Symmetries, safety, & self-supervision

Barry M. Dillon

September 23, 2021

Institute for Theoretical Physics
University of Heidelberg

Portoroz 2021: Physics of the flavourful universe

hep-ph/2108.04253

BMD, Gregor Kasieczka, Hans Olischlager, Tilman Plehn, Peter Sorrenson, and Lorenz Vogel

Universität Heidelberg
Zukunft. Seit 1386.

Introduction

1. Machine-learning already plays an important role in particle physics analyses
   - ⋆ jet tagging
   - ⋆ model-agnostic new physics searches
   - ⋆ unfolding
   - ⋆ detector simulation
   - ⋆ ...
2. Trust issues.. Interpretability? Reliance on simulation?

Introduction

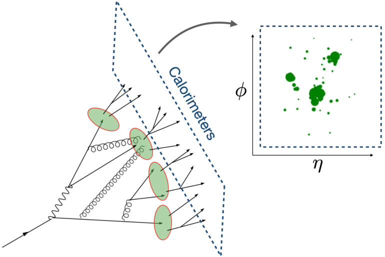
1. Machine-learning already plays an important role in particle physics analyses
   - ⋆ jet tagging
   - ⋆ model-agnostic new physics searches
   - ⋆ unfolding
   - ⋆ detector simulation
   - ⋆ . . .
2. Trust issues.. Interpretability? Reliance on simulation?
3. **Self-supervision**

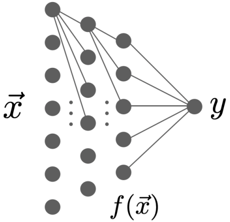
   Incorporate prior physics knowledge in neural networks w/o simulation
4. Improved performance in jet-tagging

   + many new opportunities for future research

Top-tagging with machine-learning

Neural network maps kinematical data to a predicted label





- 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

Top-tagging with machine-learning

Neural networks don’t explicitly learn the invariances associated with jets

⋆ no idea what features the network learns (..simulation artefacts?..)

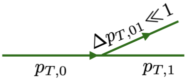
Top-tagging with machine-learning

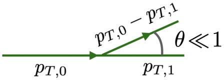
Neural networks don't explicitly learn the invariances associated with jets

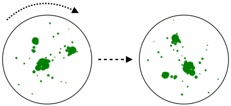
⋆ no idea what features the network learns (..simulation artefacts?..)

What do we want the network to learn?

- rotational invariance
- translational invariance
- IR safety
- Collinear safety







$f(R\vec{x}) = f(\vec{x}) = y$

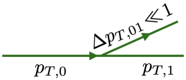
Top-tagging with machine-learning

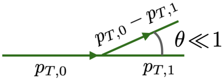
Neural networks don't explicitly learn the invariances associated with jets

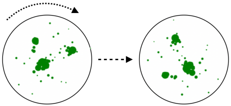
⋆ no idea what features the network learns (..simulation artefacts?..)

What do we want the network to learn?

- rotational invariance
- translational invariance
- IR safety
- Collinear safety







$f(R\vec{x}) = f(\vec{x}) = y$

Standard solution: Pre-processing & high-level observables

⋆ prevents the network learning from low-level raw data

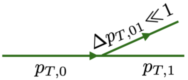
Top-tagging with machine-learning

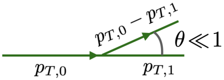
Neural networks don't explicitly learn the invariances associated with jets

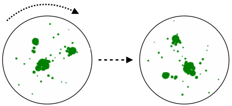
⋆ no idea what features the network learns (..simulation artefacts?..)

What do we want the network to learn?

- rotational invariance
- translational invariance
- IR safety
- Collinear safety







$$f(R\vec{x}) = f(\vec{x}) = y$$

Standard solution: Pre-processing & high-level observables

⋆ prevents the network learning from low-level raw data

Better solution: networks learn these invariances from the raw data

Optimising observables / representations

Key idea

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

Key idea

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
  map raw jet data to a new representation / observables
- ⋆ Self-supervision
  neural networks are optimised without truth-labels
  $\rightarrow$ can run directly on expt. data

Contrastive learning of jet representations

arXiv:2002.05709, Google Brain: simCLR, T. Chen, S. Kornblith, M. Norouzi, G. Hinton

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
- negative-pairs: $\{(x_i, x_j)\} \cup \{(x_i, x_j')\}$ for $i \neq j$

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

Contrastive learning of jet representations

arXiv:2002.05709, Google Brain: simCLR, T. Chen, S. Kornblith, M. Norouzi, G. Hinton

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
- negative-pairs: $\{(x_i, x_j)\} \cup \{(x_i, x_j')\}$ for $i \neq j$

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

Train a network to map to a new representation space, $f(\vec{x}_i) = \vec{z}_i$, $f : \mathcal{J} \rightarrow \mathcal{R}$

Optimise for:

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

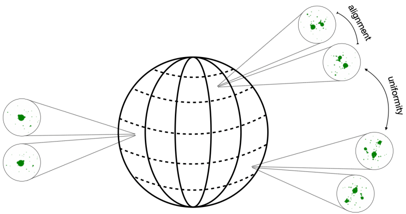
Contrastive learning of jet representations

Similarity measure in $\mathcal{R}$: $\quad s(z_i, z_j) = \frac{z_i \cdot z_j}{|z_i||z_j|}$

$\Rightarrow$ defined on unit-hypersphere

Contrastive loss:

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



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

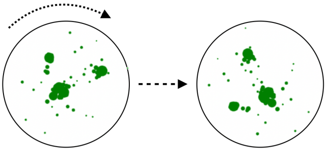
Contrastive learning of jet representations

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

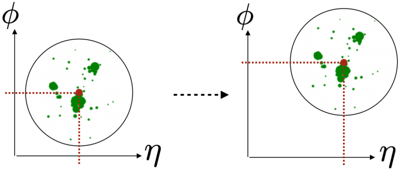
rotations

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

(η, ϕ) co-ordinates are re-sampled:

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

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

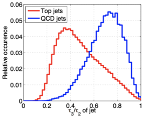
- ⋆ based on 'self-attention' mechanism
- ⋆ output invariant to constituent ordering

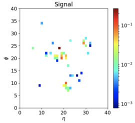
more info. in additional slides

Quality measure of observables

Benchmark representations:

- raw constituent data
- jet images
- Energy Flow Polynomials
  (Thaler et al: arXiv:1712.07124)

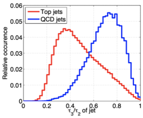


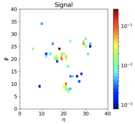


Quality measure of observables

Benchmark representations:

- raw constituent data
- jet images
- Energy Flow Polynomials
  (Thaler et al: arXiv:1712.07124)





Compare these using a Linear Classifier Test (LCT)

⋆ use top-tagging as a test

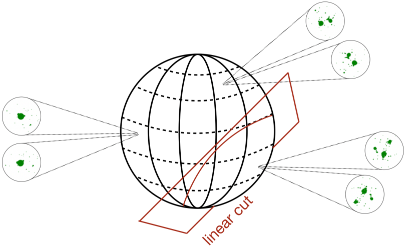
⋆ **linear cut** in the observable space

⋆ supervised - uses simulations

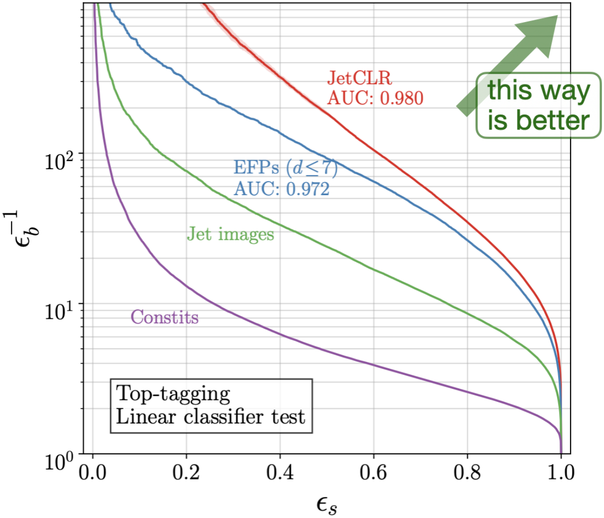
⋆ measures:

ϵ_s - true positive rate

ϵ_b - false positive rate



Linear classifier test results



Linear classifier test results

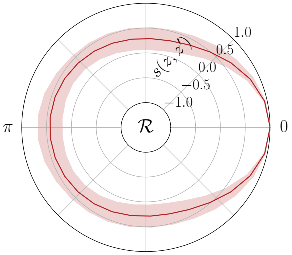
Where does the performance come from?

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

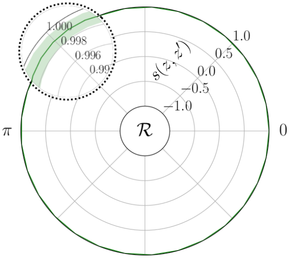
- ⋆ soft + collinear has the biggest effect
  translations + rotations also significant in final combination
- ⋆ also not very sensitive to S/B

Invariances in representation space

without rotational invariance



with rotational invariance



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

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

Summary & outlook

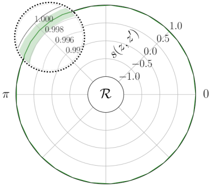
Self-supervision allows for:

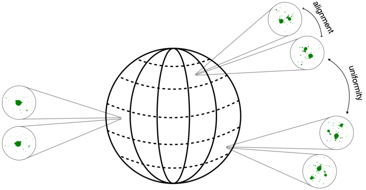
1. data-driven definition of observables
2. invariance to pre-defined symmetries/augmentations
3. high discriminative power

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

Outlook:

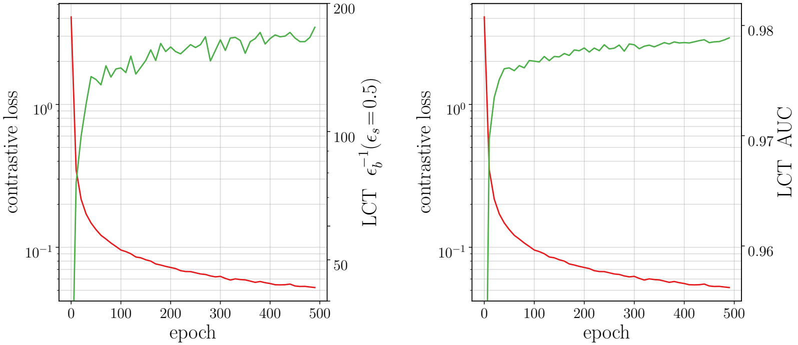
- incorporating particle-ID
- application beyond jet-tagging
- anomaly-detection
- . . .





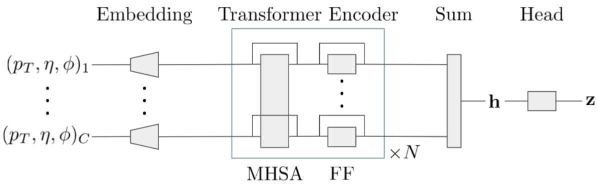
LCT - performance vs epochs

Performance as a function of training time / epochs



The network

We use a transformer-encoder network $\rightarrow$ 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.