



# Symmetry in Deep Learning Representations

Johannes Lutzeyer

Advanced Deep Learning: Lecture 8

March 4, 2025

# Overview of Today's Lecture

- 1) Symmetry as a Unifying Principle In DL<sup>1</sup>;

---

<sup>1</sup>Several of the presented ideas are based on work presented in Bronstein et al. (2021).

# Overview of Today's Lecture

- 1) Symmetry as a Unifying Principle In DL<sup>1</sup>;
- 2) Neural Collapse during the terminal phase of training.

---

<sup>1</sup>Several of the presented ideas are based on work presented in Bronstein et al. (2021).

# Invariance and Equivariance

**Motivation:** The principles of invariance and equivariance of neural networks to the actions of certain groups have recently been postulated as a unifying principle in DL and give rise to an intriguing categorisation of DL architectures.

# Invariance and Equivariance

**Motivation:** The principles of invariance and equivariance of neural networks to the actions of certain groups have recently been postulated as a unifying principle in DL and give rise to an intriguing categorisation of DL architectures.

For some group  $G$  we say that a neural network  $n(\cdot) : \mathbb{I} \rightarrow \mathbb{R}^C$  is

- *invariant* to the action of all  $g \in G$  if and only if for all  $x \in \mathbb{I}$  we have  $n(gx) = n(x)$ .

# Invariance and Equivariance

**Motivation:** The principles of invariance and equivariance of neural networks to the actions of certain groups have recently been postulated as a unifying principle in DL and give rise to an intriguing categorisation of DL architectures.

For some group  $G$  we say that a neural network  $n(\cdot) : \mathbb{I} \rightarrow \mathbb{R}^C$  is

- *invariant* to the action of all  $g \in G$  if and only if for all  $x \in \mathbb{I}$  we have  $n(gx) = n(x)$ .
- *equivariant* to the action of all  $g \in G$  if and only if for all  $x \in \mathbb{I}$  we have  $n(gx) = gn(x)$ .

# Multi Layer Perceptrons

**Input:** Multi Layer Perceptrons (MLPs) act on vector-valued input.

# Multi Layer Perceptrons

**Input:** Multi Layer Perceptrons (MLPs) act on vector-valued input.

**Relevant Symmetry Group:** None.



# Multi Layer Perceptrons

**Input:** Multi Layer Perceptrons (MLPs) act on vector-valued input.

**Relevant Symmetry Group:** None.

**Discussion:** MLPs are universal approximations on vectors and have to learn all symmetry present in our data from scratch.

# Multi Layer Perceptrons

**Input:** Multi Layer Perceptrons (MLPs) act on vector-valued input.

**Relevant Symmetry Group:** None.

**Discussion:** MLPs are universal approximations on vectors and have to learn all symmetry present in our data from scratch.

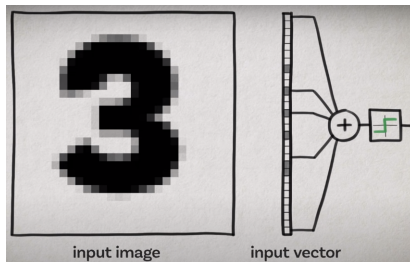


Image credit: Bronstein (2021)

# Multi Layer Perceptrons

**Input:** Multi Layer Perceptrons (MLPs) act on vector-valued input.

**Relevant Symmetry Group:** None.

**Discussion:** MLPs are universal approximations on vectors and have to learn all symmetry present in our data from scratch.

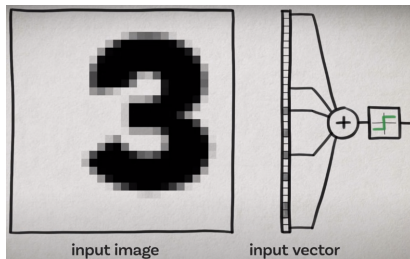


Image credit: Bronstein (2021)

# Multi Layer Perceptrons

**Input:** Multi Layer Perceptrons (MLPs) act on vector-valued input.

**Relevant Symmetry Group:** None.

**Discussion:** MLPs are universal approximations on vectors and have to learn all symmetry present in our data from scratch.

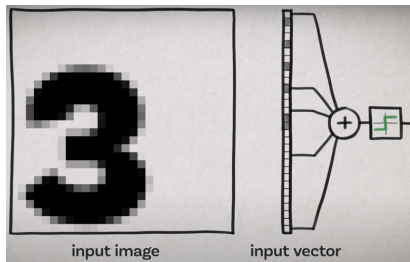


Image credit: Bronstein (2021)

# Multi Layer Perceptrons

**Input:** Multi Layer Perceptrons (MLPs) act on vector-valued input.

**Relevant Symmetry Group:** None.

**Discussion:** MLPs are universal approximations on vectors and have to learn all symmetry present in our data from scratch.

Most other DL architectures have hard-coded invariance and/or equivariance to certain symmetries present in our data. Such hard-coded invariance and/or equivariance are sometimes referred to as an *inductive bias* in these architectures.

# Convolutional Neural Networks

**Input:** Convolutional Neural Networks (CNNs) act on structured input such as sequences, images or tensor fields.

# Convolutional Neural Networks

**Input:** Convolutional Neural Networks (CNNs) act on structured input such as sequences, images or tensor fields.

**Relevant Symmetry Group:** The translation group  $T$  (acting on inputs of CNNs directly).

# Convolutional Neural Networks

**Input:** Convolutional Neural Networks (CNNs) act on structured input such as sequences, images or tensor fields.

**Relevant Symmetry Group:** The translation group  $T$  (acting on inputs of CNNs directly).

## Discussion:

- Convolutional Layers are equivariant to actions of  $t \in T$ .



# Convolutional Neural Networks

**Input:** Convolutional Neural Networks (CNNs) act on structured input such as sequences, images or tensor fields.

**Relevant Symmetry Group:** The translation group  $T$  (acting on inputs of CNNs directly).

## Discussion:

- Convolutional Layers are equivariant to actions of  $t \in T$ .
- Pooling Layers are “locally invariant” to actions of  $t \in T$ , i.e., we are invariant to actions of elements of  $T$  that translate within the patches over which we pool.

# Recurrent Neural Networks

**Input:** Recurrent Neural Networks (RNNs) act on sequence data.

# Recurrent Neural Networks

**Input:** Recurrent Neural Networks (RNNs) act on sequence data.

**Relevant Symmetry Group:** The translation group  $T$  (acting on input time indices).

# Recurrent Neural Networks

**Input:** Recurrent Neural Networks (RNNs) act on sequence data.

**Relevant Symmetry Group:** The translation group  $T$  (acting on input time indices).

**Discussion:** RNNs producing an output from each hidden state are equivariant to actions of  $t \in T$  if

# Recurrent Neural Networks

**Input:** Recurrent Neural Networks (RNNs) act on sequence data.

**Relevant Symmetry Group:** The translation group  $T$  (acting on input time indices).

**Discussion:** RNNs producing an output from each hidden state are equivariant to actions of  $t \in T$  if

- the translation of a sequence by  $\tau$  time steps into the future produces a sequence that is left-padded with  $\tau$  zero-vectors;

# Recurrent Neural Networks

**Input:** Recurrent Neural Networks (RNNs) act on sequence data.

**Relevant Symmetry Group:** The translation group  $T$  (acting on input time indices).

**Discussion:** RNNs producing an output from each hidden state are equivariant to actions of  $t \in T$  if

- the translation of a sequence by  $\tau$  time steps into the future produces a sequence that is left-padded with  $\tau$  zero-vectors;
- if for a given initial hidden state  $\mathbf{h}^{(0)}$  the RNN  $R(\mathbf{x}, \mathbf{h})$  satisfies  $\mathbf{h}^{(0)} = R(\mathbf{0}, \mathbf{h}^{(0)})$ .

**Input:** DeepSets architecture (Zaheer et al., 2017) acts on sets.

# DeepSets

**Input:** DeepSets architecture (Zaheer et al., 2017) acts on sets.

**Relevant Symmetry Group:** Permutation group  $S$ .



# DeepSets

**Input:** DeepSets architecture (Zaheer et al., 2017) acts on sets.

**Relevant Symmetry Group:** Permutation group  $S$ .

**Discussion:** DeepSets is invariant to actions of elements in  $S$ .

# DeepSets

**Input:** DeepSets architecture (Zaheer et al., 2017) acts on sets.

**Relevant Symmetry Group:** Permutation group  $S$ .

**Discussion:** DeepSets is invariant to actions of elements in  $S$ .

A function  $f(X)$  operating on a set  $X$  having elements from a countable universe, is a valid set function, i.e., invariant to the permutation of instances in  $X$ , if and only if it can be decomposed in the form  $\rho(\sum_{x \in X} \phi(x))$ , for suitable transformations  $\phi$  and  $\rho$ .

**Input:** DeepSets architecture (Zaheer et al., 2017) acts on sets.

**Relevant Symmetry Group:** Permutation group  $S$ .

**Discussion:** DeepSets is invariant to actions of elements in  $S$ .

A function  $f(X)$  operating on a set  $X$  having elements from a countable universe, is a valid set function, i.e., invariant to the permutation of instances in  $X$ , if and only if it can be decomposed in the form  $\rho(\sum_{x \in X} \phi(x))$ , for suitable transformations  $\phi$  and  $\rho$ .

Zaheer et al. (2017) propose to use MLPs, i.e., universal approximators, to learn  $\rho$  and  $\phi$ .

# Transformers

**Input:** Transformers architecture can be applied to data from many domains, sets, sequences, structured data, complete graphs.

# Transformers

**Input:** Transformers architecture can be applied to data from many domains, sets, sequences, structured data, complete graphs.

**Relevant Symmetry Group:** Permutation group  $S$ .

# Transformers

**Input:** Transformers architecture can be applied to data from many domains, sets, sequences, structured data, complete graphs.

**Relevant Symmetry Group:** Permutation group  $S$ .

**Discussion:** The self-attention mechanism, without positional encodings, is equivariant to actions of elements in  $S$ .

# Transformers

**Input:** Transformers architecture can be applied to data from many domains, sets, sequences, structured data, complete graphs.

**Relevant Symmetry Group:** Permutation group  $S$ .

**Discussion:** The self-attention mechanism, without positional encodings, is equivariant to actions of elements in  $S$ . Positional encodings are designed to break this symmetry.

# Graph Neural Networks

**Input:** Graph Neural Networks (GNNs) act on graph-structured data, i.e., graph in which nodes are attributed with vectors.



# Graph Neural Networks

**Input:** Graph Neural Networks (GNNs) act on graph-structured data, i.e., graph in which nodes are attributed with vectors.

**Relevant Symmetry Group:** Permutation group  $S$ .

# Graph Neural Networks

**Input:** Graph Neural Networks (GNNs) act on graph-structured data, i.e., graph in which nodes are attributed with vectors.

**Relevant Symmetry Group:** Permutation group  $S$ .

**Discussion:**

- the message passing function is locally invariant to actions of  $s \in S$  on the node set, i.e., the aggregation of hidden states over each neighbourhood in the graph is individually unaffected by the action of  $s \in S$ .

# Graph Neural Networks

**Input:** Graph Neural Networks (GNNs) act on graph-structured data, i.e., graph in which nodes are attributed with vectors.

**Relevant Symmetry Group:** Permutation group  $S$ .

## Discussion:

- the message passing function is locally invariant to actions of  $s \in S$  on the node set, i.e., the aggregation of hidden states over each neighbourhood in the graph is individually unaffected by the action of  $s \in S$ .  
⇒ Consequently, a message passing layer acts in a permutation-equivariant way on the attributed graph.

# Graph Neural Networks

**Input:** Graph Neural Networks (GNNs) act on graph-structured data, i.e., graph in which nodes are attributed with vectors.

**Relevant Symmetry Group:** Permutation group  $S$ .

## Discussion:

- the message passing function is locally invariant to actions of  $s \in S$  on the node set, i.e., the aggregation of hidden states over each neighbourhood in the graph is individually unaffected by the action of  $s \in S$ .  
⇒ Consequently, a message passing layer acts in a permutation-equivariant way on the attributed graph.
- for graph-level learning tasks we employ a permutation-invariant pooling function over node representations yielding a GNN that is invariant to actions of  $s \in S$ .

# Deep Learning Architectures used in Biochemical Contexts

**Input:** These DL architectures process structured data (such as graphs or point-clouds) which has coordinates in 2-dimensional or 3-dimensional space.

# Deep Learning Architectures used in Biochemical Contexts

**Input:** These DL architectures process structured data (such as graphs or point-clouds) which has coordinates in 2-dimensional or 3-dimensional space.

**Relevant Symmetry Group:** Among others: the group of isometries of a Euclidean space of dimension  $n$ , including translations and rotations,  $E(n)$ .

# Deep Learning Architectures used in Biochemical Contexts

**Input:** These DL architectures process structured data (such as graphs or point-clouds) which has coordinates in 2-dimensional or 3-dimensional space.

**Relevant Symmetry Group:** Among others: the group of isometries of a Euclidean space of dimension  $n$ , including translations and rotations,  $E(n)$ .

**GNN examples:** categorisation by (Joshi et al., 2023)

- SchNet is  $SO(3)$ -invariant (Schütt et al., 2018);
- EGNN is  $SO(3)$ -equivariant on cartesian vectors (Satorras et al., 2021);
- **TFN** (Thomas et al., 2018) and **MACE** (Batatia et al., 2022) are  $SO(3)$ -equivariant on spherical vectors.

## Tensor Field Networks (TFNs) (Thomas et al., 2018)

**Input:** TFNs process tensorfields, i.e., a set of vectors  $\mathcal{S}$  of vectors in  $\mathbb{R}^3$  each equipped with a feature vector of arbitrary dimension in some vector space.



## Tensor Field Networks (TFNs) (Thomas et al., 2018)

**Input:** TFNs process tensorfields, i.e., a set of vectors  $\mathcal{S}$  of vectors in  $\mathbb{R}^3$  each equipped with a feature vector of arbitrary dimension in some vector space.

**Relevant Symmetry Group:** Translations and rotation of the points in  $\mathbb{R}^3$  and permutation of the point indices.

## Tensor Field Networks (TFNs) (Thomas et al., 2018)

**Input:** TFNs process tensorfields, i.e., a set of vectors  $\mathcal{S}$  of vectors in  $\mathbb{R}^3$  each equipped with a feature vector of arbitrary dimension in some vector space.

**Relevant Symmetry Group:** Translations and rotation of the points in  $\mathbb{R}^3$  and permutation of the point indices.

### Discussion:

- Permutation equivariance follows, similarly to GNNs, as a consequence of the sum over all points in the point cloud  $\mathcal{S}$ .

The equation of a TFN convolution layer is the following

$$\sum_{m_f, m_i} C_{(l_f, m_f)(l_i, m_i)}^{(l_o, m_o)} \sum_{b \in \mathcal{S}} F_{cm_f}^{(l_f, l_i)}(\vec{r}_a - \vec{r}_b) V_{bcm_i}^{(l_i)},$$

where  $C_{(l_f, m_f)(l_i, m_i)}^{(l_o, m_o)}$  denotes the Clebsch-Gordan coefficients (expressing tensor products),  $V_{bcm_i}^{(l_i)}$  denotes the feature vectors and the filters are defined as  $F_{cm}^{(l_f, l_i)}(\vec{r}) = R_c^{(l_f, l_i)}(r) Y_m^{(l_f)}(\hat{r})$  with trainable scalars  $R_c^{(l_f, l_i)}(r)$  and  $Y_m^{(l_f)}(\hat{r})$  denoting the spherical harmonics (a basis of functions on the sphere).

## Tensor Field Networks (TFNs) (Thomas et al., 2018)

**Input:** TFNs process tensorfields, i.e., a set of vectors  $\mathcal{S}$  of vectors in  $\mathbb{R}^3$  each equipped with a feature vector of arbitrary dimension in some vector space.

**Relevant Symmetry Group:** Translations and rotation of the points in  $\mathbb{R}^3$  and permutation of the point indices.

### Discussion:

- Permutation equivariance follows, similarly to GNNs, as a consequence of the sum over all points in the point cloud  $\mathcal{S}$ .
- Translation invariance of point locations follows from the fact that we depend on points in  $\mathbb{R}^3$  only via their difference.

The equation of a TFN convolution layer is the following

$$\sum_{m_f, m_i} C_{(l_f, m_f)(l_i, m_i)}^{(l_o, m_o)} \sum_{b \in \mathcal{S}} F_{cm_f}^{(l_f, l_i)}(\vec{r}_a - \vec{r}_b) V_{bcm_i}^{(l_i)},$$

where  $C_{(l_f, m_f)(l_i, m_i)}^{(l_o, m_o)}$  denotes the Clebsch-Gordan coefficients (expressing tensor products),  $V_{bcm_i}^{(l_i)}$  denotes the feature vectors and the filters are defined as  $F_{cm}^{(l_f, l_i)}(\vec{r}) = R_c^{(l_f, l_i)}(r) Y_m^{(l_f)}(\hat{r})$  with trainable scalars  $R_c^{(l_f, l_i)}(r)$  and  $Y_m^{(l_f)}(\hat{r})$  denoting the spherical harmonics (a basis of functions on the sphere).

## Tensor Field Networks (TFNs) (Thomas et al., 2018)

**Input:** TFNs process tensorfields, i.e., a set of vectors  $\mathcal{S}$  of vectors in  $\mathbb{R}^3$  each equipped with a feature vector of arbitrary dimension in some vector space.

**Relevant Symmetry Group:** Translations and rotation of the points in  $\mathbb{R}^3$  and permutation of the point indices.

### Discussion:

- Permutation equivariance follows, similarly to GNNs, as a consequence of the sum over all points in the point cloud  $\mathcal{S}$ .
- Translation invariance of point locations follows from the fact that we depend on points in  $\mathbb{R}^3$  only via their difference.
- Rotation equivariance follows from the equivariance of the spherical harmonics  $Y_m^{(l_f)}(\hat{r})$  and the Clebsch-Gordan coefficients  $C_{(l_f, m_f)(l_i, m_i)}^{(l_o, m_o)}$ .

The equation of a TFN convolution layer is the following

$$\sum_{m_f, m_i} C_{(l_f, m_f)(l_i, m_i)}^{(l_o, m_o)} \sum_{b \in \mathcal{S}} F_{cm_f}^{(l_f, l_i)}(\vec{r}_a - \vec{r}_b) V_{bcm_i}^{(l_i)},$$

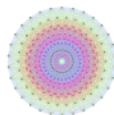
where  $C_{(l_f, m_f)(l_i, m_i)}^{(l_o, m_o)}$  denotes the Clebsch-Gordan coefficients (expressing tensor products),  $V_{bcm_i}^{(l_i)}$  denotes the feature vectors and the filters are defined as  $F_{cm}^{(l_f, l_i)}(\vec{r}) = R_c^{(l_f, l_i)}(r) Y_m^{(l_f)}(\hat{r})$  with trainable scalars  $R_c^{(l_f, l_i)}(r)$  and  $Y_m^{(l_f)}(\hat{r})$  denoting the spherical harmonics (a basis of functions on the sphere).

# Useful Resources: The Symmetry and Geometry in Neural Representations Workshop Series

The community organising these workshops is collecting and categorising a lot of up-to-date and relevant material on this topic in this repository.

## neurereps/**awesome-neural-geometry**

A curated collection of resources and research related to the geometry of representations in the brain, deep networks, and beyond



 11

Contributors

 0

Issues

 845

Stars

 57

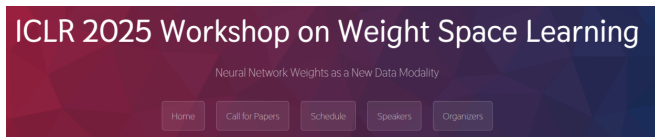
Forks



<https://github.com/neurereps/awesome-neural-geometry>

# A New Trend: Weight Space Learning

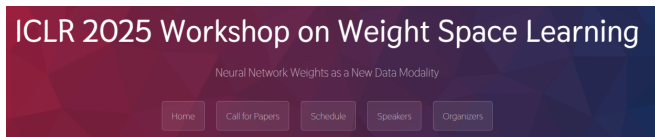
There is an ICLR workshop that will happen in April 2025 for the first time.



<https://weight-space-learning.github.io/>

## A New Trend: Weight Space Learning

There is an ICLR workshop that will happen in April 2025 for the first time.

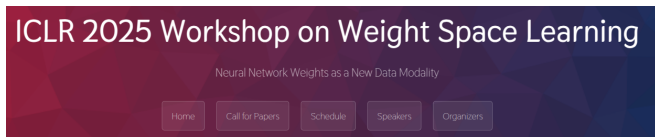


<https://weight-space-learning.github.io/>

Here we build neural networks that process trained weights of other neural networks. Neural network functions, parametrised by their weight functions, are invariant to actions of the permutation group  $S$  (Lim et al., 2024a,b).

## A New Trend: Weight Space Learning

There is an ICLR workshop that will happen in April 2025 for the first time.



<https://weight-space-learning.github.io/>

Here we build neural networks that process trained weights of other neural networks. Neural network functions, parametrised by their weight functions, are invariant to actions of the permutation group  $S$  (Lim et al., 2024a,b).

For example, consider a two-layer MLP with parameter matrices  $W_1$  and  $W_2$  and no biases, then for any permutation matrix  $P \in S$ ,

$$W_2 P^T \sigma(P W_1 x) = W_2 P^T P \sigma(W_1 x) = W_2 \sigma(W_1 x).$$



## Neural collapse during the terminal phase of deep learning training

Papayan et al. (2020)<sup>2</sup> study terminal phase of training (TPT), which begins at the epoch where training error first vanishes. During TPT, the training error stays effectively zero, while training loss is pushed toward zero.

---

<sup>2</sup>The text on this slide is a near-identical reproduction from the text contained in their abstract. 13/14

## Neural collapse during the terminal phase of deep learning training

Papayan et al. (2020)<sup>2</sup> study terminal phase of training (TPT), which begins at the epoch where training error first vanishes. During TPT, the training error stays effectively zero, while training loss is pushed toward zero.

([Brief digression](#) on overfitting and overparametrisation in DL (Bubeck & Sellke, 2023)).

## Neural collapse during the terminal phase of deep learning training

Papayan et al. (2020)<sup>2</sup> study terminal phase of training (TPT). They consider three architectures (CNNs) on seven classification datasets.

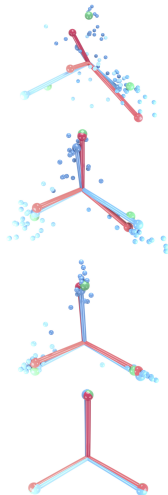
---

<sup>2</sup>The text on this slide is a near-identical reproduction from the text contained in their abstract. 13/14

## Neural collapse during the terminal phase of deep learning training

Papayan et al. (2020)<sup>2</sup> study terminal phase of training (TPT). They consider three architectures (CNNs) on seven classification datasets.

They observe and define neural collapse (NC):



---

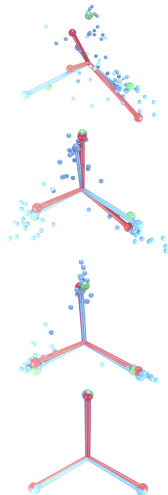
<sup>2</sup>The text on this slide is a near-identical reproduction from the text contained in their abstract.

# Neural collapse during the terminal phase of deep learning training

Papayan et al. (2020)<sup>2</sup> study terminal phase of training (TPT). They consider three architectures (CNNs) on seven classification datasets.

They observe and define neural collapse (NC):

**(NC1)** Cross-example within-class variability of last-layer training activations collapses to zero, as the individual activations themselves collapse to their class means (light blue).



---

<sup>2</sup>The text on this slide is a near-identical reproduction from the text contained in their abstract.

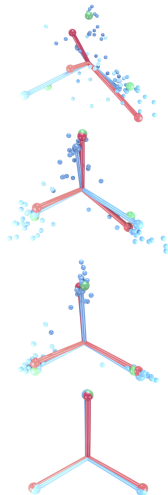
# Neural collapse during the terminal phase of deep learning training

Papayan et al. (2020)<sup>2</sup> study terminal phase of training (TPT). They consider three architectures (CNNs) on seven classification datasets.

They observe and define neural collapse (NC):

**(NC1)** Cross-example within-class variability of last-layer training activations collapses to zero, as the individual activations themselves collapse to their class means (light blue).

**(NC2)** The class means collapse to the vertices of a simplex equiangular tight frame (ETF) (green).



---

<sup>2</sup>The text on this slide is a near-identical reproduction from the text contained in their abstract.

# Neural collapse during the terminal phase of deep learning training

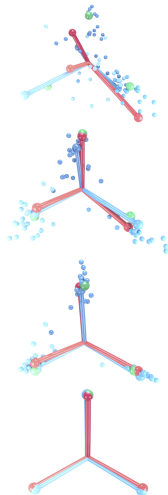
Papayan et al. (2020)<sup>2</sup> study terminal phase of training (TPT). They consider three architectures (CNNs) on seven classification datasets.

They observe and define neural collapse (NC):

**(NC1)** Cross-example within-class variability of last-layer training activations collapses to zero, as the individual activations themselves collapse to their class means (light blue).

**(NC2)** The class means collapse to the vertices of a simplex equiangular tight frame (ETF) (green).

**(NC3)** Up to rescaling, the last-layer classifiers (red) collapse to the class means.



---

<sup>2</sup>The text on this slide is a near-identical reproduction from the text contained in their abstract.

# Neural collapse during the terminal phase of deep learning training

Papayan et al. (2020)<sup>2</sup> study terminal phase of training (TPT). They consider three architectures (CNNs) on seven classification datasets.

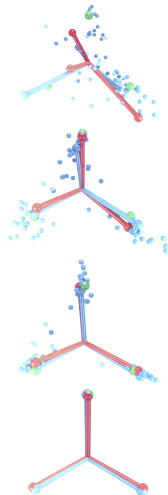
They observe and define neural collapse (NC):

**(NC1)** Cross-example within-class variability of last-layer training activations collapses to zero, as the individual activations themselves collapse to their class means (light blue).

**(NC2)** The class means collapse to the vertices of a simplex equiangular tight frame (ETF) (green).

**(NC3)** Up to rescaling, the last-layer classifiers (red) collapse to the class means.

**(NC4)** For a given activation, the classifier's decision collapses to simply choosing whichever class has the closest train class mean.



---

<sup>2</sup>The text on this slide is a near-identical reproduction from the text contained in their abstract.



# Neural collapse during the terminal phase of deep learning training

Papayan et al. (2020)<sup>2</sup> study terminal phase of training (TPT). They consider three architectures (CNNs) on seven classification datasets.

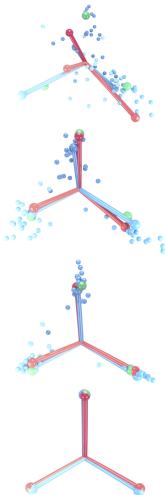
They observe and define neural collapse (NC):

**(NC1)** Cross-example within-class variability of last-layer training activations collapses to zero, as the individual activations themselves collapse to their class means (light blue).

**(NC2)** The class means collapse to the vertices of a simplex equiangular tight frame (ETF) (green).

**(NC3)** Up to rescaling, the last-layer classifiers (red) collapse to the class means.

**(NC4)** For a given activation, the classifier's decision collapses to simply choosing whichever class has the closest train class mean.



The symmetric and very simple geometry induced by the TPT confers important benefits, including *better generalization performance, better robustness, and better interpretability.*

<sup>2</sup>The text on this slide is a near-identical reproduction from the text contained in their abstract.

# Conclusions

- Considering the invariance and equivariance of different DL architectures provides a nice lens through which to view DL.


# Conclusions

- Considering the invariance and equivariance of different DL architectures provides a nice lens through which to view DL.
- It can be hugely beneficial to be aware of symmetries present in the input data domain to choose a DL model that respects this symmetry.

# Conclusions

- Considering the invariance and equivariance of different DL architectures provides a nice lens through which to view DL.
- It can be hugely beneficial to be aware of symmetries present in the input data domain to choose a DL model that respects this symmetry.
- The Neural Collapse phenomenon is cool.

# Thank you for your attention!

 @JLutzeyer

## References

- I. Batatia, D. P. Kovacs, G. Simm, C. Ortner & G. Csányi, "MACE: Higher order equivariant message passing neural networks for fast and accurate force fields," *Advances in Neural Information Processing Systems*, pp. 11423-11436, 2022.
- M. Bronstein, "Geometric Deep Learning: The Erlangen Programme of ML," *Keynote Talk at The International Conference on Learning Representations*, 2021.
- M. Bronstein, J. Bruna, T. Cohen & P. Veličković, "Geometric deep learning: Grids, groups, graphs, geodesics, and gauges," *arXiv preprint arXiv:2104.13478*, 2021.
- S. Bubeck & M. Sellke, "A universal law of robustness via isoperimetry," *Journal of the ACM*, pp. 1–18, 2023.
- C. K. Joshi, C. Bodnar, S. V. Mathis, T. Cohen & P. Lio, "On the expressive power of geometric graph neural networks," *ICML*, 2023.
- D. Lim, T. Putterman, R. Walters, H. Maron & S. Jegelka, "The Empirical Impact of Neural Parameter Symmetries, or Lack Thereof," *NeurIPS*, 2024.
- D. Lim, H. Maron, M. T. Law, J. Lorraine & J. Lucas, "Graph Metanetworks for Processing Diverse Neural Architectures," *ICLR*, 2024.

- V. Papayan, X. Y. Han & D. L. Donoho, "Prevalence of neural collapse during the terminal phase of deep learning training," *Proceedings of the National Academy of Sciences*, pp. 24652-24663, 2020.
- V. G. Satorras, E. Hoogeboom & M. Welling, "E(n) equivariant graph neural networks," *ICML*, 2021.
- K. T. Schütt, H. E. Sauceda, P. J. Kindermans, A. Tkatchenko & K. R. Müller, "SchNet—a deep learning architecture for molecules and materials," *The Journal of Chemical Physics*, 2018.
- N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, K. & P. Riley, "Tensor field networks: Rotation- and translation-equivariant neural networks for 3d point clouds," *arXiv preprint arXiv:1802.08219*, 2018.
- M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov & A. J. Smola, "Deep sets," *Advances in neural information processing systems*, 2017.