Permutation-equivariant neural networks for power spectrum estimation

TL;DR: A story about neural networks that don't care about the order of things.

A floral motivation

Due to a completely believable sequence of events,
your life depends on procuring a beautiful bouquet of flowers.

Naturally, you download the well-known MNIST database of bouquet beauty.
And as luck would have it, there is even a neural network trained on the data!

You pick out a dozen flowers and inquire:

> *Query:* "What is the beauty of this bouquet?"
> *Result:* "InvalidArgumentError: In[0] mismatch In[1] shape: 12 vs. 3"

Fine. The network is only trained on bouquets of three flowers. You ask:

> *Query:* "What is the beauty of: a peony, a daffodil, and a begonia?"
> *Result:* "7"

Conveniently suspicious of the result, you follow up with:

> *Query:* "What is the beauty of: a begonia, a daffodil, and a peony?"
> *Result:* "5"

Weird. You certainly expected that the model wouldn't be perfect, but you'd at least hope that it would give the same answer for the same bouquet! *The order shouldn't matter*.

As you just so happen to be a highly enthusiastic AI resident, you pull up Colab and start developing a permutation-equivariant neural network. After all... your life depends on it!

An introduction by analogy

The remarkable success of convolutional neural networks for image classification is arguably due to their extensive parameter sharing via convolution filters. This is justified by the natural assumption of translation invariance — essentially the requirement that each "patch" of pixels is treated "the same". For example, consider building a linear layer that extracts four features (at each pixel) from the images in the (also well-known) MNIST database of handwritten digits. Without any such assumption, one requires $\sim 2.5 \cdot 10^6$ parameters for this layer.[1] In contrast, a CNN layer with 3×3 filters has only 40.[2]

[1] $(28^2 \text{ input pixels}) \times (28^2 \text{ output pixels} \times 4 \text{ features}) + (28^2 \times 4 \text{ biases}) = (2461760 \text{ total parameters})$.
[2] $(4 \text{ filters}) \times (3^2 \text{ pixels per filter}) + (4 \text{ biases}) = (40 \text{ total parameters})$.

Analogously, we would like the network to treat each "shuffling" of our input "the same". For example, with our input of 512 (unsorted) things (each a vector of length 2), one would naïvely require $\sim 2 \cdot 10^6$ parameters.[3] By enforcing the desired permutation invariance, we require instead only 20.[4]

At first glance, sets may appear too dis*order*ed for practical use. However, their tangible applications are more ubiquitous than one might suspect. Several examples (in no particular order) are:

- analyzing a collection of samples from a distribution,
- choosing a plate of food from a buffet,
- selecting a group to complete a project,
- packing belongings for a trip,
- etc [2, 6, 1]

An analogous development

A permutation-equivariant layer

First, a baby problem. Consider all linear maps from the space of N scalar inputs to N scalar outputs. Bundling these into two vectors of length N, x^{in}_n and x^{out}_n, any such linear map is described by an $N \times N$ matrix $w_{nn'}$ (thus requiring N^2 parameters):

$$x^{\text{out}}_n = w_{nn'} x^{\text{in}}_{n'} \tag{1}$$

where we are using Einstein summation convention.[5]

Next, a toddler problem. What if, when we shuffle the order of the scalar inputs, we require the scalar outputs to move in the same way? That is, we only want linear maps that are *equivariant to permutations*. An unavoidable consequence of this requirement is that, instead of N^2 parameters, we have just 2; each scalar output may only depend on:

i. its corresponding scalar input, and

ii. the mean[6] of all scalar inputs.

Thus, the only viable academic offspring are of the form:

$$x^{\text{out}}_n = \underbrace{w^{\text{self}} x^{\text{in}}_n}_{\text{i}} + \underbrace{w^{\text{mean}} \left(\tfrac{1}{N} 1_{n'} x^{\text{in}}_{n'} \right) 1_n}_{\text{ii}} \tag{2}$$

where w^{self} and w^{mean} are scalar weights, and 1_* is the all-ones vector of length N.

[3] $(512 \times 2$ input scalars$) \times (512 \times 4$ output scalars$) + (512 \times 4$ biases$) = (2099200$ total parameters$)$.

[4] $(2$ input dimensions$) \times (4$ output dimensions$) \times (2$ equivariant operations$) + (4$ biases$) = (20$ total parameters$)$.

[5] Essentially, when objects are multiplied together, there is an implicit sum over repeated subscripts. For example, the expression $\underline{\mathrm{y}} = \underline{\underline{\mathrm{M}}} \cdot \underline{\mathrm{x}}$ is short for $y_i = \sum_j m_{ij} x_j$. Instead, we use the more compact and generalizable $y_i = m_{ij} x_j$.

[6] Taking the sum is also possible.

For neural networks of this sort, puberty involves the promotion of each scalar input to a vector of length K', each scalar output to a vector of length K, and the two scalar weights to matrices of dimension $K \times K'$. Upon graduation, we also allow for K scalar biases and apply a pointwise nonlinearity ξ. You beam with pride at their final yearbook photo, which just so happens to bear a remarkable resemblance to a general permutation-equivariant layer:

$$x^{\text{out}}_{nk} = \xi\Big(w^{\text{self}}_{kk'} x^{\text{in}}_{nk'} + w^{\text{mean}}_{kk'} \Big(\tfrac{1}{N} 1_{n'} x^{\text{in}}_{n'k'}\Big) 1_n + b_k 1_n\Big). \tag{3}$$

A permutation-invariant neural network

As we watch our neural network grow by sequentially stacking such permutation-equivariant layers, it eventually comes to a decision that all such creatures must make: to remain equivariant, or to become invariant.

If their life objective function was to classify the *individual* elements that make up the set (eg, how important is each flower to the beauty of a given bouquet), then the neural network should remain *equivariant*. The final output of the network should indeed have dimensions $N \times K_{\text{fin}}$, providing a vector of length K_{fin} for each of the N elements (in the appropriate order, of course).

However, more holistic neural networks that are concerned with properties of the *entire* set (eg, predicting the beauty of a flower bouquet), should become *invariant* by aggregating over the N outputs (eg, by taking the mean). The final output is instead a vector of length K_{fin}, providing a result for the entire set (which does not change upon shuffling the order of the N inputs).

A *power*ful task

Just as your neural network is reaching adulthood, Captain Plot Device™ swoops in and offers you the most beautiful bouquet in all the world in exchange for your help on a different problem with the same structure.[7]

Specifically, you are to predict the (truncated) power spectrum of a random periodic function $f(x)$, given N random samples $(x_n, y_n = f(x_n))$. The functions are of the form:

$$f(x) = \sum_{k=1}^{k_{\text{max}}} \sqrt{p_k} \cos(kx + \phi_k), \quad x \in [0, 2\pi]. \tag{4}$$

For each frequency k, the phase ϕ_k is sampled uniformly on the interval $[0, 2\pi]$ and the power p_k is sampled from an exponential distribution with mean $1/k^2$. As $k_{\text{max}} \to \infty$, $f(x)$ converges to periodic Brownian motion (see Fig. 1).

[7] See CL 348675714 for the code associated with this mini project. The code is rather flexible, allowing for easy change of input functions, predicted part of the spectrum, and network architecture.

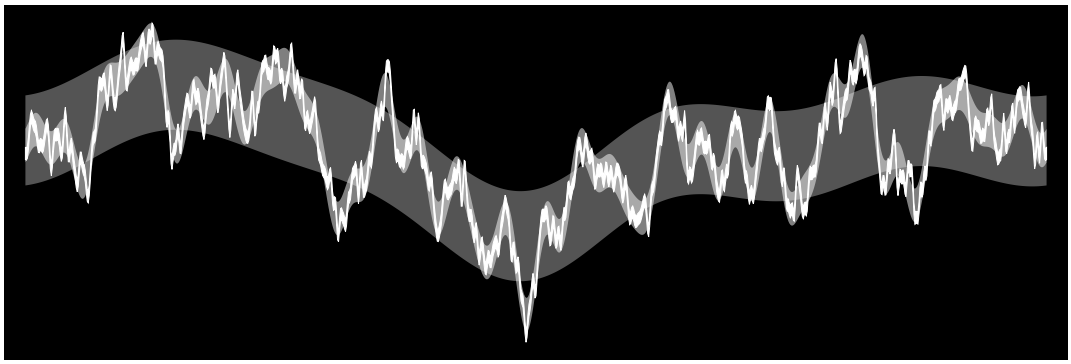

Fig. 1: **An instantiation of periodic Brownian motion.** Also known as a Brownian bridge, this process is essentially the continuous limit of a random walk that returns to its starting position after a given interval (here $[0, 2\pi]$). We trained permutation-equivariant[8] networks to predict the (truncated) power spectrum of functions sampled via this process. For each such function $f(x)$, we sampled N points $(x_n)_{n\in[N]}$ iid and uniformly in $[0, 2\pi]$. We provided the set of $(x_n, y_n = f(x_n))$ pairs as input to the networks. Shown here are functions with $k_{\text{max}} = 4$, 32, and 256.

Some obligatory experimental details

We consider neural networks with D hidden permutation-equivariant layers (the "*D*epth"), each with K_{hide} scalars for each thing in the set (the "*K*width"). Thus, the weight matrices $w^{\text{self}}_{kk'}$ and $w^{\text{mean}}_{kk'}$ in equation (3) are of dimensions $K_{\text{hide}} \times K_{\text{hide}}$.[9] All layers use the ReLU nonlinearity: $\xi(\cdot) = \max(\cdot, 0)$.

For the final layer, we first average all N things in the set (resulting in a vector of length K_{hide}). This passes through a dense layer to yield the K_{fin} scalars[10] that estimate the log power spectrum up to frequency K_{fin}.[11]

Using the TensorFlow implementation of the Adam algorithm [3] (with default parameters), we aim to minimize the mean squared error in $\log p_k$ for $k \in [K_{\text{fin}}]$. Sets contain 512 pairs, and batches contain 256 sets.

[8] Well... technically permutation-*invariant* neural networks, since changing the order of the input must not change the final output. However, as all but our last layer is permutation-equivariant (and honestly, "equivariant" has a nicer ring to it), we follow others [2, 4] and allow for this mild abuse of nomenclature.

[9] With the exception of the first layer, whose weight matrices $w^{\text{self}}_{kk'}$ and $w^{\text{mean}}_{kk'}$ both have dimensions $2 \times K_{\text{hide}}$, as the things in the input sets are (x, y) pairs.

[10] Asymptotically, the number of trainable parameters is $\Theta\Big((2K_{\text{début}} + K_{\text{fin}} + 1)K_{\text{hide}} + 2(D-1)K^2_{\text{hide}}\Big)$.

[11] In the actual code, the neural network exponentiated before outputting so as to give the estimate of the actual power spectrum.

A quadruplet of triptych results

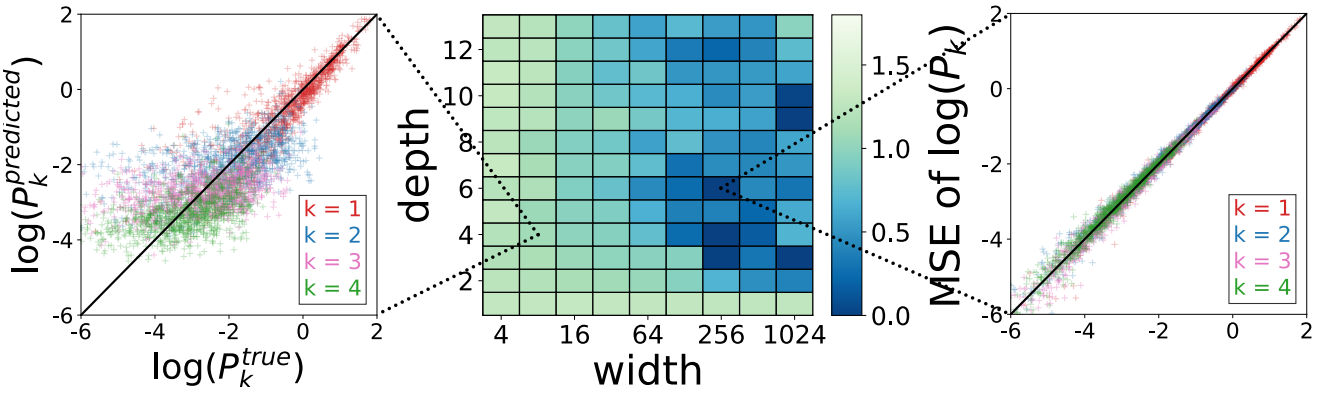

Fig. 2: **Permutation-equivariant neural networks need both depth and width.** Using equation (4) with $k_{\max} = 4$, we generate sets of size $N = 512$, and train neural networks to predict the power in the first 4 frequencies ($K_{\text{fin}} = 4$). We take 5 trained networks and measure the MSE in log power predictions on 1024 test sets.

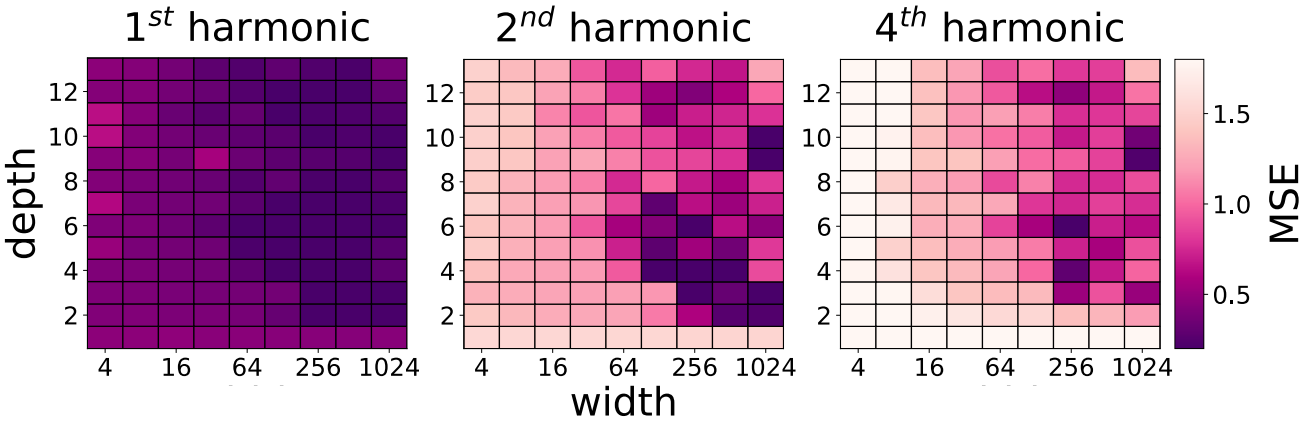

Fig. 3: **Our choice of loss function reflects the learning in *all* frequencies.** Using the same networks as in Fig. 2, we plot the MSE for the log power at each frequencies $k = 1,\ 2,$ and 4. Unsurprisingly, we find that it is harder to learn the (smaller) powers at higher frequencies.

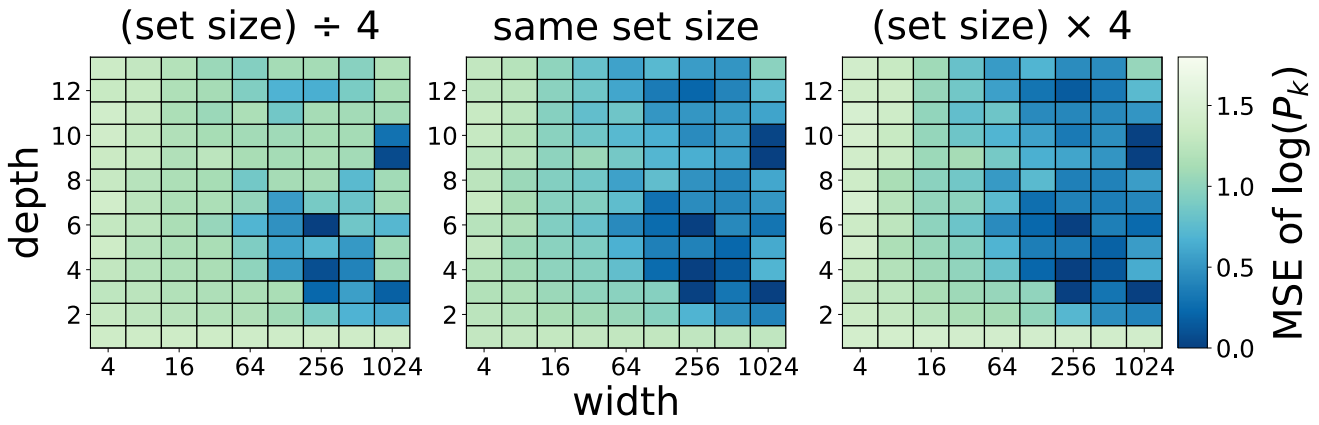

Fig. 4: **Permutation-equivariant neural networks naturally handle sets of different sizes.** Just as permutation-invariance is oblivious to the ordering of the input set, nor does it care about its size.[12] Using the same networks as in Fig. 2. we test their predictions for sets of different sizes. All test sets were generated from new realizations of random functions.

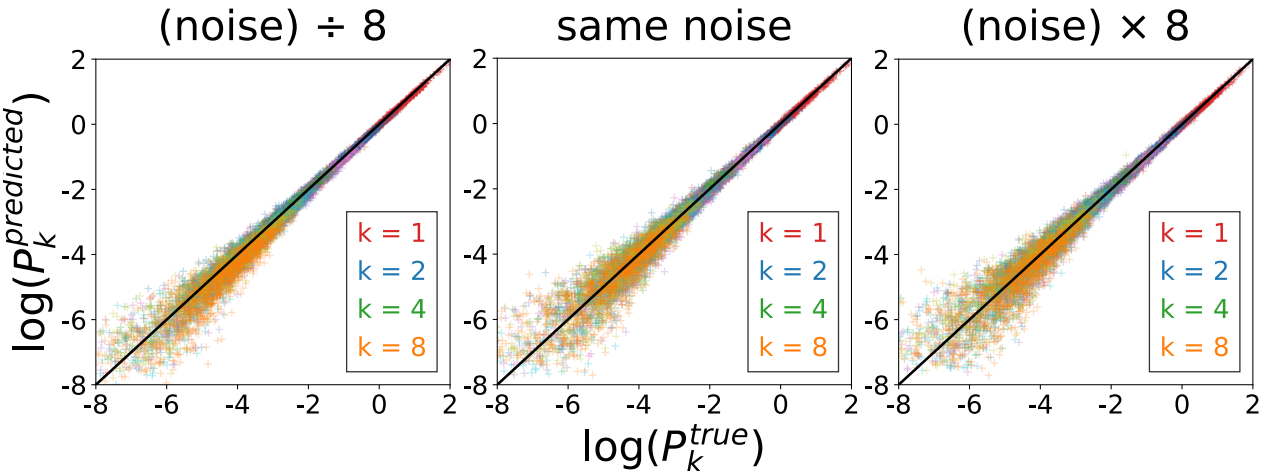

Fig. 5: **Our permutation-equivariant neural networks are robust to noise.** For this figure, we push our neural networks a bit harder, training them to predict a larger part of the power spectrum ($K_{\text{fin}} = 8$). Moreover, we train them on noisier functions ($k_{\text{max}} = 64$). Most of them did not make it. Here, we test the predictions of one of the most adept architectures (depth $= 4$, width $= 2048$). At test time, we also consider sets generated by functions with: less noise ($k_{\text{max}} = 8$, *left*), and more noise ($k_{\text{max}} = 512$, *right*). In all cases, the network performed as desired.[13]

[12] This is true when the mean is used in the aggregation operations.

[13] This is not surprising, as the noise at $k_{\text{max}} = 64$ is $\sim 90\%$ of its asymptotic ($k \to \infty$) value. Nonetheless, we are still proud of this network; distinguishing between frequencies $k = 7$ and 8 is indeed notably more challenging than between $k = 3$ and 4.

A misplaced appendix

Incorporating multiple symmetries

In this mini project, we focused attention on incorporating the permutation symmetry[14] into the structure of a neural network. As it just so happens, our particular choice of toy problem has another symmetry: the power spectrum should be invariant to translation of the x coordinate (modulo 2π).[15] In fact, the x_i don't really live in $[0, 2\pi]$, but rather on the unit circle S^1 — we should have called them θ_i.

This viewpoint is similar to that in [5]. They consider an unordered set of positions in $\mathbb{R}^3$ and their features, requiring that the output be invariant to global rotations and translations of the coordinates, in addition to permutations of the points themselves.

An inspiring coda

"Deep Mixtures"

What would the "thermodynamic" limit ($N \to \infty$) of permutation-equivariance look like? Suppose you were interested in the beauty of an infinite garden.

Query: "What is the beauty of this infinite bouquet?"
Result: "ResourceExhaustedError: maximum recursion depth exceeded."

Maybe you shouldn't always hit "remind me later" when it asks to upgrade to OS-$\aleph_0$.
You try your best with your archaic restriction to the finite:

Query: "What is the beauty of about 2000000 begonias, 3000000 daffodils, and 1000000 peonies?"
Result: 

Concerned that your query may take too long,
you open another terminal and ask the question this section is trying to get to:

Query: "What is the beauty of $\frac{1}{3}$ begonias, $\frac{1}{2}$ daffodils, and $\frac{1}{6}$ peonies?"
Result: "This feature is still under development.
Please contact gecia@ for further information."

[14] Ie, the action of the symmetric group $\mathbb{S}_N$.
[15] Ie, the action of the rotation group $SO(2)$.

Acknowledgements

Many thanks to my onboarding mentor, Kieran Murphy (murphyka@), for restoring my faith in advisors many times over. And, to my husband, Lee M. Gunderson, whose sharp wit and creative spirit is remarkably intoxicating.

References

[1] J. Gordon, D. Lopez-Paz, M. Baroni, and D. Bouchacourt. Permutation equivariant models for compositional generalization in language. In *ICRL*, 2019.
[2] N. Guttenberg, N. Virgo, O. Witkowski, H. Aoki, and R. Kanai. Permutation-equivariant neural networks applied to dynamics prediction. *preprint arXiv:1612.04530*, 2016.
[3] D. P. Kingma and J. Ba. Adam: a method for stochastic optimization. *preprint arXiv:1412.6980*, 2014.
[4] E. H. Thiede, T. S. Hy, and R. Kondor. The general theory of permutation equivarant neural networks and higher order graph variational encoders. *preprint arXiv:2004.03990*, 2020.
[5] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *preprint arXiv:1802.08219*, 2018.
[6] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *NeurIPS*, pages 3391–3401, 2017.