Can Contrastive Learning de-bias my Model?

EP-NU Meeting

Radi Radev CERN, Alex Wilkinson UCL / Fermilab
Self-Supervised Learning in Vision

• You have a lot of data but not many labelled examples
• Train some model that utilises the unlabelled data
• Then you can fine-tune the base model using the small labeled sample
Self-Supervised Learning in Vision

• You have a lot of data but not many labelled examples
• Train some model that utilises the unlabelled data
• Then you can fine-tune the base model using the small labeled sample
Self-Supervised Learning in Vision

- You have a lot of data but not many labelled examples
- Train some model that utilises the unlabelled data
- Then you can fine-tune the base model using the small labeled sample

Illustration of MAE - vision foundation model
Self-Supervised Learning in Vision

• You have a lot of data but not many labelled examples
• Train some model that utilises the unlabelled data
• Then you can fine-tune the base model using the small labeled sample
Self-Supervised Learning in Vision and HEP

- You have a lot of data but not many labelled examples
- Train some model that utilises the unlabelled data
- Then you can fine-tune the base model using the small labeled sample
- But HEP simulation comes with detailed information?
- It can help mitigate biases we have in our simulation
Mitigating Biases by Pretraining

We explore a method where we use a combination of detector systematics and handcrafted augmentations to learn a robust representation.

Our method is based roughly on SimCLR - Simple Framework for Contrastive learning of Visual Representations - 2002.05709
Contrastive Learning of Representations
Pass an event $x_i$ through a neural network $f$ to extract a vector representation $z_i$.

$z_i$ is a high-dimensional vector (in our case 768d)
Representations

Pass an augmented event $x_i$ through a neural network $f$ to extract a different vector representation $z_i$. 
Contrastive Learning

Pass pairs of **augmented events** through a **neural network** $f$ to extract **vector representations**.
Contrastive Learning

Pass pairs of augmented events through a neural network $f$ to extract vector representations.

Representations from different events - low similarity
Contrastive Learning

Pass pairs of augmented events through a neural network $f$ to extract vector representations.

Representations from same event - high similarity
Contrastive Learning

Flexibility:
Use any augmentation - What invariance do we encode?

Use any neural network - What is the most natural data structure of the event?
Data
Liquid Argon TPC

A cryostat filled with liquid argon and a strong electric field.
Liquid Argon TPC

If a neutrino interacts with the medium this produces charged particles.
Liquid Argon TPC

Charged particles free electrons and produce scintillation light. The electrons drift towards the anode.
$x, y$ - pixel positions $z$: $t_0 - t_{arr}$ the difference between time of light and time of charge arrival

Liquid Argon TPC
Dataset

Single particle interactions within a LArTPC of 5 types $\mu, \pi, \gamma, e, p$, following PILArNet 2006.01993

Realistic detector simulation using larnd-sim, detector variations of 3 parameters taken from 2309.04639

<table>
<thead>
<tr>
<th>Detector Parameter</th>
<th>Range</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Electric Field</td>
<td>[0.45, 0.55]</td>
<td>$kV/cm$</td>
</tr>
<tr>
<td>Electron Lifetime</td>
<td>[500, 5000]</td>
<td>$\mu s$</td>
</tr>
<tr>
<td>Transverse Diffusion</td>
<td>[4e-6, 14e-6]</td>
<td>$cm^2/\mu s$</td>
</tr>
</tbody>
</table>
Simulation Overview

particle generation

ParticleBomb
Simulation Overview

\[ e : p_x, p_y, p_z, E \]

particle generation

ParticleBomb
Simulation Overview

e : \( p_x, p_y, p_z, E \)

- particle generation
- particle propagation

ParticleBomb

edep-sim
Simulation Overview

\[ e : p_x, p_y, p_z, E \]

particle generation
- ParticleBomb

particle propagation
- edep-sim

detector simulation
- larnd-sim

\[ X_i \]
Method
Our input is extremely **sparse**. To capture most of the event we would have to use a $500^3$ pixel cube and only 0.01% one of those would be non-empty.

Sparse Convolution operates only on non empty-voxels.

Check out MinkowskiEngine - a sparse autodiff tensor library.
Architecture: a sparse submanifold CNN based on ConvNeXt v2

We use an MLP to get the similarity vector for CLR and a Linear layer if we are training a classifier.
Augmentations

Handcrafted:
- random scaling, translation, identity, dropping voxels
Augmentations

$X_i \xrightarrow{\text{Augment}} \text{Sparse CNN} \xrightarrow{} Z_i$

**Handcrafted:**
- random scaling, translation, identity, dropping voxels

**Detector Variations:**
- electric field strength, longitudinal diffusion coefficient, and electron lifetime
Training and Evaluating SimCLR

We only need to train the base model \textbf{once}!

- Can train \textbf{multiple} models cheaply
- All downstream models are \textbf{decorrelated} from the parameters we used for augmentations
Results
Training

3 models:
- contrastive learning model, that was then frozen - fine-tuned on nominal data only
- classifier using nominal data only
- classifier using nominal + throws
Accuracy - Detector Variations

The **contrastive** model outperforms the classifiers trained directly on either **nominal** or **nominal+throws**.

It is also less affected by the systematic shifts.
The score of the correct class from the contrastive model is less likely to change when we shift the detector parameters.
Future Work

• Fine-tune the model on another task e.g predicting **final state particles**
• Use **larger batch** sizes for the base model
• Explore other contrastive learning methods
• Compare with other methods of de-biasing (e.g DANN)

I think this is could be a very exciting way to combine novel ideas from vision enhancing the way ML is used in physics analyses!
Thank you

radi.radev@cern.ch
In practice the set of **augmentations** to be applied to the pairs is picked randomly for each training iteration.
Contrastive Learning

No labels needed - can pre-train on real data!
Results on PILArNet
CLR Results

Overall Accuracy

- Classifier
- Classifier + Aug
- CLR
CLR v Linear Classifier Baselines

![Graph showing accuracy and balanced accuracy for different models.](image)

- **Accuracy**
- **Balanced Accuracy**

Legend:
- Classifier
- Overfit Classifier
- Classifier + Augmentations
- CLR Randomly Initialized
- CLR

All models are **frozen** - logistic regression fit on top.

For the classifiers the **last layer** is removed and we fit on the features after maxpooling.

For CLR we remove the **MLP** and again use the features after maxpooling.
Augmentations:
- random scaling, translation, rotation, dropping voxels

Architecture:
- a sparse sub manifold CNN based on ConvNeXt v2

But wait aren’t CNNs already invariant to translations?
Aside - CNN Translation Invariance

But wait aren’t CNNs already invariant to translations?

Convolutions are **equivariant** to translation, but this does not directly translate to invariance.

Although architectures can be constructed to be invariant to translations, most modern CNNs are not by default

**Turns out not quite!**

Adapted From “CNNs Are Not Invariant to Translation, but They Can Learn to Be”