LEGNN Jet Tagging Network

Jason Wong
Jet Tagging Problem

- Particle collisions produce quarks which lose energy via parton showers
  - In quark’s CoM frame, this forms a jet of four-momenta
- Jet problem: **classify** jets (four-momenta from parton showers) as either a **top quark** or **light quark/gluon**
  - Simulated dataset with 1.2 million jets for training and 400k jets for validation
Other Research

- ParticleNet: Edge convolution GNN for permutation invariance
- LGN: In hidden layers, creates relativistic tensors of arbitrary order (via tensor product) and uses Clebsch-Gordan operators to decompose tensors into irreducible representations of the Lorentz group
  - Permutation invariant as well as Lorentz equivariant

\[ \mathcal{F}_i^{(p+1)} = \mathcal{L}_{CG} \left( \mathcal{F}^{(p)} \right)_i = W \cdot \left( \mathcal{F}_i^{(p)} \oplus CG \left[ \mathcal{F}_i^{(p)} \right]^\otimes2 \oplus \right. \\
\left. \oplus CG \left[ \sum_j f(p^2_{ij})p_{ij} \otimes \mathcal{F}_j^{(p)} \right] \right). \quad (25) \]
Reason for Development

- Technology requirement: jet tagging in as little memory and time as possible
- Current networks (e.g. ParticleNet) are large and slow
- Previous research uses symmetry to reduce network size with constraints
  - For jet tagging, this is a graph structure with node/edge equivariance and invariance
Equivariance

- Neural network $\phi(\vec{x})$ with input $\vec{x}$
- Rotation operator $R(\theta)$
- Equivariance: $\phi(R(\theta)\vec{x}) = R(\theta)\phi(\vec{x})$
  - Output transforms as if you transformed the input
- Lorentz equivariance contains rotations and Lorentz boosts
  - Poincare group: Lorentz group with translations
**E(n) Equivariant Model: EGNN**

- Euclidean group includes rotations and translations
- Take vectors in \( \mathbb{R}^n \) as input
- Use coordinate differences for \( E(n) \) equivariance: \( \vec{x} - \vec{y} \)
- Create invariants such as the Euclidean norm: \( h = \| \vec{x} - \vec{y} \|^2 \)
  - MLPs of invariants

\[
\begin{align*}
\mathbf{m}_{ij} &= \phi_e \left( \mathbf{h}_i^l, \mathbf{h}_j^l, \| \mathbf{x}_i^l - \mathbf{x}_j^l \|, a_{ij} \right) \\
\mathbf{x}_i^{l+1} &= \mathbf{x}_i^l + C \sum_{j \neq i} (\mathbf{x}_i^l - \mathbf{x}_j^l) \phi_x (\mathbf{m}_{ij}) \\
\mathbf{m}_i &= \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ij} \\
\mathbf{h}_i^{l+1} &= \phi_h (\mathbf{h}_i^l, \mathbf{m}_i)
\end{align*}
\]
Equivariance Strategy

- Stay within the vector space $\mathbb{R}^n$ by taking linear combinations of vectors:
  - $\vec{z} = a\vec{x} + b\vec{y}$
- Vectors transform under the desired group symmetry
- Vector input and output leads to equivariance
- For Lorentz equivariance, just use Minkowski norm
LEGNN Network

\[ \vec{m}_{ij} = \phi_e(\vec{h}_i, \vec{h}_j, \eta_{\mu\nu} \Delta x_{ij,\mu} \Delta x_{ij,\nu}, a_{ij}) \]

\[ p_{i}^{l+1,\mu} = p_{i}^{l,\mu} + C \sum_{j \neq i} (a_{\mu} p_{i}^{l,\mu} + b_{\nu} p_{j}^{l,\nu}) \phi_x(\vec{m}_{ij}) \]

\[ \vec{m}_{i} = \sum_{j \in N(i)} \vec{m}_{ij} \]

\[ \vec{h}_{i}^{l+1} = \phi_h(\vec{h}_{i}^{l}, \vec{m}_{i}) \]
Input

Invariant Embeddings

\[ \mathbf{R}^N \]

Compute Messages

\[ m_{ij} = \phi_m(h_i, h_j, \|\vec{p}_i^\mu - \vec{p}_j^\mu\|^2) \]

Message Passing Layer

\[ m_i = \sum_j m_{ij} \]

Update vectors & embeddings

Messages: \( \phi_m(h_i, h_j, r_{ij}^2) \)

Embeddings: \( \phi_n(h_i, m_i) \)

Vectors: \( \phi_p(m_i) \)

\[ p_i^{l+1} = p_i^l + c \sum_{j \neq i} (a p_i^l + b p_j^l) \phi_p(m_i) \]
Results

Caveat: Model accidentally trained with a sort of “edge invariance” instead of node invariance. Node invariance seems to train at least 10x faster and with greater accuracy (~94%), but more testing is required.
Network Equivariance
Conclusion

- Conducted novel research using physics and group theory for machine learning
- Applied research to jet tagging problem, obtaining good results
- Quickly trained neural network with significantly fewer parameters than other models
- Went through major parts of research process
References

- Jet Tagging Dataset: https://zenodo.org/record/2603256#.YUARE55Kj65
- Code repository: https://github.com/jw5243/Lorentz-Equivariant-GNN
Reducing LGN Network to LEGNN

- LEGNN network ends up being a simplified case of the LGN network
  - LGN uses irreducible representations up to any tensor order
  - LEGNN stops at vectors, or 1D-tensors
  - Like first order approximation
LGN Layer

\[ F_i^{(p+1)} = \mathcal{L}_{CG} \left( F_i^{(p)} \right)_i = W \cdot \left( F_i^{(p)} \oplus CG \left[ F_i^{(p)} \right] \otimes 2 \right) \oplus CG \left( \sum_j f \left( p_{i,j}^2 \right) p_{i,j} \otimes F_j^{(p)} \right) \]

Other particle interactions
LGN-Norm-Squared

\[ F_{i}^{(p+1)} = \mathcal{L}_{CG} \left( F_{i}^{(p)} \right) = W \cdot \left( F_{i}^{(p)} \oplus CG \left[ F_{i}^{(p)} \right] \otimes 2 \oplus \right. \]

\[ \oplus CG \left[ \sum_{j} f \left( p_{i,j}^2 \right) p_{ij} \otimes F_{j}^{(p)} \right] \]

Tensor Product

\[ p_{ij}^2 = \eta_{\mu\nu} \Delta p_{ij}^\mu \Delta p_{ij}^\nu \text{ with } \Delta p_{ij}^\mu = p_i^\mu - p_j^\mu \]
Connecting LGN to LEGNN

2 LGN layers is a superset of 1 L-EGNN layer

First LGN layer
- Returns input
- Norm-squared calculated
- MLP on scalars (e.g. norm)
- Linear combination of vectors

Second LGN layer
- Returns input
- Product of scalars (after MLP from first layer) and vectors
- Learnable operator can sum

Returns scalar when limited to 4-vectors