# Symmetries, Safety, and Self-Supervision

Peter Sorrenson

November 1, 2022

Heidelberg Colaboratory for Image Processing
University of Heidelberg

ML4Jets 2022

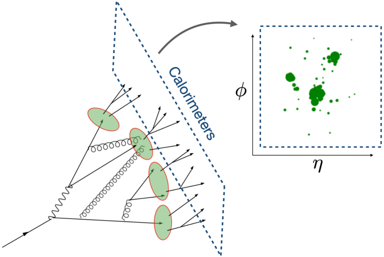
*Symmetries, Safety, and Self-Supervision*, hep-ph/2108.04253

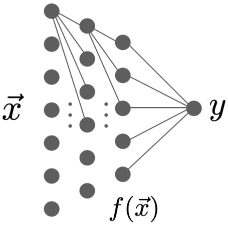
Barry M. Dillon, Gregor Kasieczka, Hans Olischlager, Tilman Plehn, Peter Sorrenson, and Lorenz Vogel

Universität Heidelberg
Zukunft. Seit 1386.

# Top-tagging with machine-learning

Neural network maps kinematical data to a predicted label (supervised)




- simulations provide training data $\{\vec{x}_i\}$ and truth-labels $\{y'_i\}$
- neural network is optimised to minimise a loss function

$$\mathcal{L}_i = y'_i \log(y_i) + (1 - y'_i) \log(1 - y_i)$$

- loss function is minimised when QCD and top jets are well-separated in $y$
- predicted label is a new observable used to tag top-jets

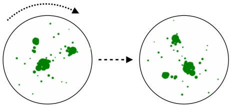
# Learning physical quantities

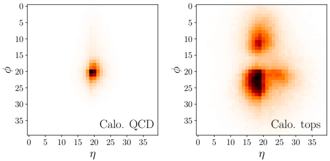
Neural networks ⇒ inductive bias

i.e. implicit assumptions made by the network on mapping input → output

→ neural nets are not invariant to physical symmetries in data

→ we typically try to solve this through 'pre-processing'





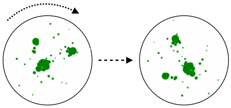
# Learning physical quantities

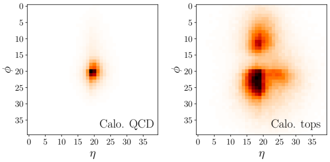
Neural networks ⇒ inductive bias

i.e. implicit assumptions made by the network on mapping input → output

→ neural nets are not invariant to physical symmetries in data

→ we typically try to solve this through 'pre-processing'






**Our goal:** control the training to ensure we learn physical quantities

→ rotational & translation invariant, permutation invariant, IRC safe

→ deep neural networks can never be completely interpretable

... but we can place limits on what they can learn

# Optimising observables / representations

**How?**

Reframe the definition of our observables as an optimisation problem to be solved with machine-learning

What do we fundamentally want from observables?

1. invariance to certain transformations / augmentations of the jets
2. discriminative within the space of jets

# Optimising observables / representations

**How?**

Reframe the definition of our observables as an optimisation problem to be solved with machine-learning

What do we fundamentally want from observables?

1. invariance to certain transformations / augmentations of the jets
2. discriminative within the space of jets

⋆ Contrastive-learning → JetCLR (SimCLR, Google Brain, Hinton et al)
map raw jet data to a new representation / observables

# Optimising observables / representations

How?

Reframe the definition of our observables as an optimisation problem to be solved with machine-learning

What do we fundamentally want from observables?

1. invariance to certain transformations / augmentations of the jets
2. discriminative within the space of jets

- ⋆ Contrastive-learning → JetCLR (SimCLR, Google Brain, Hinton et al)

  map raw jet data to a new representation / observables
- ⋆ **Self-supervision**

  neural networks are optimised using pseudo-labels, not truth labels

  → independent of signal-types

  → can run directly on expt. data

# Contrastive learning of jet representations

hep-ph/2108.04253, 'Symmetries, Safety, and Self-Supervision'
B. M. Dillon, G. Kasieczka, H. Olischlager, T. Plehn, P. Sorrenson, and L. Vogel

Dataset: mixture of top-jets and QCD-jets

From the dataset of jets $\{x_i\}$ define:

- positive-pairs: $\{(x_i, x_i')\}$ where $x_i'$ is an augmented version of $x_i$
  related by augmentation
- negative-pairs: $\{(x_i, x_j)\} \cup \{(x_i, x_j')\}$ for $i \neq j$
  not related by augmentation

Augmentation: any transformation (e.g. rotation) of the original jet

positive and negative pairs = pseudo-labels

Train a network to map raw data to a new representation space, $f : \mathcal{J} \to \mathcal{R}$
$\to$ dim($\mathcal{R}$)$= 1000$

# Contrastive learning of jet representations

Optimise for:

1. alignment: positive-pairs close together in $\mathcal{R} \Rightarrow$ invariance
2. uniformity: negative-pairs far apart in $\mathcal{R} \Rightarrow$ discriminative

# Contrastive learning of jet representations

Optimise for:

1. alignment: positive-pairs close together in $\mathcal{R} \Rightarrow$ invariance
2. uniformity: negative-pairs far apart in $\mathcal{R} \Rightarrow$ discriminative

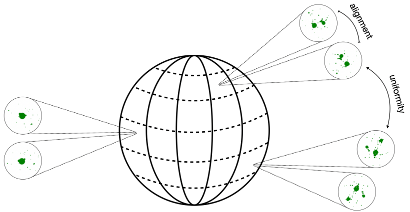
Contrastive loss:

$$\mathcal{L}_i = -\log \frac{\exp(s(z_i, z_i')/\tau)}{\sum_{x \in \text{batch}} \mathbb{I}_{i \neq j} \left[\exp(s(z_i, z_j)/\tau) + \exp(s(z_i, z_j')/\tau)\right]}$$

Similarity measure in $\mathcal{R}$:

$$s(z_i, z_j) = \frac{z_i \cdot z_j}{|z_i||z_j|}, \quad z_i = f(x_i)$$

$\Rightarrow$ defined on unit-hypersphere




JetCLR $\rightarrow$ code at https://github.com/bmdillon/JetCLR

# Contrastive learning of jet representations

The training procedure:

1. sample batch of jets, $x_i$
2. create an augmented batch of jets, $x_i'$
3. forward-pass both through the network
4. compute the loss & update weights

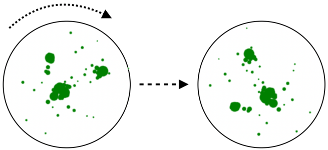
# Contrastive learning of jet representations

The training procedure:

1. sample batch of jets, $x_i$
2. create an augmented batch of jets, $x_i'$
3. forward-pass both through the network
4. compute the loss & update weights

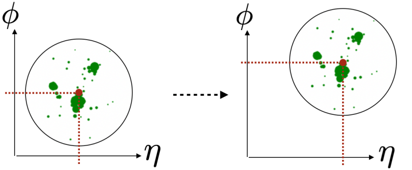
rotations

Angles sampled from $[0, 2\pi]$



translations

Translation distance sampled randomly



# Contrastive learning of jet representations

The training procedure:

1. sample batch of jets, $x_i$
2. create an augmented batch of jets, $x_i'$
3. forward-pass both through the network
4. compute the loss & update weights

### collinear splittings

some constituents randomly split,

$$p_{T,a} + p_{T,b} = p_T, \quad \eta_a = \eta_b = \eta$$
$$\phi_a = \phi_b = \phi$$

### low $p_T$ smearing

$(\eta, \phi)$ co-ordinates are re-sampled:

$$\eta' \sim \mathcal{N}\left(\eta, \frac{\Lambda_{\text{soft}}}{p_T}\right)$$
$$\phi' \sim \mathcal{N}\left(\phi, \frac{\Lambda_{\text{soft}}}{p_T}\right).$$

# Contrastive learning of jet representations

The training procedure:

1. sample batch of jets, $x_i$
2. create an augmented batch of jets, $x_i'$
3. forward-pass both through the network
4. compute the loss & update weights

permutation invariance

Transformer-encoder network

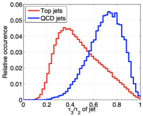
- ⋆ based on 'self-attention' mechanism
- ⋆ output invariant to constituent ordering

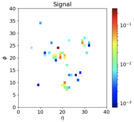
more info. in additional slides

# Quality measure of observables

Many representations used in practice:

- raw constituent data (dim $\sim$ 300)
- jet images (dim $\sim$ 1600)
- Energy Flow Polynomials (dim $\sim$ 1000)
  (Thaler et al: arXiv:1712.07124)

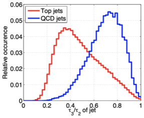


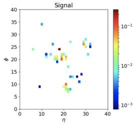



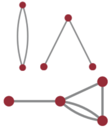

# Quality measure of observables

Many representations used in practice:

- raw constituent data (dim $\sim$ 300)
- jet images (dim $\sim$ 1600)
- Energy Flow Polynomials (dim $\sim$ 1000)
  (Thaler et al: arXiv:1712.07124)



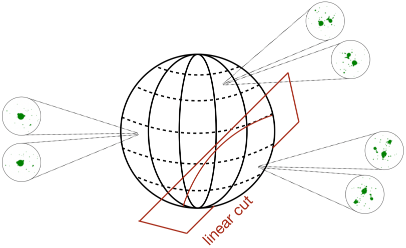



Compare these using a Linear Classifier Test (LCT)

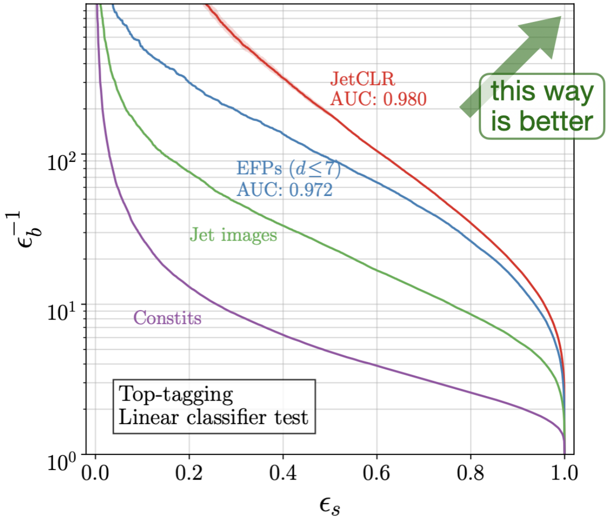
- ⋆ use top-tagging as a test
- ⋆ **linear cut** in the observable space
- ⋆ supervised - uses simulations
- ⋆ measures:

  $\epsilon_s$ - true positive rate

  $\epsilon_b$ - false positive rate



# Linear classifier test results



# Linear classifier test results

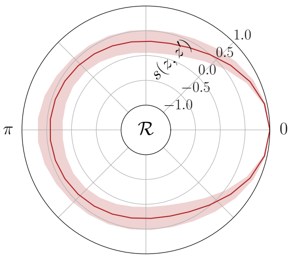
Where does the performance come from?

| Augmentation | $\epsilon_b^{-1}(\epsilon_S = 0.5)$ | AUC |
|---|---|---|
| none | 15 | 0.905 |
| translations | 19 | 0.916 |
| rotations | 21 | 0.930 |
| soft+collinear | 89 | 0.970 |
| all combined (default) | 181 | 0.980 |

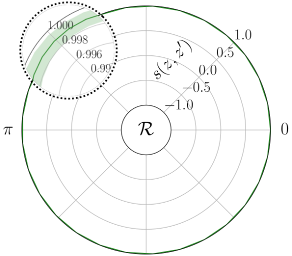
- ⋆ soft + collinear has the biggest effect
  translations + rotations also significant in final combination
- ⋆ also not very sensitive to S/B

# Invariances in representation space

without rotational invariance




with rotational invariance




⋆ $s(z, z') = \frac{z \cdot z'}{|z||z'|}$, $z = f(\vec{x})$, $z' = f\left(R(\theta)\vec{x}\right)$

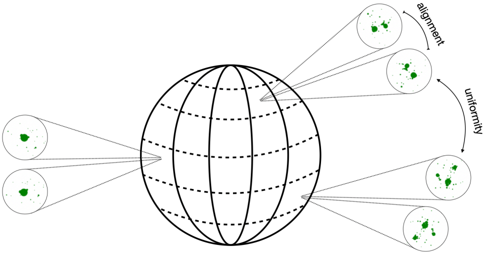
⇒ The network $f(\vec{x})$ is approx rotationally invariant

# Conclusion

Self-supervision allows for:

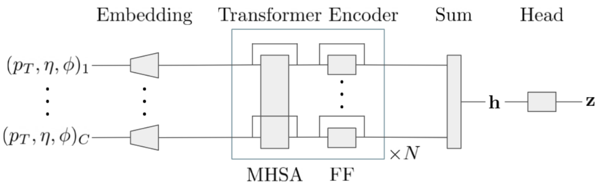
1. data-driven definition of observables
2. invariance to pre-defined symmetries/augmentations
3. high discriminative power

An example: **JetCLR** (contrastive learning of jet observables)



# The network

We use a transformer-encoder network $\longrightarrow$ permutation invariance




Equivariance $\longrightarrow$ invariance is similar to Deep-Sets/Energy-Flow-Networks: arXiv:1810.05165, P. T. Komiske, E. M. Metodiev, J. Thaler

The attention mechanism captures correlations between constituents by allowing each constituent to assign attention weights to every other constituent.