometric functions, and complex exponentiation.

One can also interpret our work as an initial investigation into the use of random Fourier feature methods for approximating common kernel learning methods in computational chemistry. Kernel methods are well-studied and highly performant in machine learning for computational chemistry; examples of such methods include Rupp et al. (2012); Montavon et al. (2012); Bartók et al. (2013); Christensen et al. (2020). It is a natural question to ask whether these methods can benefit from fast inference times while maintaining high test accuracy when approximated by random features. Our method does not implement an exact approximation to any of the kernel methods above, but our experiments show promise in this area. Our test errors are near those of FCHL19 on the QM7 dataset, and versions of our model using only 10% of the optimal number of parameters have reasonable test errors. However, when training our model on the QM9 dataset, we have seen that the particular element-type encoding we have used creates ill-conditioned feature matrices and makes optimization difficult. To fully realize the potential of random feature methods in accelerating kernel learning for computational chemistry, a new element-type encoding is needed.

References

Brandon Anderson, Truong Son Hy, and Risi Kondor. Cormorant: Covariant molecular neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/03573b32b2746e6e8ca98b9123f2249b-Paper.pdf`.

Albert P. Bartók, Risi Kondor, and Gábor Csányi. On representing chemical environments. *Physical Review B*, 87(18):184115, May 2013. ISSN 1098-0121, 1550-235X. doi: 10.1103/PhysRevB.87.184115. URL `https://link.aps.org/doi/10.1103/PhysRevB.87.184115`.

Lorenz C. Blum and Jean-Louis Reymond. 970 Million Druglike Small Molecules for Virtual Screening in the Chemical Universe Database GDB-13. *Journal of the American Chemical Society*, 131(25):8732–8733, July 2009. ISSN 0002-7863, 1520-5126. doi: 10.1021/ja902302h. URL `https://pubs.acs.org/doi/10.1021/ja902302h`.

Anders S Christensen, Lars A Bratholm, and Felix A Faber. FCHL revisited: Faster and more accurate quantum machine learning. *The Journal of Chemical Physics*, pp. 16, 2020.

Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical CNNs. *arXiv:1801.10130 [cs, stat]*, February 2018a. URL `http://arxiv.org/abs/1801.10130`. arXiv: 1801.10130.

Taco S. Cohen, Mario Geiger, and Maurice Weiler. Intertwiners between induced representations (with applications to the theory of equivariant neural networks), 2018b. URL `https://arxiv.org/abs/1803.10743`.

ATLAS Collaboration. The atlas experiment at the cern large hadron collider. *Journal of Instrumentation*, 3 (08):S08003, August 2008. ISSN 1748-0221. doi: 10.1088/1748-0221/3/08/S08003. URL `https://dx.doi.org/10.1088/1748-0221/3/08/S08003`.

Ralf Drautz. Atomic cluster expansion for accurate and transferable interatomic potentials. *Physical Review B*, 99(1):014104, January 2019. ISSN 2469-9950, 2469-9969. doi: 10.1103/PhysRevB.99.014104. URL `https://link.aps.org/doi/10.1103/PhysRevB.99.014104`.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1263–1272. JMLR.org, 2017.

Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12): 4338–4364, December 2021. ISSN 1939-3539. doi: 10.1109/TPAMI.2020.3005434. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch-Gordan Nets: a Fully Fourier Space Spherical Convolutional Neural Network, November 2018. URL `http://arxiv.org/abs/1806.09231`. arXiv:1806.09231 [cs, stat].

Dávid Péter Kovács, Cas van der Oord, Jiri Kucera, Alice E. A. Allen, Daniel J. Cole, Christoph Ortner, and Gábor Csányi. Linear Atomic Cluster Expansion Force Fields for Organic Molecules: Beyond RMSE. *Journal of Chemical Theory and Computation*, 17(12):7696–7711, December 2021. ISSN 1549-9618, 1549-9626. doi: 10.1021/acs.jctc.1c00647. URL `https://pubs.acs.org/doi/10.1021/acs.jctc.1c00647`.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Learning with invariances in random features and kernel models. In Mikhail Belkin and Samory Kpotufe (eds.), *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pp. 3351–3418. PMLR, 15–19 Aug 2021. URL `https://proceedings.mlr.press/v134/mei21a.html`.

Grégoire Montavon, Katja Hansen, Siamac Fazli, Matthias Rupp, Franziska Biegler, Andreas Ziehe, Alexandre Tkatchenko, O. Anatole von Lilienfeld, and Klaus-Robert Müller. Learning invariant representations of molecules for atomization energy prediction. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 440–448, Red Hook, NY, USA, 2012. Curran Associates Inc.

Adrien Poulenard, Marie-Julie Rakotosaona, Yann Ponty, and Maks Ovsjanikov. Effective Rotation-Invariant Point CNN with Spherical Harmonics Kernels. In *2019 International Conference on 3D Vision (3DV)*, pp. 47–56, September 2019. doi: 10.1109/3DV.2019.00015. ISSN: 2475-7888.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, April 2017. URL `http://arxiv.org/abs/1612.00593`. arXiv:1612.00593 [cs].

Zhuoran Qiao, Matthew Welborn, and Animashree Anandkumar. OrbNet: Deep learning for quantum chemistry using symmetry-adapted atomic-orbital features. *The Journal*, pp. 11, 2020.

Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL `https://proceedings.neurips.cc/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf`.

Ali Rahimi and Benjamin Recht. Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (eds.), *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. URL `https://proceedings.neurips.cc/paper/2008/file/0efe32849d230d7f53049ddc4a4b0c60-Paper.pdf`.

Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1(1):140022, December 2014. ISSN 2052-4463. doi: 10.1038/sdata.2014.22. URL `http://www.nature.com/articles/sdata201422`.

Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum, and Jean-Louis Reymond. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *Journal of Chemical Information and Modeling*, 52(11):2864–2875, November 2012. ISSN 1549-9596, 1549-960X. doi: 10.1021/ci300415d. URL `https://pubs.acs.org/doi/10.1021/ci300415d`.

Matthias Rupp, Alexandre Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Fast and Accurate Modeling of Molecular Atomization Energies with Machine Learning. *Physical Review Letters*, 108(5):058301, January 2012. ISSN 0031-9007, 1079-7114. doi: 10.1103/PhysRevLett.108.058301. URL `https://link.aps.org/doi/10.1103/PhysRevLett.108.058301`.

K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller. SchNet – A deep learning architecture for molecules and materials. *The Journal of Chemical Physics*, 148(24):241722, June 2018. ISSN 0021-9606, 1089-7690. doi: 10.1063/1.5019779. URL `http://aip.scitation.org/doi/10.1063/1.5019779`.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds, May 2018. URL `http://arxiv.org/abs/1802.08219`. Number: arXiv:1802.08219 arXiv:1802.08219 [cs].

William J. Thompson. *Angular momentum: an illustrated guide to rotational symmetries for physical systems*. Wiley, New York, 1994. ISBN 978-0-471-55264-2.

Oliver T. Unke and Markus Meuwly. PhysNet: A Neural Network for Predicting Energies, Forces, Dipole Moments, and Partial Charges. *Journal of Chemical Theory and Computation*, 15(6):3678–3693, June 2019. ISSN 1549-9618, 1549-9626. doi: 10.1021/acs.jctc.9b00181. URL `https://pubs.acs.org/doi/10.1021/acs.jctc.9b00181`.

Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 10402–10413, Red Hook, NY, USA, 2018. Curran Associates Inc.

Zelin Xiao, Hongxin Lin, Renjie Li, Lishuai Geng, Hongyang Chao, and Shengyong Ding. Endowing Deep 3d Models With Rotation Invariance Based On Principal Component Analysis. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, London, United Kingdom, July 2020. IEEE. ISBN 978-1-72811-331-9. doi: 10.1109/ICME46284.2020.9102947. URL `https://ieeexplore.ieee.org/document/9102947/`.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1912–1920, Boston, MA, USA, June 2015. IEEE. ISBN 978-1-4673-6964-0. doi: 10.1109/CVPR.2015.7298801. URL `https://ieeexplore.ieee.org/document/7298801/`.

A Full Integration Details

We wish to solve the following integral, which appears in the definition of rotation-invariant random features (6):

$$\int_{SO(3)} \langle Q \circ p_i, g \rangle^2 dQ \tag{10}$$

From Section 4.1, we know this integral can decompose into

$$\int_{SO(3)} \langle Q \circ p_i, g \rangle^2 dQ = \sum_{j_1, j_2 = 1}^{N_i} \int_{SO(3)} g(Qx_{i,j_1}) g(Qx_{i,j_2}) dQ \tag{11}$$

And we also have chosen a particular functional form for our random function g:

$$g(x) = \sum_{k=1}^{K} \sum_{\ell=0}^{L} \sum_{m=-\ell}^{\ell} w_{m,k}^{(\ell)} Y_m^{(\ell)}(\hat{x}) R_k(\|x\|) \tag{12}$$

Repeated application of the linearity of integration gives us:

$$\int_{SO(3)} \langle Q \circ p_i, g \rangle^2 dQ = \tag{13}$$

$$= \int_{SO(3)} g(Qx_{i,j_1}) g(Qx_{i,j_2}) dQ \tag{14}$$

$$= \sum_{j_1, j_2 = 1}^{N_i} \sum_{k_1, k_2 = 1}^{K} \sum_{\ell_1, \ell_2 = 0}^{L} \sum_{m_1 = -\ell_1}^{\ell_1} \sum_{m_2 = -\ell_2}^{\ell_2} w_{m_1, k_1}^{(\ell_1)} w_{m_2, k_2}^{(\ell_2)} R_{k_1}(\|x_{i,j_1}\|) R_{k_2}(\|x_{i,j_2}\|) \int_{SO(3)} Y_{m_1}^{(\ell_1)}(Q\hat{x}_{i,j_1}) Y_{m_2}^{(\ell_2)}(Q\hat{x}_{i,j_2}) dQ \tag{15}$$

$$= \sum_{j_1, j_2 = 1}^{N_i} \sum_{k_1, k_2 = 1}^{K} \sum_{\ell_1, \ell_2 = 0}^{L} \sum_{m_1 = -\ell_1}^{\ell_1} \sum_{m_2 = -\ell_2}^{\ell_2} w_{m_1, k_1}^{(\ell_1)} w_{m_2, k_2}^{(\ell_2)} R_{k_1}(\|x_{i,j_1}\|) R_{k_2}(\|x_{i,j_2}\|) \int_{SO(3)} Y_{m_1}^{(\ell_1)}(Q\hat{x}_{i,j_1}) Y_{m_2}^{(\ell_2)}(Q\hat{x}_{i,j_2}) dQ \tag{16}$$

We can now focus on the integral in (15), and apply the rotation rule for spherical harmonics introduced in Section 3.1.

$$\int_{SO(3)} Y_{m_1}^{(\ell_1)}(Q\hat{x}_{i,j_1}) Y_{m_2}^{(\ell_2)}(Q\hat{x}_{i,j_2}) dQ \tag{17}$$

$$= \int_{SO(3)} \sum_{m_1' = -\ell_1}^{\ell_1} Y_{m_1'}^{(\ell_1)}(\hat{x}_{i,j_1}) D^{(\ell_1)}(Q)_{m_1, m_1'} \sum_{m_2' = -\ell_2}^{\ell_2} Y_{m_2'}^{(\ell_2)}(\hat{x}_{i,j_2}) D^{(\ell_2)}(Q)_{m_2, m_2'} dQ \tag{18}$$

$$= \sum_{m_1' = -\ell_1}^{\ell_1} \sum_{m_2' = -\ell_2}^{\ell_2} Y_{m_1'}^{(\ell_1)}(\hat{x}_{i,j_1}) Y_{m_2'}^{(\ell_2)}(\hat{x}_{i,j_2}) \int_{SO(3)} D^{(\ell_1)}(Q)_{m_1, m_1'} D^{(\ell_2)}(Q)_{m_2, m_2'} dQ \tag{19}$$

$$\tag{20}$$

At this point, we are able to apply the integration rule for elements of Wigner-D matrices introduced in Section 3.1.

$$\int_{SO(3)} Y^{(\ell_1)}_{m_1}(Q\hat{x}_{i,j_1}) Y^{(\ell_2)}_{m_2}(Q\hat{x}_{i,j_2}) dQ \tag{21}$$

$$= \sum_{m_1'=-\ell_1}^{\ell_1} \sum_{m_2'=-\ell_2}^{\ell_2} Y^{(\ell_1)}_{m_1'}(\hat{x}_{i,j_1}) Y^{(\ell_2)}_{m_2'}(\hat{x}_{i,j_2}) (-1)^{m_1 - m_1'} \frac{8\pi^2}{2\ell+1} \delta_{-m_1,m_2} \delta_{-m_1',m_2'} \delta_{\ell_1,\ell_2} \tag{22}$$

$$= \delta_{\ell_1,\ell_2} \delta_{-m_1,m_2} \frac{8\pi^2}{2\ell+1} \sum_{m_1'=-\ell_1}^{\ell_1} (-1)^{m_1-m_1'} Y^{(\ell_1)}_{m_1'}(\hat{x}_{i,j_1}) Y^{(\ell_1)}_{-m_1'}(\hat{x}_{i,j_2}) \tag{23}$$

To put it all together, we substitute (23) into (15), and we see many terms drop out of the sums:

$$\sum_{j_1,j_2=1}^{N_i} \sum_{k_1,k_2=0}^{K} \sum_{\ell_1,\ell_2=0}^{L} \sum_{m_1=-\ell_1}^{\ell_1} \sum_{m_2=-\ell_2}^{\ell_2} w^{(\ell_1)}_{m_1,k_1} w^{(\ell_2)}_{m_2,k_2} R_{k_1}(\|x_{i,j_1}\|) R_{k_2}(\|x_{i,j_2}\|) \int_{SO(3)} Y^{(\ell_1)}_{m_1}(Q\hat{x}_{i,j_1}) Y^{(\ell_2)}_{m_2}(Q\hat{x}_{i,j_2}) dQ \tag{24}$$

$$= \sum_{j_1,j_2=1}^{N_i} \sum_{k_1,k_2=0}^{K} \sum_{\ell_1,\ell_2=0}^{L} \sum_{m_1=-\ell_1}^{\ell_1} \sum_{m_2=-\ell_2}^{\ell_2} w^{(\ell_1)}_{m_1,k_1} w^{(\ell_2)}_{m_2,k_2} R_{k_1}(\|x_{i,j_1}\|) R_{k_2}(\|x_{i,j_2}\|) \delta_{\ell_1,\ell_2} \delta_{-m_1,m_2} \frac{8\pi^2}{2\ell+1} \sum_{m_1'=-\ell_1}^{\ell_1} (-1)^{m_1-m_1'} Y^{(\ell_1)}_{m_1'}(\hat{x}_{i,j_1}) Y^{(\ell_1)}_{-m_1'}(\hat{x}_{i,j_2}) \tag{25}$$

$$= \sum_{j_1,j_2=1}^{N_i} \sum_{k_1,k_2=0}^{K} \sum_{\ell_1=0}^{L} \sum_{m_1=-\ell_1}^{\ell_1} w^{(\ell_1)}_{m_1,k_1} w^{(\ell_1)}_{-m_1,k_2} R_{k_1}(\|x_{i,j_1}\|) R_{k_2}(\|x_{i,j_2}\|) \frac{8\pi^2}{2\ell+1} \sum_{m_1'=-\ell_1}^{\ell_1} (-1)^{m_1-m_1'} Y^{(\ell_1)}_{m_1'}(\hat{x}_{i,j_1}) Y^{(\ell_1)}_{-m_1'}(\hat{x}_{i,j_2}) \tag{26}$$

Once the spherical harmonics and radial functions are evaluated, this sum has $O(N^2KL^3)$ terms.

A.1 Basis Expansion

When implementing (26) to compute rotationally-invariant random features, one can see that a lot of computational effort can be re-used between different evaluations of random features. A table of spherical harmonic evaluations for all the points $\{\hat{x}_{i,1}, \dots \hat{x}_{i,N_i}\}$ can be pre-computed once and re-used. Also, by re-arranging the order of summation in (26), we arrive at this expression:

$$\int_{SO(3)} \langle Q \circ p_i, g \rangle^2 dQ = \sum_{k_1,k_2=1}^{K} \sum_{\ell_1=0}^{L} \sum_{m_1=-\ell_1}^{\ell_1} w^{(\ell_1)}_{m_1,k_1} w^{(\ell_1)}_{-m_1,k_2} \sum_{j_1,j_2=1}^{N_i} R_{k_1}(\|x_{i,j_1}\|) R_{k_2}(\|x_{i,j_2}\|) \frac{8\pi^2}{2\ell+1} \sum_{m_1'=-\ell_1}^{\ell_1} (-1)^{m_1-m_1'} Y^{(\ell_1)}_{m_1'}(\hat{x}_{i,j_1}) Y^{(\ell_1)}_{-m_1'}(\hat{x}_{i,j_2}) \tag{27}$$

Pre-computing everything after the line break allows for the re-use of a large amount of computational work between different random feature evaluations. We can interpret this pre-computation as an expansion of our point cloud in a rotationally-invariant basis B, which depends on the choice of radial function. Appendix B relates this basis expansion to the Atomic Cluster Expansion method.

$$B[\ell, m, k_1, k_2] \quad = \quad \sum_{j_1,j_2=1}^{N_i} R_{k_1}(\|x_{i,j_1}\|)R_{k_2}(\|x_{i,j_2}\|)\frac{8\pi^2}{2\ell+1}\sum_{m'=-\ell}^{\ell}(-1)^{m-m'}Y^{(\ell)}_{m'}(\hat{x}_{i,j_1})Y^{(\ell)}_{-m'}(\hat{x}_{i,j_2}) \tag{28}$$

$$B[\ell, m, k_1, k_2] = \sum_{j_1,j_2=1}^{N_i} R_{k_1}(\|x_{i,j_1}\|)R_{k_2}(\|x_{i,j_2}\|)\int_{SO(3)} Y^{(\ell)}_{m}(Q\hat{x}_{i,j_1})Y^{(\ell)}_{-m}(Q\hat{x}_{i,j_2})dQ \tag{29}$$

In our implementation we precompute this B tensor once for each sample, and then we perform a tensor contraction with the random weights:

$$\int_{SO(3)} \langle Q \circ p_i, g\rangle^2 dQ = \sum_{k_1,k_2=1}^{K}\sum_{\ell_1=0}^{L}\sum_{m_1=-\ell_1}^{\ell_1} w^{(\ell_1)}_{m_1,k_1} w^{(\ell_1)}_{-m_1,k_2} B[\ell_1, m_1, k_1, k_2] \tag{30}$$

B Connection between our method and randomized ACE methods

B.1 Overview of the ACE basis

The atomic cluster expansion (ACE) method builds a rotationally-invariant basis for point clouds that are an extension of our pre-computed basis expansion described in Appendix A.1. In the atomic cluster expansion (ACE) method, a preliminary basis $A_{k,\ell,m}$ is formed by projecting a point cloud function $p_i(x) = \sum_j \delta(x - x_{i_j})$ with a single-particle basis function $\varphi_{k,\ell,m}(x)$:

$$\begin{aligned} \varphi_{k,\ell,m}(x) &= R_k(\|x\|)Y^{(\ell)}_m(x) && (31)\\ A_{k,\ell,m}(p_i) &= \langle \varphi_{k,\ell,m}, p_i\rangle && (32)\\ &= \sum_j \varphi_{k,\ell,m}(x_{i_j}) && (33) \end{aligned}$$

where R_k is a set of radial functions. In Kovács et al. (2021), this set of radial functions is defined by taking a nonlinear radial transformation. Then the radial functions are a set of orthogonal polynomials defined on the transformed radii. This basis set is augmented with auxiliary basis functions to ensure the potential energy of two atoms at extremely nearby points diverges to infinity.

The $A_{k,\ell,m}$ basis is a first step, but it only models single particles at a time. To model pairwise particle interactions, or higher-order interactions, the A basis is extended by taking tensor products. First, a "correlation order" N is chosen. Then, the A basis is extended via tensor products:

$$A_{\underline{k},\underline{\ell},\underline{m}}(p_i) = \prod_{\alpha=1}^{N} A_{k_\alpha,\ell_\alpha,m_\alpha}(p_i) \tag{34}$$

where $\underline{k} = (k_\alpha)_{\alpha=1}^N$, $\underline{\ell} = (\ell_\alpha)_{\alpha=1}^N$, and $\underline{m} = (m_\alpha)_{\alpha=1}^N$ are multi-indices.

Finally, to ensure invariance with respect to permutations, reflections, and rotations of the point cloud, the ACE method forms a Haar measure dg over $O(3)$ and computes a symmetrized version of the A basis:

$$B_{\underline{k},\underline{\ell},\underline{m}}(p_i) = \int_{O(3)} A_{\underline{k},\underline{\ell},\underline{m}}(H \circ p_i)dH \tag{35}$$

The reference Drautz (2019) gives a detailed description of how this integral is performed. Applications of representation theory give concise descriptions of which multi-indices $\underline{k}, \underline{\ell}, \underline{m}$ remain after performing this integral; many are identically zero.

Method	**Test Error (eV)**
Spherical CNNs	0.1405
FCHL19	0.0510
Ours (Random Features)	0.0619
Ours (Linear Model of B basis)	0.1476

Table 4: Reported are test errors on the QM7 dataset for Spherical CNNs, FCHL19, and two variants of our method. The first variant is the full rotation-invariant random features model. The second variant is a linear model of our basis expansion. We note that while our random feature model is the second-best performing model on this benchmark, our linear model is the worst-performing model.

B.2 Connection to our method

If one chooses to compute an ACE basis with correlation order $N = 2$, the resulting basis is very similar to our pre-computed basis expansion in (29).

$$
\begin{aligned}
B_{\underline{k},\underline{\ell},\underline{m}}(p_i) &= \int_{O(3)} A_{\underline{k},\underline{\ell},\underline{m}}(H \circ p_i) dH && (36)\\
&= \int_{O(3)} A_{k_1,\ell_1,m_1}(H \circ p_i) A_{k_2,\ell_2,m_2}(H \circ p_i) dH && (37)\\
&= \int_{O(3)} \left(\sum_{j_1} \varphi_{k_1,\ell_1,m_1}(H x_{i_{j_1}}) \right) \left(\sum_{j_2} \varphi_{k_2,\ell_2,m_2}(H x_{i_{j_2}}) \right) dH && (38)\\
&= \int_{O(3)} \left(\sum_{j_1} R_{k_1}(\|x_{i_{j_1}}\|) Y^{(\ell_1)}_{m_1}(H \hat{x}_{i_{j_1}}) \right) \left(\sum_{j_2} R_{k_2}(\|x_{i_{j_2}}\|) Y^{(\ell_2)}_{m_2}(H \hat{x}_{i_{j_2}}) \right) dH && (39)\\
&= \sum_{j_1,j_2} R_{k_1}(\|x_{i_{j_1}}\|) R_{k_2}(\|x_{i_{j_2}}\|) \int_{O(3)} Y^{(\ell_1)}_{m_1}(H \hat{x}_{i_{j_1}}) Y^{(\ell_2)}_{m_2}(H \hat{x}_{i_{j_2}}) dH && (40)
\end{aligned}
$$

The remaining difference between our basis expansion in (29) and the ACE basis expansion with correlation order $N = 2$ is the ACE basis integrates over all of $O(3)$, while our method only enforces invariance with respect to $SO(3) \subset O(3)$.

So one can describe our random features as random nonlinear projections of a simplified version of the ACE basis with correlation order $N = 2$ using extremely simple radial functions.

B.3 Benefit of Random Features

We conduct the following experiment to quantify the benefit of using random nonlinear features over fitting a linear model in our simplified version of the ACE basis. Using the same hyperparameters as in our energy regression experiments, we compute our basis expansion for all the samples in the QM7 dataset. Using the same train/validation/test split, we perform ridge regression by fitting a linear model in our basis B with L^2 regularization. We use the validation set to identify the best-performing model and report errors on the test set in Table 4. We note that the best-performing linear model has more than twice the error of the best performing random features model.

C Representation Theory of the 3D Rotation Group

C.1 Conjugation of Wigner-D Matrices

In the following, we derive a formula for conjugating an element of a Wigner-D matrix. First, we use definition 6.44 in Thompson (1994) for the Wigner-D matrices:

$$D^{(\ell)}_{m',m}(\alpha,\beta,\gamma) = e^{-im'\alpha} d^{(\ell)}_{m',m}(\beta) e^{-im\gamma} \tag{41}$$

Here, (α,β,γ) are Euler angles parameterizing a rotation, and $d^{(\ell)}_{m',m}$ is an element of a Wigner-d (small d) matrix. We omit the exact definition of $d^{(\ell)}_{m',m}$, but we note the following symmetry properties:

$$d^{(\ell)}_{-m',-m}(\beta) = d^{(\ell)}_{m',m}(-\beta) \tag{42}$$

$$d^{(\ell)}_{m',m}(-\beta) = d^{(\ell)}_{m,m'}(\beta) \tag{43}$$

$$d^{(\ell)}_{m,m'}(\beta) = (-1)^{m-m'} d^{(\ell)}_{m',m}(\beta) \tag{44}$$

The relationship between the Wigner d matrix elements allows us to derive a formula for conjugating an element of a Wigner D matrix:

$$\begin{aligned}
D^{(\ell)}_{m',m}(\alpha,\beta,\gamma)^* &= e^{im'\alpha} d^{(\ell)}_{m',m}(\beta)^* e^{im\gamma} && (45)\\
&= e^{-i(-m')\alpha} d^{(\ell)}_{m',m}(\beta) e^{-i(-m)\gamma} && (46)\\
&= e^{-i(-m')\alpha} d^{(\ell)}_{-m',-m}(-\beta) e^{-i(-m)\gamma} && (47)\\
&= e^{-i(-m')\alpha} d^{(\ell)}_{-m,-m'}(\beta) e^{-i(-m)\gamma} && (48)\\
&= e^{-i(-m')\alpha} (-1)^{-m+m'} d^{(\ell)}_{-m',-m}(\beta) e^{-i(-m)\gamma} && (49)\\
&= (-1)^{m'-m} D^{(\ell)}_{-m',-m}(\alpha,\beta,\gamma) && (50)
\end{aligned}$$

C.2 Orthogonality Relations

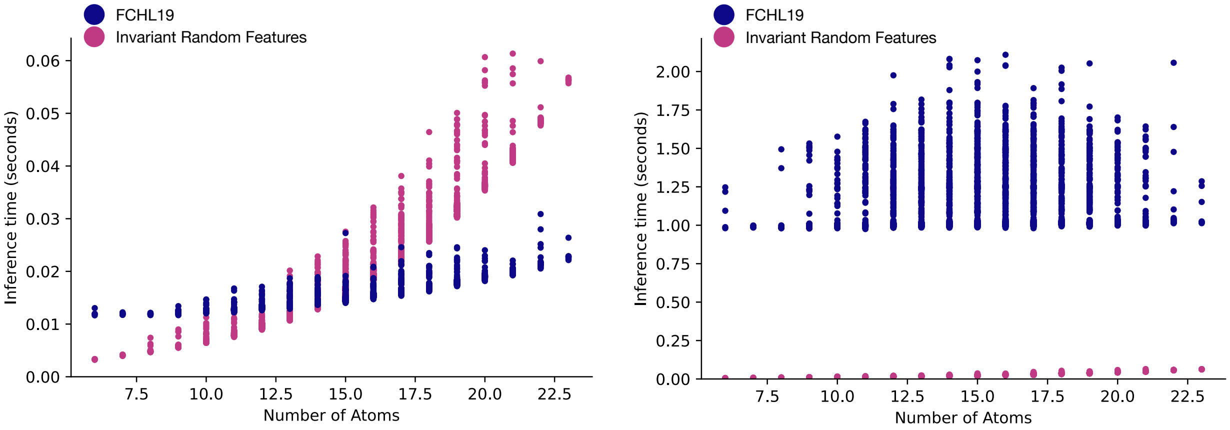
We use the following orthogonality relationship between elements of Wigner D matrices:

$$\int_{SO(3)} D^{(\ell_1)}(Q)^*_{m_1,k_1} D^{(\ell_2)}(Q)_{m_2,k_2} dQ = \frac{8\pi^2}{2\ell+1}\delta_{m_1,m_2}\delta_{k_1,k_2}\delta_{\ell_1,\ell_2} \tag{51}$$

This is a well-known fact about the Wigner-D matrices. For a proof, one can derive this relationship from Schur orthogonality relations, which are a consequence of Schur's lemma. A direct calculation can be found in section 6.4.2 of Thompson (1994). Combining the two facts from above, we arrive at the following identity, which we use to compute our rotation-invariant random features.

$$\int_{SO(3)} D^{(\ell_1)}(Q)_{m_1,k_1} D^{(\ell_2)}(Q)_{m_2,k_2} dQ = (-1)^{m_1-k_1}\frac{8\pi^2}{2\ell+1}\delta_{-m_1,m_2}\delta_{-k_1,k_2}\delta_{\ell_1,\ell_2} \tag{52}$$

D Supplementary Figures



(a) Latency for small models

(b) Latency for large models

Figure 4: The number of atoms in a molecule affects the prediction latency for both FCHL19 and the invariant random features method. In Figure 4a, we show the prediction latencies for the smallest models tested, corresponding to 250 random features for our method and 200 training samples for FCHL19. For this model size, a quadratic scaling in the number of atoms largely determines the prediction latency. Figure 4b shows that for larger models (2,000 random features for our method and 5,000 training samples for FCHL19), the random features method is still in a regime where prediction latency is determined by the number of atoms, while FCHL19's prediction latency is dominated by the number of inner products required to evaluate the kernel.