

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC



LỚP: MÁY TÍNH VÀ KHOA HỌC THÔNG TIN CLC
MÔN: THỰC TẬP THỰC TẾ PHÁT TRIỂN PHẦN
MỀM

Geometric Deep Learning

Sinh viên:

Nguyễn Đồng Trí Tâm -
21001581

Lecturers:

TS. Nguyễn Bích Thủy
TS. Vũ Tiến Dũng

Supervisors:

TS. Đỗ Đức Hạnh

Hà Nội - 2024

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA TOÁN - CƠ - TIN HỌC



LỚP: MÁY TÍNH VÀ KHOA HỌC THÔNG TIN CLC
MÔN: THỰC TẬP THỰC TẾ PHÁT TRIỂN PHẦN
MỀM

Geometric Deep Learning

Sinh viên:

Nguyễn Đồng Trí Tâm -
21001581

Lecturers:

TS. Nguyễn Bích Thủy
TS. Vũ Tiến Dũng

Supervisors:

TS. Đỗ Đức Hạnh

Hà Nội - 2024

Mục lục

Danh sách hình vẽ	ii
1 GIỚI THIỆU	1
2 CÁC CÔNG TRÌNH LIÊN QUAN	4
3 CƠ SỞ LÝ THUYẾT	5
3.1 Lý thuyết nhóm	5
3.2 Lý thuyết biểu diễn	7
3.3 Khái niệm về equivariance	8
3.4 Tensor field networks	9
3.4.1 Permutation equivariance	10
3.4.2 Translation equivariance	10
3.4.3 Rotation equivariance	10
3.4.4 Cấu trúc của các lớp TFN	11
3.4.4.1 Point convolution	12
3.4.4.2 Định nghĩa cấu trúc của layer	15
3.4.4.3 Self-interaction	16
3.4.4.4 Nonlinearity	16
4 GEOMETRIC DEEP LEARNING	18
4.1 Tổng quan về geometric deep learning	18

4.1.1	Mạng nơ-ron đồ thị và tiền thân hóa học của chúng	18
4.1.2	Các dữ liệu trong GDL	20
4.1.3	Dữ liệu đồ thị	22
4.1.4	Dữ liệu đa tạp	28
4.1.5	Dữ liệu dạng grid	29
4.1.6	Convolution và tính equivariant của nó với phép dịch chuyển	30
4.1.7	Mô hình hóa convolution cho không gian tổng quát	33
4.1.7.1	Tích chập trên miền số thực	33
4.1.7.2	Tích chập trên mặt cầu	34
4.1.7.3	Tích chập cho đa tạp	36
4.2	Geometric Deep learning cho AMR trong NLP	47
4.3	Phân loại hình 3D sử dụng TFN	56
4.3.1	Tổng quan bài toán	56
4.3.2	Dữ liệu sử dụng	57
4.3.3	Phương pháp thực nghiệm	58
4.3.4	Kết quả thực nghiệm và đánh giá	59
5	KẾT LUẬN	62
TÀI LIỆU THAM KHẢO		62

Danh sách hình vẽ

3.1	Hình ảnh minh họa về <i>invariant</i> và <i>equivariant</i> [4]	9
4.1	Ví dụ minh họa về các dạng dữ liệu trong GDL[4]	21
4.2	Các mô hình đã có và tính chất của chúng[4]	22
4.3	Ví dụ minh họa về biểu diễn đồ thị của một số dữ liệu thực thế[4]	23
4.4	Minh họa dữ liệu đồ thị trong GNN[4]	23
4.5	Ảnh minh họa về <i>invariant</i> và <i>equivariant</i> trong GNN[4] . . .	24
4.6	Cấu trúc của một mạng GNN, trong đó pooling là phép ánh xạ đưa một đồ thị thành một vector[4]	25
4.7	Cấu trúc của " <i>message passing</i> " trong GNN, trong đó ô vuông là phép tổng hợp đặc trưng của các lân cận như là phép lấy tổng, trung bình, ...[4]	25
4.8	Cấu trúc của hàm tổng hợp thông tin của GCN và GAT[4] . . .	26
4.9	Ảnh minh họa DeepSet[4]	27
4.10	Ảnh minh họa Transformer[4]	27
4.11	Ảnh minh họa dữ liệu đồ thị hình học và các phép toán đại số tác động nén nó[4]	28
4.12	Ảnh minh họa hai phương pháp trong <i>manifold learning</i> [4] . . .	29
4.13	Ảnh minh họa dữ liệu dạng <i>grid</i> và cấu trúc hàm tổng hợp đặc trưng[4]	30
4.14	Ảnh minh họa hàm tổng hợp đặc trưng đối với dữ liệu <i>grid</i> [4] .	30

4.15	Ảnh minh họa về ma trận <i>circulant</i> [4]	31
4.16	Ảnh minh họa về ma trận <i>Shift</i> [4]	32
4.17	Ví dụ minh họa về dữ liệu dạng <i>Sphere</i> [4]	35
4.18	Ảnh minh họa dữ liệu đa tạp và meshes[4]	37
4.19	Ảnh minh họa protein[4]	39
4.20	Ảnh minh họa tích đường đi của <i>filter</i> trên không gian Euclid[4]	40
4.21	Ảnh minh họa tích đường đi của <i>filter</i> trên <i>sphere</i> [4]	40
4.22	Ảnh minh họa tích đường đi của <i>filter</i> trên mặt <i>mobius</i> [4] . . .	41
4.23	Ảnh minh họa hai dạng <i>invariant</i> đối với dữ liệu <i>manifold</i> [4] . .	42
4.24	Ảnh minh họa filter trên đa tạp có tác động của một nhóm[4] .	44
4.25	Ứng dụng của tích chập cho manifold trong computer graphic[4]	45
4.26	Tổng quan quá trình đưa chuyển động của khuôn mặt vào trong đồ họa máy tính[4]	46
4.27	Ứng dụng tái cấu trúc lại hình dạng 3D của một vật thể từ một hình ảnh[4]	46
4.28	Ví dụ minh họa về AMR[2]	48
4.29	Biểu diễn AMR dạng text của ví dụ trong hình 4.28	49
4.30	Biểu diễn AMR của hai câu đồng nghĩa	51
4.31	Ý tưởng cấu trúc mới về AMR. (LLM functors nghĩa là sử dụng LLM để chuyển một câu sang AMR)	53
4.32	Ví dụ minh họa bằng cách dịch một câu sang ngôn ngữ khác. Những điểm trong hình tròn là các câu khác nhau nhưng cùng nghĩa. F, G, F là các phép dịch một ngôn ngữ này sang ngôn ngữ khác.	54
4.33	Motivation từ Curvature in Riemannian Geometry cho cấu trúc AMR trong NLP	55
4.34	Ảnh minh họa một số dữ liệu trong tập dữ liệu ModelNet10 . .	57
4.35	Biểu đồ theo dõi tính độ chính xác của mô hình trong quá trình training	59

4.36 Hình ảnh thống kê kết quả dự đoán của mô hình 61

Chương 1

GIỚI THIỆU

Một trong những giả thuyết tồn tại từ lâu trong deep learning, đó là nghiên cứu cách mà một mạng DL hoạt động dưới tác động của một nhóm biến dạng vô cùng nhỏ. Tôi quan tâm tới vấn đề này sau bài nói chuyện của Stephanee Mallet tại trường ĐHKHTN năm 2023. Cụ thể hơn, một lý thuyết deep learning tốt phải thỏa mãn một số tiêu chuẩn: -Bất biến dưới tác động của nhóm các biến dạng vô cùng nhỏ. Ví dụ, như bài toán nhận dạng, phải giữ vững được kết quả sau khi bức ảnh bị biến dạng rất ít. -Bất biến dưới tác động của nhóm các phép biến đổi, ví dụ bất biến dưới tác động của các phép quay, các nhóm biến đổi bảo toàn góc, các biến đổi tịnh tiến. Thông thường, người ta thường tiếp cận vấn đề này bằng cách tăng thêm dữ liệu, ví dụ như thay vì chỉ training một bức ảnh, người ta training tất cả các bức ảnh có thể bằng cách quay, bóp méo. Vấn đề ở đây, việc tang thêm dữ liệu sẽ làm chi phí training và kích thước mô hình trở nên cực lớn.

Chúng ta xét một bài toán cơ bản: Tìm tất cả các đa thức và là hàm chẵn đi qua 3 điểm. Thông thường, chúng ta sẽ lấy đối xứng các điểm này qua trục Oy, sau đó tìm một đa thức bậc 5 thông qua lý thuyết nội suy Lagrangian

$$f(x) = \sum_{i=0,5} f(x_i) \prod_{j \neq i} \frac{x-x_j}{x_i-x_j}$$

Tuy nhiên, kết quả chúng ta thu được là một đa thức bậc bốn. Tuy nhiên, nếu chúng ta chỉ xét các đa thức có dạng $ax^4 + bx^2 + c$ thì việc tính toán sẽ đơn giản hơn rất nhiều do tính đối xứng của đa thức. Hoàn toàn tương tự, câu chuyện cũng xảy ra với lý thuyết regression phi tuyến và do đó cả với lý thuyết deep learning.

Đối với deep learning, chúng tôi dự tính cách tiếp cận như sau: 1-Thiết kế các lý thuyết mạng bắt biến với phép đối xứng. 2-Thực hiện việc traing thông qua các phiên bản gradient descent mà bắt biến với phép đối xứng. Ở đây, thay vì gradient descent cổ điển, ta tìm $\text{Argmin}_v f(x + v)$ Trong đó loss function bắt biến dưới tác động của nhóm đối xứng.

Để thực hiện được việc này, chúng tôi nghiên cứu các kết hợp các lãnh vực toán học hiện đại, cụ thể hơn, lý thuyết biểu diễn nhóm, lý thuyết nhóm Lie và đại số Lie, lý thuyết bất biến, hình học Riemanian, giải tích điều hòa không giao hoán với lý thuyết deep learning.

Vì rằng ví dụ không tầm thường đầu tiên của lý thuyết là nhóm các phép quay $SO(3)$, nên chúng tôi nghiên cứu giải tích điều hòa trên nhóm $SO(3)$, và vì vậy các nghiên cứu của chúng tôi có ứng dụng trực tiếp cho quantum mechanics trong việc phân tích phổ nguyên tử. Đồng thời, vì không gian biểu diễn của $SO(3)$ là trên các phân thớ xoắn của S^2 , nên từ đây chúng ta đưa tới lý thuyết giải tích điều hòa trên không gian thuần nhất, và lý thuyết deep learning trên không gian thuần nhất. Cụ thể hơn, chúng ta giới thiệu convolution on the homogeneous spacec, cũng như lý thuyết về filter/ kernel liên quan.

Tiếp theo, bởi vì bản chất của các không gian thuần nhất thường có cấu trúc của các đa tạp Riemannian, nên hình học Riemanian xuất hiện trong lý thuyết

của chúng tôi.

Cuối cùng, được khích lệ bởi lý thuyết holonomy trong hình học Riemanian, chúng tôi nghiên cứu lý thuyết đối xứng xuất hiện một cách tự nhiên trong NLP, và đề xuất ra lý thuyết symmetry trên các AMR graph, và cấu trúc của probabilistic category trong NLP. Đây là các kết quả chưa từng xuất hiện trong các tài liệu liên quan.

Chương 2

CÁC CÔNG TRÌNH LIÊN QUAN

Chương 3

CƠ SỞ LÝ THUYẾT

3.1 Lý thuyết nhóm

Lý thuyết nhóm là một nhánh trong toán học có mục đích là nghiên cứu về các cấu trúc đại số được gọi là các nhóm. Một nhóm là một tập các tác động nào đó như là tập các phép hoán vị của n phần tử thì được gọi là một nhóm. Một nhóm được định nghĩa bởi một bộ (G, \cdot) trong đó G là một tập các phần tử của nhóm và \cdot là phép toán của nhóm, phép toán này sẽ cho chúng ta biết các phần tử $g \in G$ kết hợp như thế nào. Phép toán \cdot cần thỏa mãn một số tính chất sau:

1. Tính đóng: nhóm G đóng với phép toán \cdot , nghĩa là với mọi $g_1, g_2 \in G$ thì ta có $g_1 \cdot g_2 \in G$
2. Phần tử đơn vị: tồn tại một phần tử e sao cho với mỗi $g \in G$ thì $e \cdot g = g \cdot e = g$
3. Nghịch đảo: với mọi $g \in G$ thì sẽ tồn tại một phần tử $g^{-1} \in G$ sao cho

$$g \cdot g^{-1} = e$$

4. Tính kết hợp: với mọi tập các phần tử $g_1, g_2, g_3 \in G$, ta có $(g_1 \cdot g_2) \cdot g_3 = g_1 \cdot (g_2 \cdot g_3)$

Ví dụ: Trong không gian R^2 , ta có phép quay một góc θ trong không gian này là một ma trận:

$$R(\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$$

Như vậy, tập hợp tất cả các góc θ có thể sẽ tạo thành một nhóm các phép quay trên R^2 và nhóm này được gọi là $SO(2)$. Có thể thấy $SO(2)$ đều thỏa mãn cả bốn tính chất của một nhóm:

1. Tính đóng (Closure):

- Nếu $R(\theta_1)$ và $R(\theta_2)$ là hai ma trận quay trong $SO(2)$, thì tích của chúng $R(\theta_1)R(\theta_2)$ cũng là một ma trận quay trong $SO(2)$.

- Thật vậy, tích của hai ma trận quay:

$$\begin{aligned} R(\theta_1)R(\theta_2) &= \begin{pmatrix} \cos \theta_1 & -\sin \theta_1 \\ \sin \theta_1 & \cos \theta_1 \end{pmatrix} \begin{pmatrix} \cos \theta_2 & -\sin \theta_2 \\ \sin \theta_2 & \cos \theta_2 \end{pmatrix} \\ &= \begin{pmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) \end{pmatrix} \end{aligned}$$

- Như vậy, $R(\theta_1 + \theta_2)$ cũng là một ma trận quay trong $SO(2)$.

2. Phần tử đơn vị (Identity element):

- Ma trận đơn vị trong $SO(2)$ là ma trận $R(0)$, tương ứng với góc quay 0 độ:

$$R(0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- Khi nhân bất kỳ ma trận quay nào với ma trận đơn vị này, kết quả vẫn là ma trận quay ban đầu.

3. Phần tử nghịch đảo (Inverse element):

- Với mỗi ma trận quay $R(\theta)$, phần tử nghịch đảo của nó là $R(-\theta)$, tức là quay ngược lại góc θ :

$$R(\theta)R(-\theta) = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} \cos (-\theta) & -\sin (-\theta) \\ \sin (-\theta) & \cos (-\theta) \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- Như vậy, mỗi ma trận quay trong $SO(2)$ đều có một ma trận nghịch đảo.

4. Tính kết hợp (Associativity):

- Tích của các ma trận quay tuân theo tính kết hợp:

$$(R(\theta_1)R(\theta_2))R(\theta_3) = R(\theta_1)(R(\theta_2)R(\theta_3))$$

- Điều này xuất phát từ tính chất kết hợp của phép nhân ma trận.

3.2 Lý thuyết biểu diễn

Giả sử ta có một nhóm G bất kỳ, khi đó, ta muốn biết cách mà nhóm G tác động lên một không gian R^d là như thế nào. Do đó, lý thuyết biểu diễn bắt đầu xuất hiện để giải thích cho điều này.

Định nghĩa: Một biểu diễn tuyến tính ρ của một nhóm compact G trên một không gian vectơ (gọi là không gian biểu diễn) R^d là một tập các phép ánh

xạ từ G sang nhóm $GL(R^d)$ (nhóm này bao gồm tập các ma trận khả nghịch trong R^d). Cụ thể, đó là một phép ánh xạ $\rho : G \rightarrow R^{d \times d}$ sao cho:

$$\rho(g_1g_2) = \rho(g_1)\rho(g_2) \quad \forall g_1, g_2 \in G$$

$$\rho(e) = I_d$$

Một cách nói khác thì ρ là một ma trận khả nghịch có kích thước là $d \times d$ với d được hiểu là kích thước của biểu diễn.

Chúng ta có kết hợp hai biểu diễn để có thể tạo ra một biểu diễn lớn hơn thông qua **tổng trực tiếp**.

Cho các biểu diễn $\rho_1 : G \rightarrow \mathbb{R}^{d_1 \times d_1}$ và $\rho_2 : G \rightarrow \mathbb{R}^{d_2 \times d_2}$, **tổng trực tiếp** $\rho_1 \oplus \rho_2 : G \rightarrow \mathbb{R}^{(d_1+d_2) \times (d_1+d_2)}$ được định nghĩa như sau

$$(\rho_1 \oplus \rho_2)(g) = \begin{pmatrix} \rho_1(g) & 0 \\ 0 & \rho_2(g) \end{pmatrix}$$

Tác động này được tạo bởi hai tác động độc lập là ρ_1 và ρ_2 trên các không gian con trực giao \mathbb{R}^{d_1} và \mathbb{R}^{d_2} của $\mathbb{R}^{d_1+d_2}$.

3.3 Khái niệm về equivariance

Ở đây, báo cáo sẽ sử dụng ký hiệu biểu diễn của một nhóm khác với phần trước, ta gọi biểu diễn D của một nhóm G là một hàm ánh xạ từ G thành một ma trận vuông sao cho:

$$D(g)D(h) = D(gh) \quad \forall g, h \in G$$

Một hàm $\mathcal{L} : \mathcal{X} \rightarrow \mathcal{Y}$ (với mọi không gian vector \mathcal{X} và \mathcal{Y}) là một ánh xạ *equivariant* đối với nhóm G và hai biểu diễn $D^{\mathcal{X}}$ và $D^{\mathcal{Y}}$ khi và chỉ khi với mọi $g \in G$,

$$\mathcal{L} \circ D^{\mathcal{X}}(g) = D^{\mathcal{Y}}(g) \circ \mathcal{L}$$

Ánh xạ *invariant* là một trường hợp của *equivariant* khi $D^{\mathcal{Y}}(g)$ là ma trận đơn vị với mọi $g \in G$.

\mathfrak{G} -invariance $f(\rho(g)x) = f(x)$ \mathfrak{G} -equivariance $f(\rho(g)x) = \rho(g)f(x)$



Hình 3.1: Hình ảnh minh họa về *invariant* và *equivariant*[4]

3.4 Tensor field networks

Tensor field networks (TFN) tác động trên các điểm với các đặc trưng liên quan. Một lớp \mathcal{L} lấy một tập hợp hữu hạn \mathcal{S} là các vector trong \mathbb{R}^3 và một vector trong \mathcal{X} tại mỗi điểm trong \mathcal{S} và đầu ra là một vector trong \mathcal{Y} tại mỗi điểm trong \mathcal{S} , trong đó \mathcal{X} và \mathcal{Y} là các không gian vector[10]. Báo cáo viết điều này như sau

$$\mathcal{L}(\vec{r}_a, x_a) = (\vec{r}_a, y_a)$$

trong đó $\vec{r}_a \in \mathbb{R}^3$ là các tọa độ điểm và $x_a \in \mathcal{X}, y_a \in \mathcal{Y}$ là các vector đặc trưng (với a là một chỉ mục trên các điểm trong \mathcal{S}). Sự kết hợp này của \mathbb{R}^3 với một

không gian vector khác có thể được viết là $\mathbb{R}^3 \oplus \mathcal{X}$, trong đó \oplus là phép nối (concatenation).

3.4.1 Permutation equivariance

$$\text{Điều kiện: } \mathcal{L} \circ P_\sigma = P_\sigma \circ \mathcal{L}$$

Trong đó: $P_\sigma(\vec{r}_a, x_a) := (\vec{r}_{\sigma(a)}, x_{\sigma(a)})$ và σ là phép hoán vị các điểm mà các chỉ số tham chiếu đến.

3.4.2 Translation equivariance

$$\text{Điều kiện: } \mathcal{L} \circ \mathcal{T}_{\vec{t}} = \mathcal{T}_{\vec{t}} \circ \mathcal{L}$$

trong đó $\mathcal{T}_{\vec{t}}(\vec{r}_a, x_a) := (\vec{r}_a + \vec{t}, x_a)$. Điều kiện này tương tự như điều kiện tính tiền equivariance cho CNN.

3.4.3 Rotation equivariance

Nhóm các phép quay 3D (thực) được gọi là $SO(3)$, một đa tạp có thể được tham số hóa bằng 3 số (xem Goodman và Wallach[6]). Cho $D^\mathcal{X}$ là một biểu diễn của $SO(3)$ trên không gian vectơ \mathcal{X} (và $D^\mathcal{Y}$ trên \mathcal{Y}). Tác động với $g \in SO(3)$ trên $\vec{r} \in \mathbb{R}^3$ chúng ta viết là $R(g)\vec{r}$, và tác động trên $x \in \mathcal{X}$ cho kết quả là $D^\mathcal{X}(g)x$.

Điều kiện:

$$\mathcal{L} \circ [R(g) \oplus D^\mathcal{X}(g)] = [R(g) \oplus D^\mathcal{Y}(g)] \circ \mathcal{L} \quad (3.1)$$

trong đó $[R(g) \oplus D^{\mathcal{X}}(g)](\vec{r}_a, x_a) = (R(g)\vec{r}_a, D^{\mathcal{X}}(g)x_a)$. (Đối với các layer trong bài báo này, chỉ có tác động của $D^{\mathcal{Y}}(g)$ trên \mathcal{Y} là không tầm thường, vì vậy chúng ta sẽ sử dụng quy ước bỏ qua đầu ra tầng \mathbb{R}^3 trong các phương trình của chúng ta.)

TFN đạt được tính equivariant với phép quay cục bộ bằng cách giới hạn các bộ lọc tích chập của chúng ta ở một dạng cụ thể. Các đặc trưng có các loại khác nhau tương ứng với việc chúng biến đổi dưới dạng vô hướng, vectơ hoặc tensor bậc cao hơn[10].

TFN phân tích các biểu diễn thành các biểu diễn bất khả quy để đơn giản hóa phân tích của chúng ta. Các biểu diễn bất khả quy của $SO(3)$ có kích thước $2l + 1$ cho $l \in \mathbb{N}$ (bao gồm $l = 0$) và là *unitary*. Báo cáo sẽ sử dụng thuật ngữ “bậc quay” để chỉ l trong biểu thức này. Các bậc quay $l = 0, 1, 2$ tương ứng với vô hướng, vectơ trong không gian 3 chiều và ma trận đối xứng không dấu vết, tương ứng.

Các phần tử nhóm được biểu diễn bằng $D^{(l)}$, được gọi là *ma trận Wigner-D* (xem Gilmore [5]); chúng ánh xạ các phần tử của $SO(3)$ thành ma trận $(2l+1) \times (2l+1)$ chiều. Đối với vô hướng và vectơ không gian 3 chiều, *ma trận Wigner-D* (thực) là

$$D^{(0)}(g) = 1 \quad \text{và} \quad D^{(1)}(g) = R(g).$$

3.4.4 Cấu trúc của các lớp TFN

Đầu vào và đầu ra của TFN là một tập hữu hạn \mathcal{S} của các điểm trong \mathbb{R}^3 và một vector trong một biểu diễn của $SO(3)$ liên quan đến mỗi điểm[10].

Chúng ta phân tích biểu diễn này thành các biểu diễn bất khả quy. Nói chung, có nhiều trường hợp của mỗi biểu diễn bất khả quy bậc quay l . Chúng tương tự với những gì thường được gọi là "độ sâu" của tích chập trong CNN tiêu chuẩn, vì vậy chúng ta sẽ đề cập đến những trường hợp khác nhau này là các kênh (channels). Chúng ta triển khai đối tượng này $V_{acm}^{(l)}$ như một từ điển với key l (bậc quay) của các mảng đa chiều, mỗi mảng có hình dạng $[|S|, n_l, 2l + 1]$ (trong đó n_l là số lượng kênh) tương ứng với [chỉ số điểm, chỉ số kênh, chỉ số biểu diễn]. Xem hình ?? cho một ví dụ về cách mã hóa một hệ thống đơn giản trong ký hiệu này.

Báo cáo sẽ định nghĩa ba tầng TFN và chứng minh rằng chúng là equivariant. Tất cả những tầng này sẽ rõ ràng là invariant hoán vị và equivariant tịnh tiến, vì vậy để chứng minh rằng một tầng là equivariant, chúng ta chỉ cần chứng minh tính equivariant với phép quay cho một bậc quay tùy ý. Điều này yêu cầu cho thấy rằng khi đám mây điểm xoay và các đặc trưng đầu vào được biến đổi thành tensor, thì đầu ra đặc trưng cũng biến đổi thành tensor.

3.4.4.1 Point convolution

Lớp này là một thành phần cốt lõi của TFN. Tại mỗi lớp, đầu vào sẽ có cấu trúc là $V_{acm}^{(l)}$. Point convolution thực hiện phép toán giống nhau cho mỗi điểm a và lấy mọi điểm còn lại như đầu vào. Ở đây, ta ký hiệu \vec{r} là một vector trong không gian 3 chiều, \hat{r} là vector \vec{r} được chuẩn hóa có độ dài bằng 1 và r là độ dài của \vec{r} .

3.4.4.1.1 Spherical harmonics và filters

Spherical harmonics $Y_m^{(l)}$ là các hàm từ các điểm trên một hình cầu đến các số phức hoặc số thực (trong đó l là một số nguyên không âm và m là bất kỳ số nguyên nào từ $-l$ đến l , bao gồm cả $-l$ và l). Spherical harmonics (thực) cho $l = 0$ (vô hướng) và $l = 1$ (vectơ không gian 3 chiều) là

$$Y^{(0)}(\hat{r}) \propto 1 \quad \text{và} \quad Y^{(1)}(\hat{r}) \propto \hat{r}.$$

Các hàm này equivariant với $SO(3)$; nghĩa là, với mọi $g \in SO(3)$ và \hat{r} ,

$$Y_m^{(l)}(R(g)\hat{r}) = \sum_{m'} D_{mm'}^{(l)}(g) Y_{m'}^{(l)}(\hat{r}).$$

Để thiết kế một phép tích chập điểm equivariant với phép quay, chúng ta muốn các bộ lọc equivariant với phép quay. Để các bộ lọc của chúng ta equivariant với phép quay, chúng ta giới hạn chúng ở dạng sau[10]:

$$F_{cm}^{(l_f, l_i)}(\hat{r}) = R_c^{(l_f, l_i)}(r) Y_m^{(l_f)}(\hat{r}) \quad (3.2)$$

(trong đó l_i và l_f là các số nguyên không âm tương ứng với bậc quay của đầu vào và bộ lọc, tương ứng và $R_c^{(l_f, l_i)} : \mathbb{R}^{\geq 0} \rightarrow \mathbb{R}$ là các hàm được học, chứa hầu hết các tham số trong một TFN). Các bộ lọc có dạng này kế thừa thuộc tính biến đổi của các *spherical harmonic* trong các phép quay vì $R(r)$ là một vô hướng trong m . Lựa chọn giới hạn bộ lọc này tương tự như việc sử dụng các *circular harmonics* trong Worrall và các đồng nghiệp[13] (mặc dù chúng ta không có tương tự với độ lệch pha do tính không giao hoán của $SO(3)$).

3.4.4.1.2 Kết hợp biểu diễn bằng tích tensor

Bộ lọc và layer đầu vào đều kế thừa các biểu diễn của $SO(3)$ (nghĩa là, cả hai đều mang các chỉ số l và m). Để tạo ra đầu ra mà chúng ta có thể đưa vào các tầng phía sau, chúng ta cần kết hợp đầu vào tầng và bộ lọc theo cách mà đầu ra cũng biến đổi một cách thích hợp (bằng cách kế thừa biểu diễn của $SO(3)$)[10].

Tích tensor của các biểu diễn là một quy tắc để kết hợp hai biểu diễn $D^{\mathcal{X}}$ và $D^{\mathcal{Y}}$ để nhận được một biểu diễn khác $D^{\mathcal{X}} \otimes D^{\mathcal{Y}}$ trên không gian vectơ $\mathcal{X} \otimes \mathcal{Y}$. Thuộc tính quan trọng của tích tensor là nó equivariant:

$$D^{\mathcal{X}} \otimes D^{\mathcal{Y}} = D^{\mathcal{X} \otimes \mathcal{Y}}$$

Bây giờ hãy xem xét tích tensor của hai biểu diễn có bậc l_1 và l_2 (trong đó chúng ta sử dụng quy ước ký hiệu thông thường là chỉ sử dụng l để chỉ không gian vectơ $(2l + 1)$ chiều biểu diễn bậc quay l). Đối với các biểu diễn không thể rút gọn với $u^{(l_1)} \in l_1$ và $v^{(l_2)} \in l_2$, $(u^{(l_1)} \otimes v^{(l_2)}) \in l_1 \otimes l_2$ có thể được tính bằng cách sử dụng các hệ số Clebsch–Gordan (xem Griffiths [7]) (ký hiệu là C):

$$(u \otimes v)_m^{(l)} = \sum_{m_1=-l_1}^{l_1} \sum_{m_2=-l_2}^{l_2} C_{(l_1, m_1), (l_2, m_2)}^{(l, m)} u_{m_1}^{(l_1)} v_{m_2}^{(l_2)}$$

Tích tensor này tạo ra các giá trị khác không chỉ cho l nằm trong khoảng từ $|l_1 - l_2|$ đến $(l_1 + l_2)$ (bao gồm cả $|l_1 - l_2|$ và $(l_1 + l_2)$) (m là bất kỳ số nguyên nào từ $-l$ đến l , bao gồm cả $-l$ và l). Các hệ số Clebsch–Gordan (thực) cho $l = 0$ và $l = 1$ chỉ là những cách quen thuộc để kết hợp các vô hướng và vectơ:

Đối với $1 \otimes 1 \rightarrow 0$ và $1 \otimes 1 \rightarrow 1$,

$$C_{(1,i),(1,j)}^{(0,0)} \propto \delta_{ij} \quad C_{(1,j),(1,k)}^{(1,i)} \propto \epsilon_{ijk}$$

đó là tích vô hướng và tích có hướng cho các vectơ 3D, tương ứng. Trường hợp $0 \otimes 0 \rightarrow 0$ chỉ là phép nhân thông thường của hai vô hướng và $0 \otimes 1 \rightarrow 1$ và $1 \otimes 0 \rightarrow 1$ tương ứng với phép nhân vô hướng của một vectơ.

3.4.4.2 Định nghĩa cấu trúc của layer

Một đầu vào cho trước sẽ kế thừa một biểu diễn, một bộ lọc sẽ kế thừa một biểu diễn khác và cùng nhau chúng tạo ra đầu ra ở nhiều bậc quay có thể[10] . Chúng ta có thể kết hợp mọi thứ lại với nhau trong định nghĩa tầng tích chập theo điểm của chúng ta:

$$\mathcal{L}_{acm_o}^{(l_o)}(\vec{r}_a, V_{acm_i}^{(l_i)}) := \sum_{m_f, m_i} C_{(l_f, m_f), (l_i, m_i)}^{(l_o, m_o)} \sum_{b \in \mathcal{S}} F_{cm_f}^{(l_f, l_i)}(\vec{r}_{ab}) V_{bcm_i}^{(l_i)} \quad (3.3)$$

(trong đó $\vec{r}_{ab} := \vec{r}_a - \vec{r}_b$ và các chỉ số i, f và o biểu thị cho các biểu diễn của đầu vào, bộ lọc và đầu ra, tương ứng). Một phép tích chập điểm của một bộ lọc l_f trên một đầu vào l_i mang lại đầu ra ở $2\min(l_i, l_f) + 1$ bậc quay l_o khác nhau (một cho mỗi số nguyên từ $|l_i - l_f|$ đến $(l_i + l_f)$, bao gồm cả $|l_i - l_f|$ và $(l_i + l_f)$), mặc dù khi thiết kế một mạng cụ thể, chúng ta có thể chọn không tính toán hoặc sử dụng một số đầu ra đó[10].

Tính equivariant của bộ lọc F (phương trình 3.2) và tính equivariant của các hệ số Clebsch–Gordan cũng ngụ ý rằng các phép tích chập điểm là equivariant.

3.4.4.3 Self-interaction

Báo cáo làm theo Schütt et al. [9] trong việc sử dụng các phép tích chập điểm để chia tỷ lệ các vectơ đặc trưng theo từng phần tử và sử dụng các lớp *Self-interaction* để trộn các thành phần của vectơ đặc trưng tại mỗi điểm. Các layer *Self-interaction* tương tự như các tích chập 1×1 và chúng hoạt động giống như các bộ lọc $l = 0$ (vô hướng):

$$\sum_{c'} W_{cc'}^{(l)} V_{ac'm}^{(l)}$$

Nói chung, mỗi bậc quay có trọng số khác nhau vì có thể có số lượng kênh khác nhau tương ứng với bậc quay đó. Tuy nhiên, cùng một trọng số được sử dụng cho mọi m cho một bậc cho trước; điều này rất cần thiết để duy trì tính equivariant. Đối với $l = 0$, chúng ta cũng có thể sử dụng độ lệch.

Các ma trận D giao hoán với ma trận trọng số W vì W không có chỉ số biểu diễn m , vì vậy tầng này là equivariant với $l > 0$. Tính equivariant với $l = 0$ là đơn giản vì $D^{(0)} = 1$.

3.4.4.4 Nonlinearity

Lớp phi tuyến (*nonlinearity*) hoạt động như một phép biến đổi vô hướng trong không gian l (nghĩa là, dọc theo chiều m). Đối với các kênh $l = 0$, chúng ta có thể sử dụng

$$\eta^{(0)}(V_{ac}^{(0)} + b_c^{(0)}) \quad \text{và} \quad \eta^{(l)}(\|V\|_{ac}^{(l)} + b_c^{(l)}) V_{acm}^{(l)} \quad \text{trong đó} \quad \|V\|_{ac}^{(l)} := \sqrt{\sum_m |V_{acm}^{(l)}|^2}$$

cho một số hàm $\eta^{(l)} : \mathbb{R} \rightarrow \mathbb{R}$ (có thể khác nhau cho mỗi l) và độ lệch $b_c^{(l)}$. Lưu ý rằng

$$\|D(g)V\| = \|V\|$$

vì D là một biểu diễn đơn vị. Do đó, tầng này là một phép biến đổi vô hướng trong chỉ số biểu diễn m , vì vậy nó equivariant.

Chương 4

GEOMETRIC DEEP LEARNING

4.1 Tổng quan về geometric deep learning

4.1.1 Mạng nơ-ron đồ thị và tiền thân hóa học của chúng

Hóa học là một nhánh của khoa học tự nhiên nhằm nghiên cứu về thành phần, cấu trúc, tính chất, và sự thay đổi của vật chất. Các chủ đề chính trong hóa học là nguyên tố, hợp chất, nguyên tử, phân tử, và các phản ứng hóa học. Để nghiên cứu về các lĩnh vực này, các nhà hóa học cần phải nghiên cứu phản ứng hay công thức phân tử, ... Tuy nhiên, với sự phát triển của công nghệ thì giờ ta hoàn toàn có thể sử dụng tin học trong hóa học. Đầu tiên, ta sẽ bắt đầu từ tin học trong hóa học thời sơ khai.

Nó bắt đầu bằng nguồn gốc của hóa tin học, được đánh dấu bằng các ví dụ như "Chemisches Zentralblatt" (tạp chí tóm tắt hóa học đầu tiên), "Beilstein

Handbuch" (một tập hợp toàn diện các hợp chất hữu cơ) và sự xuất hiện của Dịch vụ Tóm tắt Hóa học (CAS). Những nỗ lực sơ khai này đã nhấn mạnh nhu cầu sắp xếp và quản lý thông tin hóa học [4].

Thẻ đục lỗ cho máy tính ban đầu biểu thị phương pháp sử dụng ban đầu máy tính trong hóa học, thiết lập nền tảng cho các phương pháp dựa trên dữ liệu hơn nữa[4].

Tuy nhiên, việc biểu diễn và so sánh cấu trúc phân tử vẫn còn là một thách thức. Sơ đồ nêu bật những hạn chế của "mã hóa hóa học" ban đầu, vốn thường không nắm bắt được sự tương đồng về cấu trúc giữa các phân tử[4]. Một ví dụ được đưa ra với hai phân tử được biểu diễn bằng các ký hiệu tuyến tính khác nhau trong khi có chung cấu trúc.

Tiếp theo, báo cáo đưa ra mối liên hệ giữa lý thuyết đồ thị và hóa học. Năm 1878, James J. Sylvester đã ghi nhận mối quan hệ giữa hóa học và lý thuyết đồ thị[4]. Ông đã giới thiệu thuật ngữ "đồ thị" trong bối cảnh hóa học, nhận ra rằng cấu trúc phân tử có thể được biểu diễn hiệu quả bằng đồ thị. Ý tưởng có tầm nhìn xa này đã đặt nền móng cho các biểu diễn phân tử dựa trên đồ thị.

Dưới đây là một số nhà khoa học đã đóng góp để tạo ra mạng nơ-ron đồ thị đầu tiên:

- Alessandro Sperduti (1994): Làm việc trên việc gán nhãn RAAM, một kiến trúc mạng nơ-ron cho dữ liệu có cấu trúc đồ thị[4].
- Christoph Goller (1996): Khám phá lan truyền ngược thông qua cấu trúc, điều này rất quan trọng để huấn luyện GNN[4].
- Kristian Kersting và các đồng nghiệp: Giới thiệu thuật ngữ "Mạng nơ-ron đồ thị" và phát triển các kiến trúc GNN hơn nữa[4].

- Yujia Li (2015): Làm việc trên Gated GNN, giúp nâng cao việc biểu diễn đồ thị[4].

Tài liệu tham khảo [4] đã nhấn mạnh công trình của David Duvenaud và Justin Gilmer, những người đã đi tiên phong trong việc sử dụng GNN để dự đoán thuộc tính hóa học. Các phương pháp của họ, sử dụng GNN truyền thông điệp, đã cho thấy kết quả đầy hứa hẹn trong việc dự đoán các thuộc tính phân tử, vượt trội hơn các phương pháp tính toán truyền thống như lý thuyết hàm mật độ (DFT) về tốc độ trong khi vẫn duy trì độ chính xác.

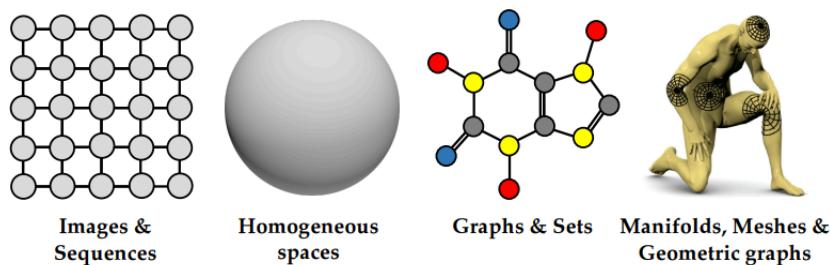
Cuối cùng, AlphaFold - một mô hình AI có thể dự đoán cấu trúc protein với độ chính xác đáng chú ý[4]. Sự kiện này được mệnh danh là khoảnh khắc "ImageNet" của sinh học cấu trúc, ngụ ý một bước đột phá mang tính chuyển đổi trong lĩnh vực này.

4.1.2 Các dữ liệu trong GDL

Thông thường chúng ta hay làm việc với dữ liệu dạng Euclidean. Tuy nhiên, trong GDL có một số dạng dữ liệu khác phức tạp hơn mà chúng ta thường chưa được làm việc với chúng. Điều này dễ hiểu bởi vì những dạng dữ liệu này phức tạp hơn và ta sẽ phải dùng đến nhiều kiến thức toán học hơn như lý thuyết nhóm, lý thuyết biểu diễn, giải tích điều hòa, ... Dưới đây là các dạng dữ liệu trong GDL:

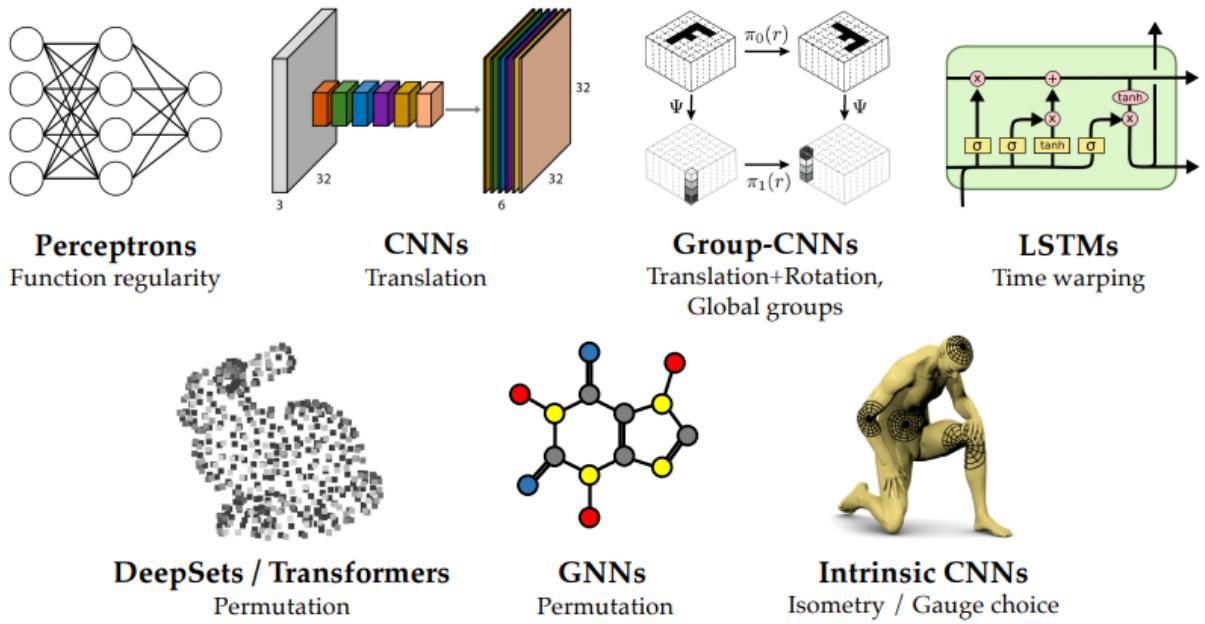
1. Hình ảnh và chuỗi: Dạng dữ liệu này được biểu diễn bằng lưới hai chiều (2D) gồm các điểm ảnh. Mỗi điểm ảnh lưu trữ thông tin về màu sắc và cường độ sáng. Các chuỗi, như dữ liệu chuỗi thời gian, cũng được biểu diễn trong dạng này, với thông tin được sắp xếp theo trình tự tuyến tính.

2. Không gian đồng nhất (Homogeneous spaces): Dạng dữ liệu này biểu thị không gian liên tục mà tại đó mọi điểm đều có chung một tập hợp các thuộc tính. Nói cách khác, không có điểm "đặc biệt" hay khác biệt nào trong không gian đồng nhất.
3. Đồ thị và tập hợp (Graphs and Sets): Dạng dữ liệu này bao gồm các nút (đỉnh) và các cạnh kết nối chúng. Các nút đại diện cho các thực thể và các cạnh thể hiện mối quan hệ giữa chúng. Các tập hợp là các bộ sưu tập các đối tượng không có thứ tự cụ thể.
4. Đa tạp, lưới và đồ thị hình học (Manifolds, Meshes and Geometric graphs): Dạng dữ liệu này biểu thị các cấu trúc phức tạp có thể được nhúng trong không gian chiều cao hơn.
 - Đa tạp: các không gian liên tục, cục bộ giống như không gian Euclidean nhưng có thể có cấu trúc tôpô phức tạp hơn ở quy mô toàn cục.
 - Lưới: các biểu diễn rời rạc của các bề mặt sử dụng các đa giác hoặc đa diện.
 - Đồ thị hình học: các đồ thị mà các nút và cạnh được liên kết với thông tin hình học, chẳng hạn như vị trí trong không gian.



Hình 4.1: Ví dụ minh họa về các dạng dữ liệu trong GDL[4]

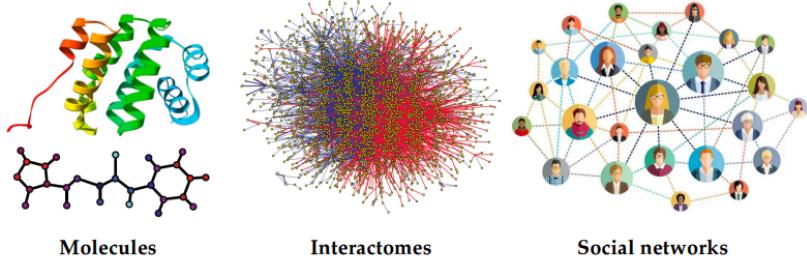
Hình 4.2 là một số mô hình đã có để xử lý các bài toán khác nhau trong GDL:



Hình 4.2: Các mô hình đã có và tính chất của chúng[4]

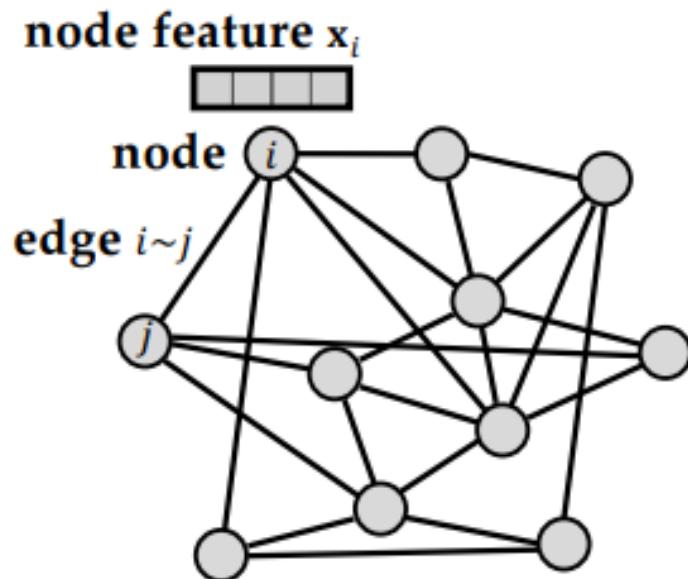
4.1.3 Dữ liệu đồ thị

Trong thực tế, có rất nhiều bài toán có thể đưa về dạng đồ thị như bài toán liên quan đến phân tử hóa học, mạng xã hội, ... Ta có thể hiểu đồ thị là một hệ thống của các mối liên hệ và sự tương tác - điều này được thể hiện rõ nhất ở cạnh giữa hai đỉnh trong một đồ thị. Để tường minh hơn, Hình 4.3 sẽ cho một ví dụ cụ thể hơn:



Hình 4.3: Ví dụ minh họa về biểu diễn đồ thị của một số dữ liệu thực thế[4]

Tuy nhiên, ngoài tập các đỉnh và tập các cạnh của đồ thị ra thì trong mạng nơ-ron đồ thị (GNN) còn thêm một biểu diễn nữa là đặc trưng của các đỉnh. Các đặc trưng này sẽ thể hiện tính chất của các đỉnh ví dụ như đối với mạng xã hội thì là tuổi tác, số lượng bạn bè, ...

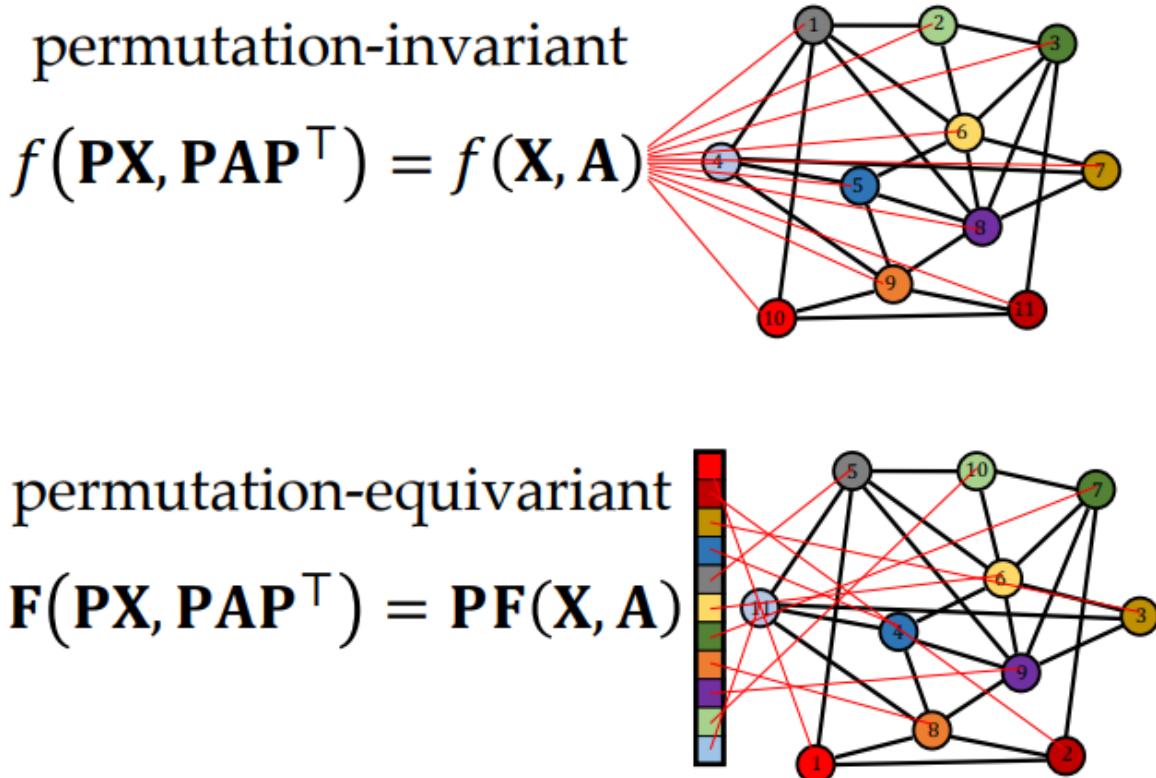


Hình 4.4: Minh họa dữ liệu đồ thị trong GNN[4]

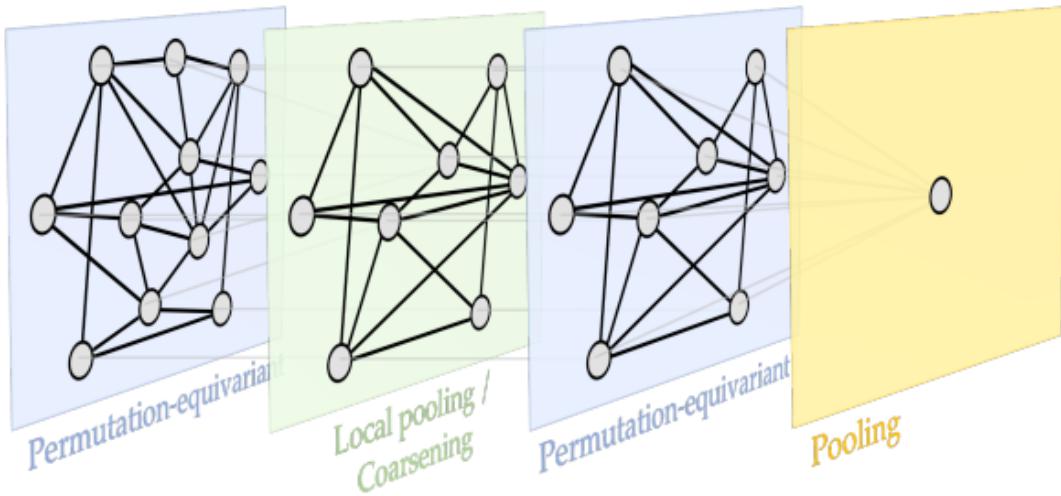
Trong GNN, ta có thể đánh số thứ tự các đỉnh tùy ý mà không làm thay đổi cấu trúc của đồ thị, do đó ta cần thiết kế một hàm có đầu vào là đặc trưng các đỉnh và ma trận kè sao cho nó phải bất biến với phép đánh thứ tự các đỉnh.

Hay nói cách khác, hàm này phải là một hàm *invariant permutation*. Điều này sẽ giúp cho đặc trưng của các đỉnh sau khi được tổng hợp không bị thay đổi khi ta đánh lại thứ tự các đỉnh.

Tuy nhiên, mặc dù hàm tổng hợp đặc trưng của các đỉnh là bất biến với phép hoán vị nhưng thứ tự các đặc trưng sau khi bị hoán vị vẫn phải thay đổi. Cụ thể hơn, nếu như thứ tự đặc trưng đầu vào bị thay đổi thì thứ tự đặc trưng sau khi tổng hợp đặc trưng cho các đỉnh cũng phải thay đổi, ví dụ: đồ thị có 3 đỉnh thì ta đánh thứ tự là 1, 2, 3 tuy nhiên khi ta hoán vị vị trí các đỉnh là 2, 1, 3 thì sau khi tổng hợp đặc trưng thì thứ tự cũng phải là 2, 1, 3 chứ không phải là 1, 2, 3. Điều này thể hiện tính *equivariant* của GNN.



Hình 4.5: Ảnh minh họa về *invariant* và *equivariant* trong GNN[4]



Hình 4.6: Cấu trúc của một mạng GNN, trong đó pooling là phép ánh xạ đưa một đồ thị thành một vector[4]

Tiếp theo, báo cáo sẽ đưa ra cấu trúc của một hàm tổng hợp đặc trưng cho một đỉnh trong GNN. Ý tưởng trung của GNN là đặc trưng của một đỉnh sẽ bị ảnh hưởng bởi những đỉnh lân cận và chính nó. Do đó, các cấu trúc GNN khác nhau chỉ là các cách tổng hợp đặc trưng khác nhau và chúng được gọi chung bằng một cái tên là "*message passing*".

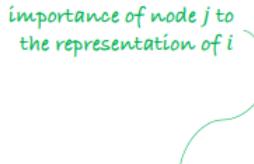
$$f(\mathbf{x}_i) = \phi \left(\mathbf{x}_i, \bigcup_{j \in \mathcal{N}_i} \psi(\mathbf{x}_i, \mathbf{x}_j) \right)$$

“message passing”

Hình 4.7: Cấu trúc của “*message passing*” trong GNN, trong đó ô vuông là phép tổng hợp đặc trưng của các lân cận như là phép lấy tổng, trung bình, ...[4]

Cụ thể hơn, báo cáo sẽ trình bày về hai ví dụ cụ thể của GNN là mạng nơ-ron tích chập (GCN) và mạng nơ-ron chú ý (GAT). Đầu tiên, đối với GCN thì ta chỉ lấy tổng các đặc trưng của lân cận nhân với một trọng số c_{ij} thể hiện mức

độ ảnh hưởng đến đỉnh của nó[8]. Tuy nhiên hệ số này lại không được tính thông qua bất kì biến nào mà chỉ là một tham số học thông thường. Dó đó, GAT [12] đã thay thế trọng số c_{ij} thành một hàm tính trọng số với đầu vào là hai đặc trưng của đỉnh hiện tại i và đỉnh lân cận j . Điều này sẽ giúp mô hình có được hệ số chính xác hơn về mức độ ảnh hưởng của các lân cận j đến i .

$$f(\mathbf{x}_i) = \phi \left(\mathbf{x}_i, \bigcup_{j \in \mathcal{N}_i} c_{ij} \psi(\mathbf{x}_j) \right)$$


“convolutional”

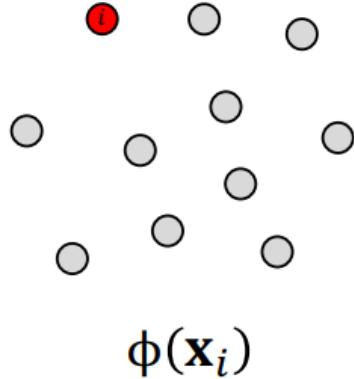
$$f(\mathbf{x}_i) = \phi \left(\mathbf{x}_i, \bigcup_{j \in \mathcal{N}_i} a(\mathbf{x}_i, \mathbf{x}_j) \psi(\mathbf{x}_j) \right)$$

“attentional”

Hình 4.8: Cấu trúc của hàm tổng hợp thông tin của GCN và GAT[4]

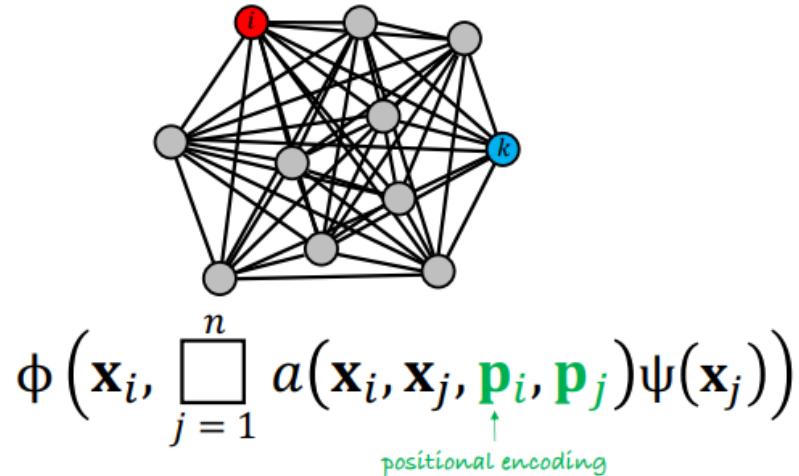
Mặc dù GNN thường được biết đến là mạng nơ-ron làm việc với dữ liệu dạng đồ thị nhưng thực tế, nó có một số dạng đặc biệt mà được gọi với tên gọi khác. Dưới đây là một số trường hợp đặc biệt của GNN:

- DeepSets[15]: Đây là một mạng nơ-ron làm việc trên tập các điểm ví dụ như point cloud. Do đó, ta có thể coi mạng này là một GNN làm việc với đồ thị không có cạnh.



Hình 4.9: Ảnh minh họa DeepSet[4]

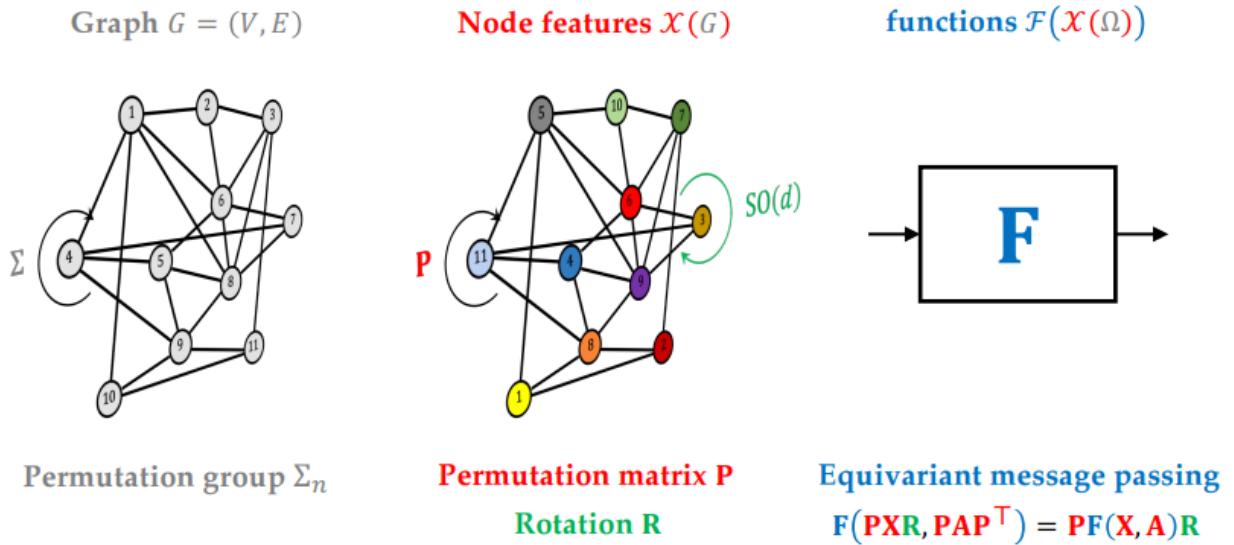
- Transformers[11]: Đây là một kiến trúc mạng nơ-ron rất nổi tiếng trong xử lý ngôn ngữ tự nhiên (NLP). Transformers tính hệ số attentions từ một từ đến tất cả các từ khác trong câu. Do đó, nếu ta coi các từ này là một đỉnh của đồ thị và trọng số attention chính là các cạnh thì đây chính là một đồ thị đầy đủ, cho nên, có thể coi Transformers là một mạng GNN làm việc trên đồ thị đầy đủ.



Hình 4.10: Ảnh minh họa Transformer[4]

Thông thường đồ thị mà chúng ta làm việc với GNN là chỉ có đặc trưng của các đỉnh mà chưa hề có tọa độ của các đỉnh, ví dụ như đồ thị biểu diễn của

một chất hóa học. Do đó, nếu ta làm việc với đồ thị được nhúng vào trong một không gian Euclid, ngoài việc các hàm phải *invariant* và *equivariant* đối với phép hoán vị chỉ số, ta sẽ cần phải thiết kế các hàm *invariant* và *equivariant* đối với các tác động của nhóm phép quay, nghịch đảo, dịch chuyển, phản chiếu[4].



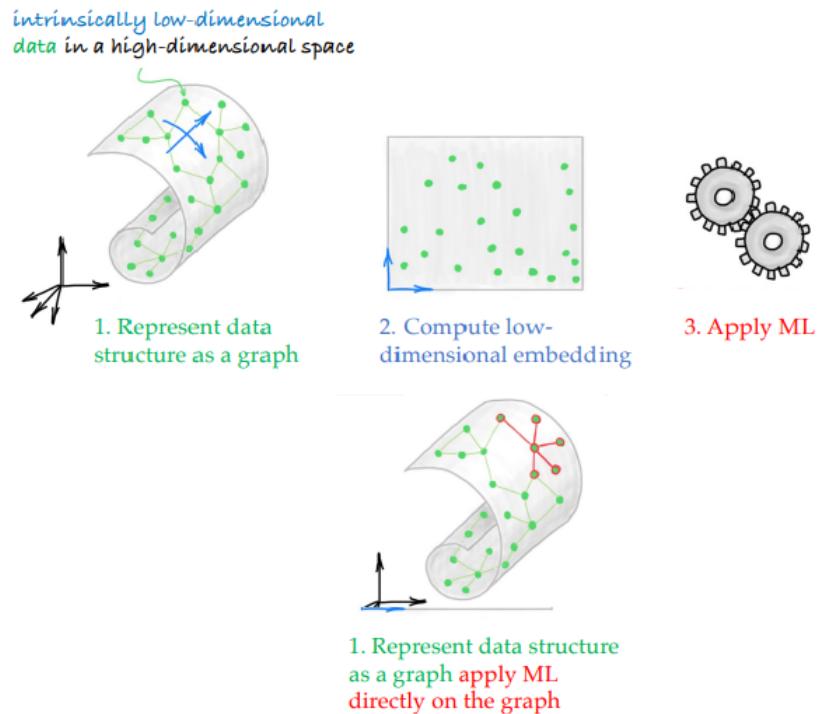
Hình 4.11: Ảnh minh họa dữ liệu đồ thị hình học và các phép toán đại số tác động nén nó[4]

4.1.4 Dữ liệu đa tạp

Đa tạp[3] bản chất là một cấu trúc hình học mà trong đó cục bộ là một không gian Euclid. Ví dụ đơn giản nhất chính là trái đất mà chúng ta đang sống, ta đều biết rằng trái đất có cấu trúc hình học là hình cầu tuy nhiên nếu nhìn cục bộ trên trái đất thì lại là một mặt phẳng (giống như những gì chúng ta nhìn thấy khi đang đứng trên trái đất).

Mặc dù đa tạp có cấu trúc phức tạp nhưng nó lại có tính chất rất quan trọng là lân cận. Do đó, để có thể sử dụng các phương pháp học máy để xử lý dạng dữ liệu này thì ta hoàn toàn có thể biểu diễn nó dưới dạng đồ thị, từ đó ta có

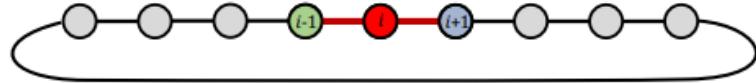
thể nhúng nó vào một không gian rồi tính toán trên không gian này[4]. Tuy nhiên đây là cách khá là lâu đài, do đó, ta có thể thực hiện một cách khác là sử dụng GNN lên đồ thị mà không cần phải nhúng đồ thị này vào một không gian nào đó[4].



Hình 4.12: Ảnh minh họa hai phương pháp trong *manifold learning*[4]

4.1.5 Dữ liệu dạng grid

Ta có thể coi *grid* là một đồ thị vòng (*ring graph*)[4]. Dữ liệu ở dạng này thì các đỉnh sẽ bị cố định cấu trúc lân cận và do đó, ta chỉ cần định nghĩa một hàm tổng hợp đặc trưng cho mỗi đỉnh khá là đơn giản. Hình 4.13 là một ví dụ minh họa về *grid*:

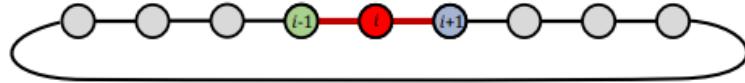


local aggregation function

$$f(\mathbf{x}_i) = \phi(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$$

Hình 4.13: Ảnh minh họa dữ liệu dạng *grid* và cấu trúc hàm tổng hợp đặc trưng[4]

Như ta có thể thấy trong hình 4.13 thì hàm tổng hợp đặc trưng chỉ đơn giản là một hàm nhận vào 3 tham số mà không cần đưa input vào một hàm nào khác như GNN. Để tổng quát hơn, ta có thể coi hàm này là tổ hợp tuyến tính của ba tham số đầu vào: $f(x_i) = ax_{i-1} + bx_i + cx_{i+1}$ [4]



linear local aggregation function

$$f(\mathbf{x}_i) = a\mathbf{x}_{i-1} + b\mathbf{x}_i + c\mathbf{x}_{i+1}$$

Hình 4.14: Ảnh minh họa hàm tổng hợp đặc trưng đối với dữ liệu *grid*[4]

4.1.6 Convolution và tính equivariant của nó với phép dịch chuyen

Để có cái nhìn trực quan và dễ hiểu, báo cáo sẽ lấy ví dụ đối với dữ liệu dạng *grid*. Đối với dạng dữ liệu này, ta có thể tổng quát hóa công thức tổ hợp tuyến tính ở phần trước thành một ma trận *circulant*[4]. Khi đó, ma trận này chính

là một phép tích chập.

$$f(\mathbf{X}) = \begin{bmatrix} b & c \\ a & b & c \\ & a & b & c \\ c & & a & b \end{bmatrix} \begin{bmatrix} a \\ \mathbf{X} \end{bmatrix}$$

circulant matrix = convolution

Hình 4.15: Ảnh minh họa về ma trận *circulant*[4]

Hình 4.15 minh họa về phép tính convolution đối với dữ liệu dạng *grid* với những phần để trống là có giá trị bằng 0. Ở đây, ma trận X sẽ là các vector hàng tương ứng với các vector đặc trưng của mỗi đỉnh trong dữ liệu.

Để chứng minh Convolution equivariant với phép dịch chuyển thì trước hết ta cần xây dựng một ma trận dịch chuyển gọi là *Shift*. Ma trận này sẽ có dạng như hình 4.16:

shift S

$$\begin{bmatrix} & 1 & & \\ & & 1 & \\ & & & 1 \\ 1 & & & \end{bmatrix}$$

Hình 4.16: Ảnh minh họa về ma trận $Shift[4]$

Ta thấy rằng nếu ta đem ma trận *circulant* (ký hiệu là C) nhân với ma trận *Shift* (ký hiệu là S) thì sẽ làm dịch chuyển các giá trị của ma trận *circulant* sang một cột. Tuy nhiên, phép nhân giữa hai ma trận này có một tính chất đặc biệt là giao hoán - nghĩa là $CS = SC$. Tính chất này đã thể hiện tính equivariant của ma trận *circulant* đối với phép dịch chuyển hãy nói rõ hơn thì phép *convolution* equivariant với phép dịch chuyển.

$$\left[\begin{array}{ccccc} b & c & & & a \\ a & b & c & & \\ & a & b & c & \\ & & a & b & c \\ c & & & a & b \end{array} \right] \left[\begin{array}{cccc} & 1 & & \\ & & 1 & \\ & & & 1 \\ 1 & & & \end{array} \right] = \left[\begin{array}{cccc} & 1 & & \\ & & 1 & \\ & & & 1 \\ 1 & & & \end{array} \right] \left[\begin{array}{ccccc} b & c & & & a \\ a & b & c & & \\ & a & b & c & \\ & & a & b & c \\ c & & & a & b \end{array} \right]$$

convolution \Rightarrow shift-equivariant

$$CS = SC$$

Trong đại số tuyến tính có một định lý được phát biểu rằng: "tích của hai ma trận mà có tính giao hoán thì khi ta chéo hóa hai ma trận thì chúng sẽ có chung

các vector riêng". Do đó thay vì ta tính vector riêng và giá trị riêng cho ma trận *circulant*, ta hoàn toàn có thể tính chúng thông qua chéo hóa ma trận *Shift* mà ma trận này chéo hóa rất nhanh vì chỉ có giá trị 0 hoặc 1. Điều này giúp ta giảm các phép tính toán đi rất nhiều lần cho việc chéo hóa ma trận *circulant*. Hơn nữa, với mọi ma trận *circulant* khác nhau ta cũng đều có thể dùng chung vector riêng và do đó, ta chỉ cần tính lại giá trị riêng là hoàn thành việc chéo hóa.

$$\begin{bmatrix} b & c & a \\ a & b & c \\ c & a & b \end{bmatrix} = \begin{bmatrix} | & & | \\ u_1 & \dots & u_n \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_n \end{bmatrix} \begin{bmatrix} --- u_1^* --- \\ \vdots \\ --- u_n^* --- \end{bmatrix}$$

4.1.7 Mô hình hóa convolution cho không gian tổng quát

4.1.7.1 Tích chập trên miền số thực

Dầu tiên, báo cáo sẽ mô hình hóa convolution với một ví dụ đơn giản nhất là trên trường số thực. Giả sử ta tính tích chập hai hàm x và ψ kí hiệu là $(x \star \psi)(u)$, khi đó bản chất ta tính tích chập giữa hai hàm này là ta shift hàm ψ trên trực số thực rồi nhân với hàm x [4], ta kí hiệu phép shift này là $T_u\psi$. Do đó, ta có công thức của phép tính tích chập trên miền số thực như sau:

$$(x \star \psi)(u) = \langle x, T_u\psi \rangle = \int_{-\infty}^{+\infty} x(v)\psi(u-v)dv$$

Để tổng quát hơn, báo cáo sẽ đưa phép shift này thành tác động của nhóm các

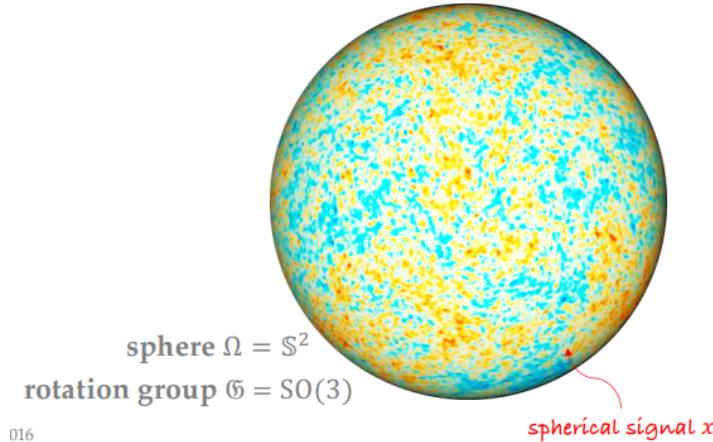
phép dịch chuyển gọi là G , do đó, miền lấy tích phân sẽ là miền giá trị đầu vào của hai hàm x và ψ gọi là Ω . Cuối cùng, ta có công thức sau:

$$(x \star \psi)(g) = \langle x, \rho(g)\psi \rangle = \int_{\Omega} x(v)\psi(g^{-1}v)dv$$

Tuy nhiên, đây mới chỉ là phép tích chập cho miền số thực, cho nên, tiếp theo báo cáo sẽ trình bày phép tích chập cho các miền không không gian tổng quát hơn.

4.1.7.2 Tích chập trên mặt cầu

Mạng nơ-ron tích chập (CNN) đã trở thành phương pháp được lựa chọn cho các bài toán học liên quan đến ảnh phẳng 2D. Tuy nhiên, một số vấn đề của sự quan tâm gần đây đã tạo ra nhu cầu về các mô hình có thể phân tích hình ảnh hình cầu[1]. Ví dụ bao gồm tầm nhìn đa hướng cho máy bay không người lái, robot và ô tô tự hành, bài toán hồi quy phân tử, và mô hình thời tiết và khí hậu toàn cầu. Một ứng dụng đơn giản của mạng tích chập vào phép chiếu phẳng của tín hiệu hình cầu chắc chắn sẽ thất bại, bởi vì các biến dạng thay đổi theo không gian được tạo ra bởi tín hiệu đó một phép chiếu sẽ làm cho việc chia sẻ trọng số tịnh tiến không hiệu quả[1]. Do đó, ta sẽ cần phải định nghĩa được phép tích chập mới trên tín hiệu hình cầu.



Hình 4.17: Ví dụ minh họa về dữ liệu dạng Sphere[4]

Ta mong muốn có thể định nghĩa tích chập trên miền mặt cầu S_2 nhưng khi ta tác động một phép quay lên hình cầu này thì các tín hiệu cầu sẽ bị thay đổi vị trí. Do đó, điều ra của phép tích chập này sẽ là một hàm theo phép quay R :

$$(x * \psi)(R) = \int_{S^2} x(u)\psi(R^{-1}u)du$$

Phép quay R ở đây bản chất là ta di chuyển vị trí u đến vị trí v trên mặt cầu S_2 và tác động phép quay này chính là tác động của nhóm $SO(3)$. Tuy nhiên, trong $SO(3)$ thì có rất nhiều phép quay để đi từ vị trí u đến vị trí v , do đó, ta cần phải tính thêm tích chập trên nhóm $SO(3)$. Khi đó, ta có công thức tính tích chập như sau:

$$((x * \psi) * \phi)(R) = \int_{SO(3)} (x * \psi)(Q)\phi(R^{-1}Q)dQ$$

Để thấy được sự hữu dụng của phép tích chập trên mặt cầu, báo cáo sẽ đưa ra một số ứng dụng cụ thể của chúng. Cụ thể hơn, báo cáo sẽ đưa ra một số thực nghiệm của một bài báo nổi tiếng là *Spherical CNN*[1].

Đầu tiên là thực nghiệm so sánh hiệu suất phân loại của *Spherical CNN* với mạng CNN phẳng trên bộ dữ liệu MNIST được chiếu lên mặt cầu. Kết quả cho

thấy *Spherical CNN* đạt độ chính xác 96% trên tập kiểm tra xoay khi được huấn luyện trên tập huấn luyện xoay, trong khi CNN phẳng chỉ đạt 23%[1]. Điều này chứng tỏ khả năng học biểu diễn bất biến xoay vượt trội của *Spherical CNN*.

Tiếp theo, tác giả áp dụng *Spherical CNN* cho bài toán phân loại hình dạng 3D, sử dụng bộ dữ liệu SHREC17 bao gồm các mô hình 3D được xoay ngẫu nhiên. Trên bộ dữ liệu này, *Spherical CNN* đạt điểm F1 là 0.699, chỉ số NDCG là 0.756 và độ chính xác trên tập dữ liệu không xoay là 0.701[1]. Kết quả này cho thấy *Spherical CNN* đạt hiệu suất gần như tốt nhất so với các phương pháp hiện đại khác, chứng tỏ tiềm năng của nó trong việc xử lý dữ liệu 3D.

Cuối cùng, tác giả nghiên cứu khả năng của *Spherical CNN* trong việc dự đoán năng lượng nguyên tử hóa của các phân tử dựa trên cấu trúc hình học của chúng. Sử dụng bộ dữ liệu QM7 và biểu diễn các phân tử dưới dạng tín hiệu cầu dựa trên năng lượng Coulomb, *Spherical CNN* đã đạt được RMSE là 8.47 kcal/mol[1]. Kết quả này vượt trội hơn hẳn so với các phương pháp dựa trên nhân (RMSE từ 10.82 đến 16.06 kcal/mol) và mạng MLP được huấn luyện trên ma trận Coulomb (RMSE là 12.59 kcal/mol)[1].

4.1.7.3 Tích chập cho đa tạp

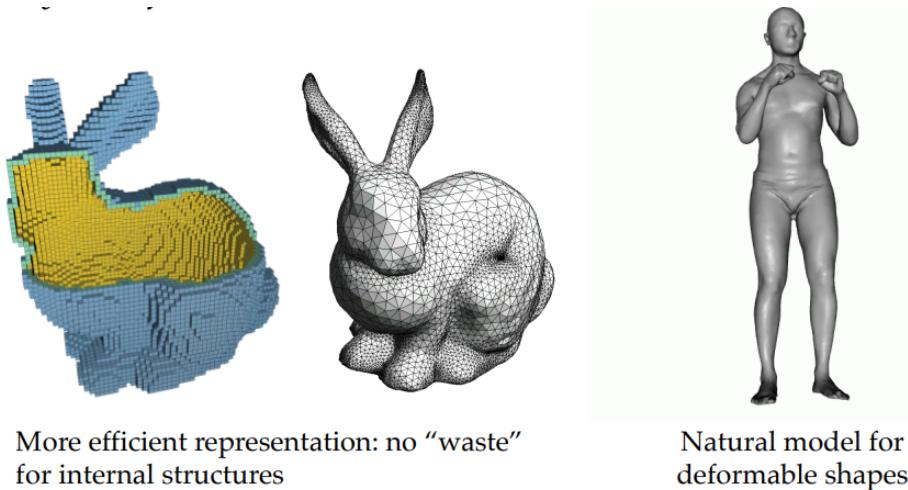
Manifold được sử dụng trong nhiều lĩnh vực, đặc biệt là trong học máy và xử lý dữ liệu, vì nó giúp chúng ta hiểu rõ hơn về cấu trúc bên trong của dữ liệu phức tạp. Manifold đặc biệt hữu ích đối với 3D shape (hình dạng 3D) vì nó giúp biểu diễn và xử lý các cấu trúc phức tạp trong không gian 3D một cách hiệu quả hơn.

Ví dụ, để mô hình một con thỏ theo dạng 3D, ta thường dùng cách khối lập phương chồng lên nhau để mô tả hình dạng và thể tích của chúng (hình ảnh

con thỏ bên trái của hình 4.18). Tuy nhiên, nếu ta không quan tâm bên trong hình dạng này thì việc biểu diễn bằng các khối lặp phương rất lãng phí không gian lưu trữ và tính toán[4]. Do đó, đa tạp giúp biểu diễn hình dạng 3D một cách hiệu quả hơn bằng cách tập trung vào các bề mặt của đối tượng, bỏ qua các cấu trúc bên trong không cần thiết.

Do đó, ta có thể sử dụng *manifold* để biểu diễn dữ liệu 3D, nơi chỉ các điểm dữ liệu nằm trên bề mặt của đối tượng 3D được lưu trữ và sử dụng để tạo thành mạng lưới tam giác (mesh). Bởi vậy, *manifold* không "lãng phí" tài nguyên để lưu trữ các cấu trúc bên trong, chỉ tập trung vào bề mặt, từ đó tạo ra một biểu diễn nhẹ và hiệu quả hơn[4] (hình ảnh con thỏ ở giữa trong hình 4.18). Đây cũng là cách được áp dụng trong *computer graphic* liên quan đến vật thể 3D.

Hơn thế nữa, *manifold* được sử dụng như một mô hình tự nhiên cho các hình dạng có thể biến dạng (deformable shapes)[4]. Ví dụ như hình dạng của con người, với các bộ phận có thể co giãn, di chuyển hoặc thay đổi hình dạng, là một ví dụ điển hình về việc sử dụng manifold (hình ảnh bên phải của hình 4.18). Do đó, *manifold* phù hợp với các hình dạng có thể biến dạng, vì nó có thể dễ dàng mô phỏng và điều chỉnh các biến dạng mà không làm mất đi tính toàn vẹn của hình dạng ban đầu[4].



Hình 4.18: Ảnh minh họa dữ liệu đa tạp và meshes[4]

Tuy nhiên, một ứng dụng quan trọng khác của *manifold* là mô hình hóa protein. Trong quá trình mô hình hóa protein, mục tiêu là biểu diễn các phân tử protein sao cho dễ dàng phân tích các tương tác và biến đổi cấu hình (conformation changes) của chúng. Protein có cấu trúc rất phức tạp với nhiều lớp và các thành phần khác nhau, nhưng không phải tất cả các chi tiết cấu trúc đều quan trọng cho việc nghiên cứu các tương tác chức năng của protein. Dưới đây là hai lý do thể hiện tính hữu ích của *manifold* đối với bài toán này:

- Manifold giúp "trừu tượng hóa" (abstract out) các cấu trúc bên trong của protein mà không cần thiết cho việc tương tác. Điều này có nghĩa là chỉ tập trung vào các yếu tố quan trọng như bề mặt của protein, nơi xảy ra các tương tác với các phân tử khác[4]. Bằng cách này, chúng ta có thể giảm độ phức tạp của mô hình mà không làm mất đi thông tin cần thiết cho việc phân tích.
- Protein không phải là những cấu trúc cố định, chúng có thể thay đổi hình dạng hoặc cấu hình để thích ứng với các chức năng sinh học khác nhau. Manifold cung cấp một cách linh hoạt để mô hình hóa những thay đổi này, giữ nguyên tính toàn vẹn của các vùng tương tác quan trọng trong khi cho phép protein thực hiện các biến đổi cấu hình cần thiết[4].

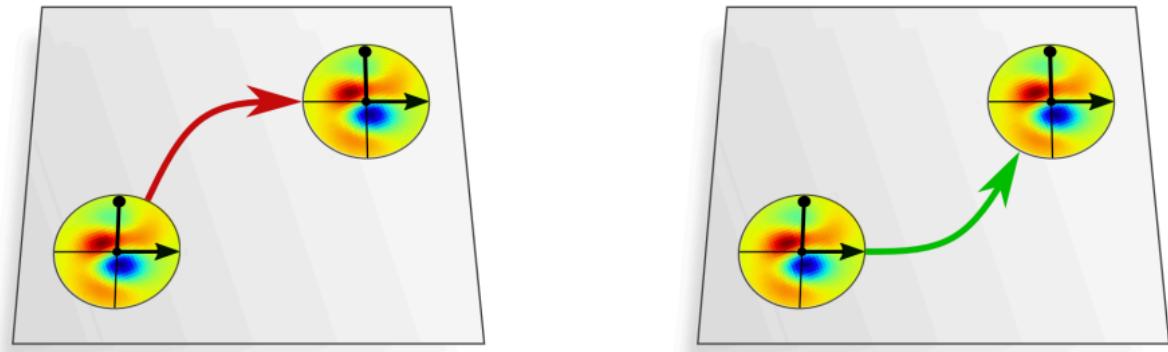
In protein modeling,
abstract out internal
structure that is irrelevant
for interactions + allow
some conformation
changes



Hình 4.19: Ảnh minh họa protein[4]

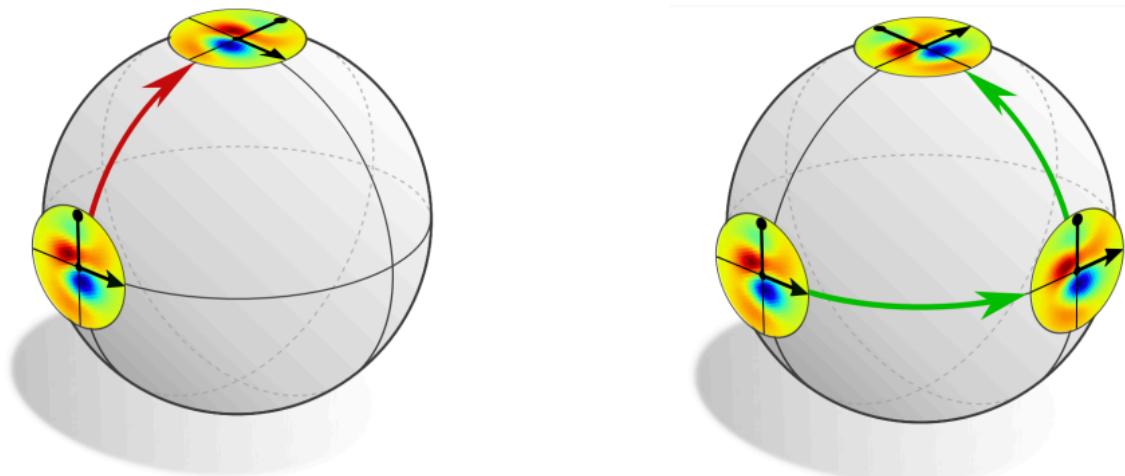
Thông thường, nếu ta sử dụng tích chập trên không gian Euclid thì việc ta di chuyển *filter* từ vị trí A đến vị trí B thì đều sẽ cho kết quả như nhau vì chúng đều đến vị trí B và tính tích chập tại đó (hình 4.20). Tuy nhiên, đối với *manifold* thì lại lại khác vì nếu ta đi từ A đến B theo hai con đường khác nhau thì *filter* sẽ bị thay đổi hướng ban đầu và do đó kết quả tính tích chập tại vị trí B sẽ khác nhau. Để thể hiện rõ điều này, báo cáo sẽ đưa ra hai ví dụ sau:

- Tích chập trên hình cầu: Hình 4.21 cho thấy hướng của *filter* khác nhau của hai đường đi khác nhau mặc dù chúng có cùng điểm xuất phát và điểm đến.
- Tích chập trên mặt *mobius*: Hình 4.22 cho thấy *filter* của đường đi thứ 2 đã bị lật ngược so với đường đi đầu tiên mặc dù chúng có cùng điểm xuất phát và điểm đến.



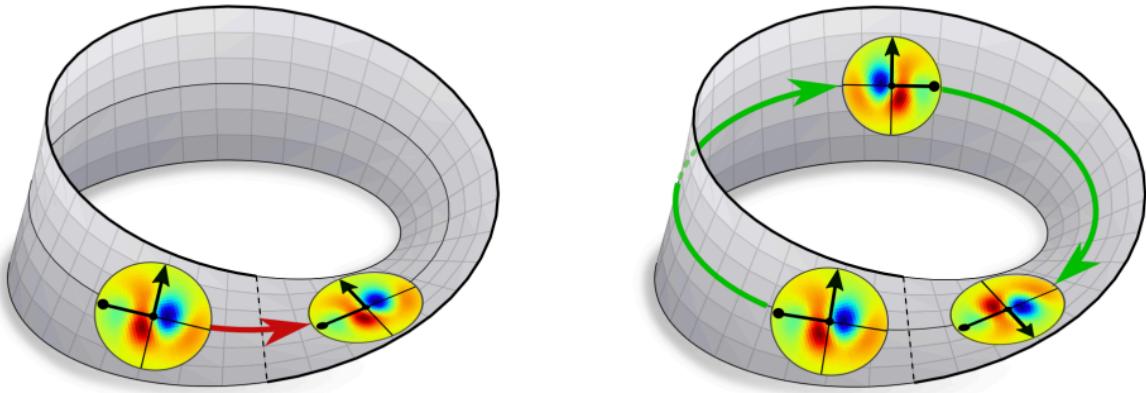
Euclidean space: Transport the filter around the domain

Hình 4.20: Ảnh minh họa tích đường đi của *filter* trên không gian Euclid[4]



Manifold: Result of transport is *path dependent*

Hình 4.21: Ảnh minh họa tích đường đi của *filter* trên *sphere*[4]

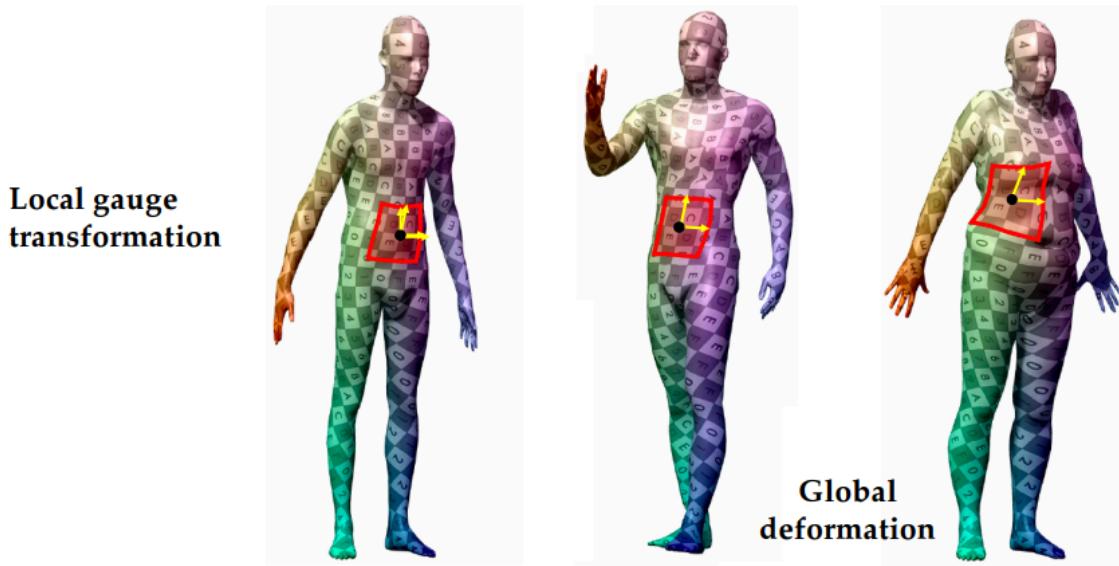


Manifold: Result of transport is *path dependent*

Hình 4.22: Ảnh minh họa tích đường đi của *filter* trên mặt *mobius*[4]

Đối với dữ liệu *manifold*, có hai dạng *invariant* dưới các tác động nhóm như sau:

- Biến đổi cục bộ gauge: đây là tác động mà chỉ làm thay đổi tính chất cục bộ của đa tạp mà không ảnh hưởng đến hình dạng tổng thể[4]. Ví dụ, ta có thể co bóp bụng và điều này làm thay đổi cấu trúc vùng bụng của chúng ta tuy nhiên về mặt tổng thể thì điều này không làm ảnh hưởng đến chúng ta là ai (hình bên trái của hình 4.23).
- Biến đổi toàn cục: đây là tác động làm thay đổi toàn bộ hình dạng của đối tượng, như việc cơ thể người có thể bị kéo giãn, co lại hoặc thay đổi tư thế, tuy nhiên điều này không làm thay đổi đối tượng đấy là gì [4]. Ví dụ như người A béo lên thì đây chính là tác động làm toàn bộ cơ thể bị thay đổi nhưng về bản chất thì người A vẫn là người A mà không bị biến đổi thành người khác (hai hình bên phải hình 4.23).



Hình 4.23: Ảnh minh họa hai dạng *invariant* đối với dữ liệu *manifold*[4]

Bây giờ, vấn đề được đặt ra cho việc tính tích chập trên đa tạp là chúng ta cần định nghĩa *filter*. Rất may là đối với đa tạp thì locally ta có thể biểu diễn chúng lên không gian tiếp xúc tại một điểm trên đa tạp (*tangent space*) và nó đẳng cấu với không gian R^2 . Do đó, ta có thể định nghĩa được filter trên không gian này. Khi đó, ta định nghĩa công thức tính tích chập trên *manifold* như sau:

$$(x * \psi)(u) = \int_{T_u \Omega} \psi(v) x(\exp_u v) dv$$

trong đó:

- $T_u \Omega$ là không gian tiếp xúc (*tangent space*) tại điểm u trên đa tạp
- \exp_u là một phép ánh xạ nội tại nhằm ánh xạ một điểm trong *tangent space* sang một đa tạp Ω

tuy nhiên, có một điểm cần lưu ý ở đây là filter của chúng ta sẽ phải invariant với các phép biến đổi đẳng metric (*isometries*).

Ở công thức trên báo cáo đã xây dựng được công thức tính tích chập cho *manifold*, tuy nhiên vector \mathbf{v} trong công thức là một vector trừu tượng, nghĩa là vector này chưa có tọa độ, cho nên phép ánh xạ \exp_u không thể ánh xạ \mathbf{v} sang *manifold*. Do đó, ta cần phải cho \mathbf{v} tọa độ trong \mathbb{R}^2 và có một phép biến đổi để ánh xạ \mathbf{v} vào không gian tiếp xúc $T_u\Omega$ - ta gọi phép ánh xạ này là ω_u . Khi đó, ta có công thức tính tích chập sau:

$$(x * \psi)(u) = \int_{\mathbb{R}^2} \psi(\mathbf{v}) x(\exp_u \omega_u \mathbf{v}) d\mathbf{v}$$

Tuy nhiên, đây chỉ là công thức cho việc tính tích chập khi chưa có tác động của một nhóm. Do đó, ta sẽ cần xây dựng công thức cho phép biến đổi này, cụ thể hơn là phép biến đổi cục bộ (*gauge transform*). Ta ký hiệu phép biến đổi này như sau:

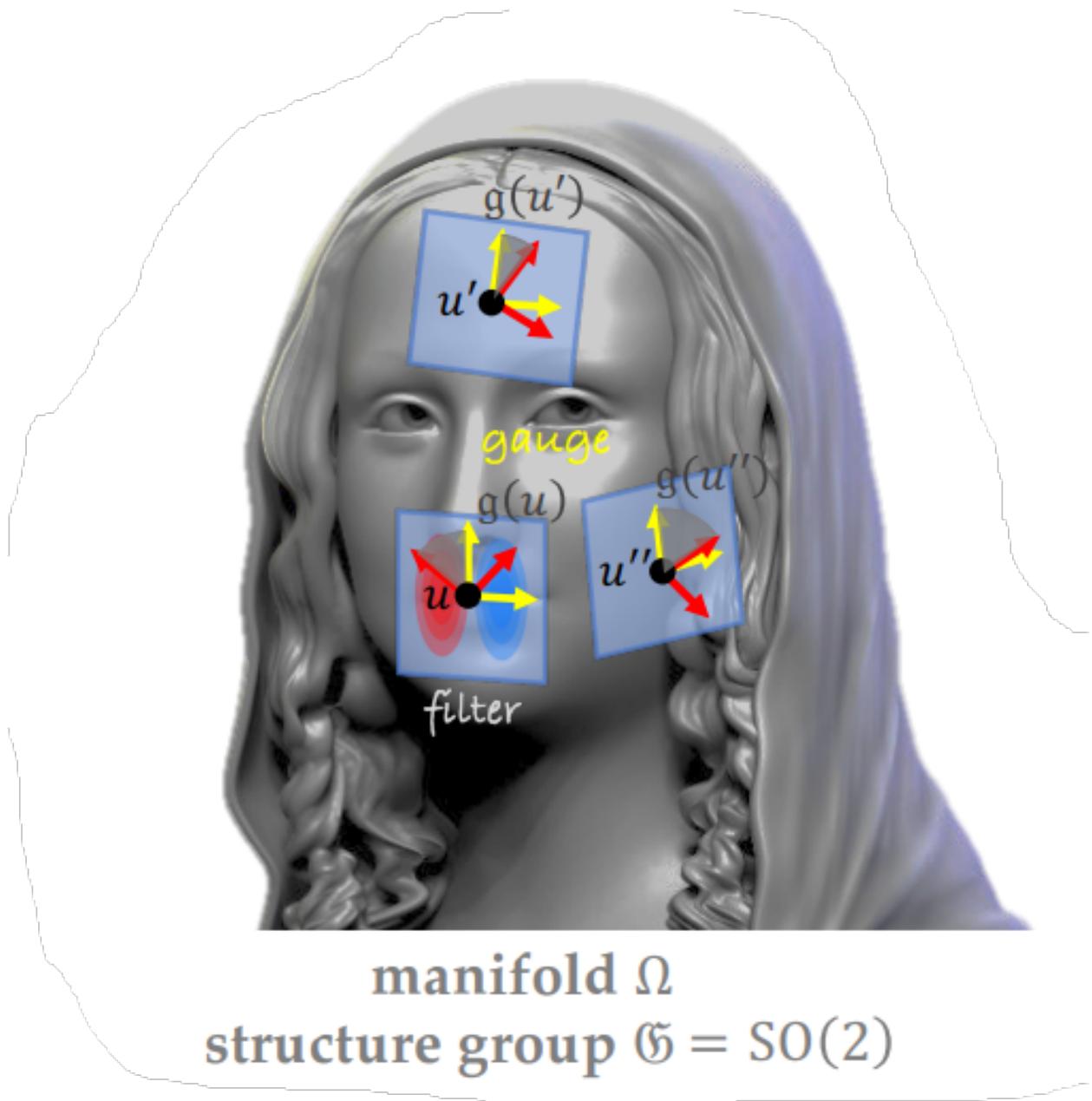
$$g : \Omega \rightarrow SO(2)$$

Cuối cùng, ta có công thức tính tích chập dưới tác động của một nhóm như sau:

$$(x \star \psi)(u) = \int_{\mathbb{R}^2} \psi(v) \rho(g) x(\exp_u \omega_u v) d\mathbf{v}$$

trong đó, *filter* sẽ phải *equivariant* đối với *gauge transform*, nghĩa là:

$$\psi(g^{-1}\mathbf{v}) = \rho(g^{-1})\psi(\mathbf{v})\rho(g)$$



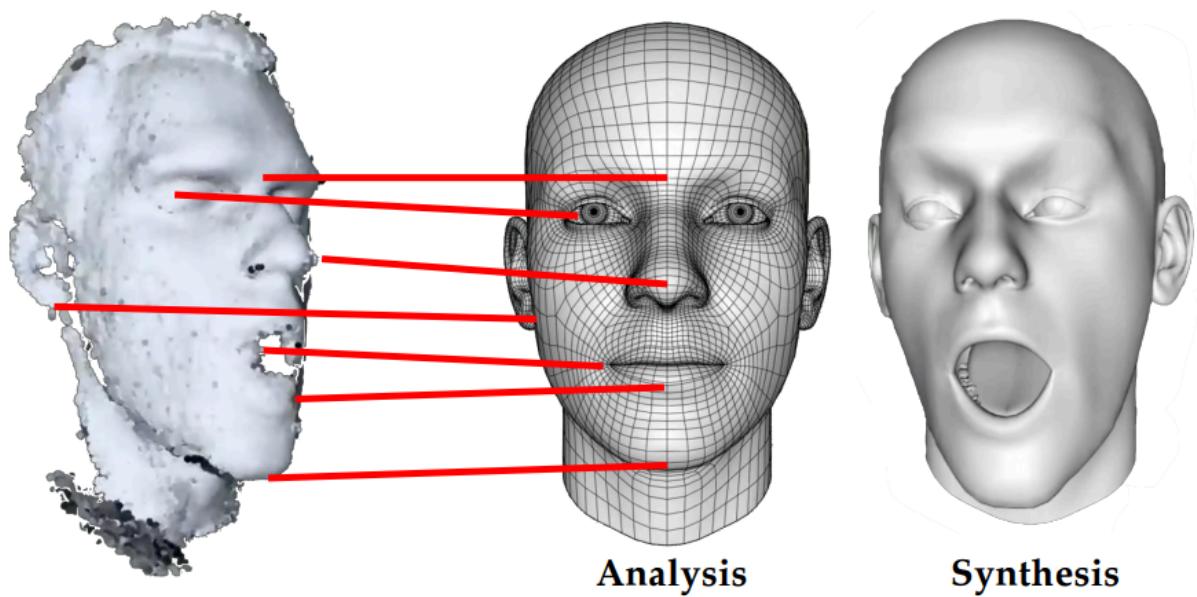
Hình 4.24: Ảnh minh họa filter trên đa tạp có tác động của một nhóm[4]

Tiếp theo, báo cáo sẽ đưa ra một số ứng dụng thực tế của tích chập trên đa tạp. Một ứng dụng gần gũi với chúng ta nhất đây là làm đồ họa 3D cho game. Ở đây, việc chúng ta cần làm là chuyển đổi các chuyển động mẫu của khuôn mặt người vào trong đồ họa máy tính.



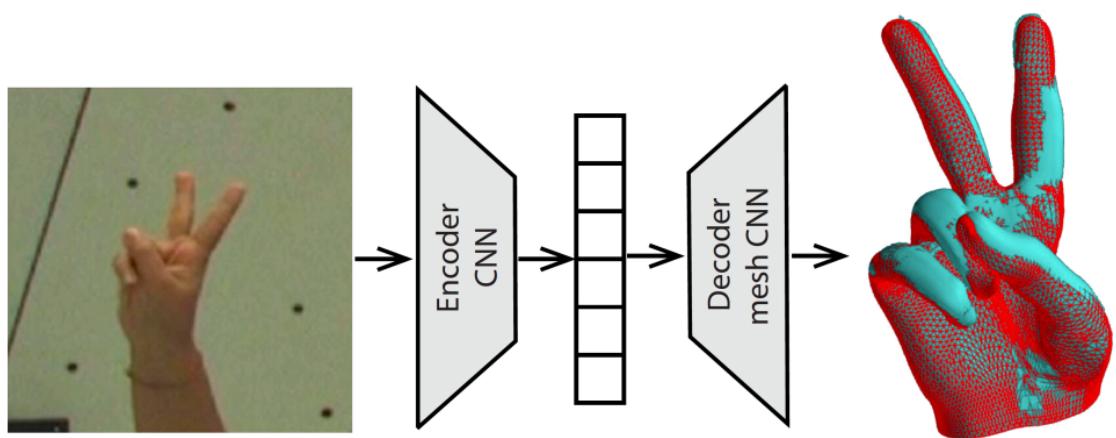
Hình 4.25: Ứng dụng của tích chập cho manifold trong computer graphic[4]

Để có thể chuyển được các chuyển động của khuôn mặt sang cho nhân vật trong đồ họa máy tính, đầu tiên là ta sẽ cần phải scan khuôn mặt người thật liên tục để có được input 3D. Khi đó, ta cần ánh xạ đầu vào này vào một khuôn mặt dạng mesh tiêu chuẩn rồi bắt đầu thực hiện các phép biến đổi trên đó để hình thành được chuyển động của khuôn mặt. Tuy nhiên, đầu vào 3D sau khi scan có khá là nhiều nhiễu và do đó, ta cần phải sử dụng mesh-convolution để có thể ánh xạ chính xác nhất có thể từ đầu vào sang khuôn mặt dạng mesh tiêu chuẩn[4].



Hình 4.26: Tổng quan quá trình đưa chuyển động của khuôn mặt vào trong đồ họa máy tính[4]

Một ứng dụng khác của tích chập trên đa tạp đó là tái cấu trúc lại hình dạng 3D từ một hình ảnh 2D. Nghĩa là đầu vào sẽ là một hình ảnh và đầu ra sẽ là hình dạng 3D của vật thể trong hình ảnh đấy[4].



Hình 4.27: Ứng dụng tái cấu trúc lại hình dạng 3D của một vật thể từ một hình ảnh[4]

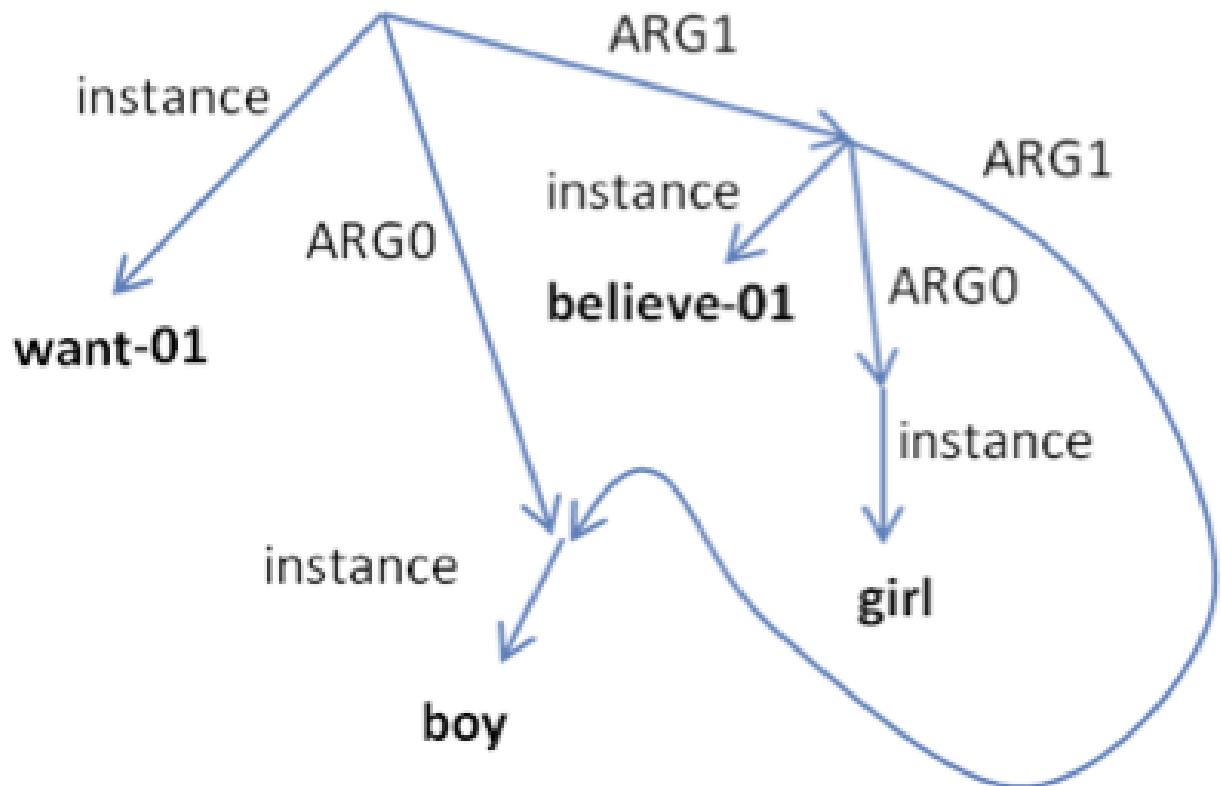
4.2 Geometric Deep learning cho AMR trong NLP

Lưu ý: đây chỉ là lý thuyết báo cáo tự đề xuất ra cho nên vẫn chưa được kiểm chứng hoàn toàn.

Abstract Meaning Representation (AMR) là một dạng biểu diễn ngữ nghĩa trừu tượng, dùng để mô tả ý nghĩa của câu theo cách ngôn ngữ độc lập[2]. AMR tập trung vào việc biểu diễn cấu trúc ngữ nghĩa của câu dưới dạng đồ thị, trong đó các đỉnh là các khái niệm hoặc thực thể, và các cạnh biểu diễn mối quan hệ giữa các khái niệm này.

AMR nắm bắt được “ai đang làm gì với ai” trong một câu. Mỗi câu được biểu diễn dưới dạng một đồ thị có gốc, có hướng, và không chu trình với các nhãn trên các cạnh (quan hệ) và các lá (khái niệm)[2].

Giống như một cây phân tích cú pháp, AMR cung cấp một cấu trúc duy nhất có thể duyệt qua mà tính đến tất cả các từ. Nó không phải là một tập hợp các lớp chú thích rời rạc. Không giống như cây phân tích cú pháp, AMR là trừu tượng. Nó có thể biểu diễn nhiều câu ngôn ngữ tự nhiên khác nhau. AMR không chú thích các từ riêng lẻ trong một câu, như một phân tích phụ thuộc[2].



Hình 4.28: Ví dụ minh họa về AMR[2]

Câu AMR trong hình 4.28 có nghĩa là: Có một sự kiện mong muốn, trong đó ARG0 (người mong muốn) là một cậu bé, và ARG1 (điều được mong muốn) là một sự kiện tin tưởng. Sự kiện tin tưởng này có ARG0 (người tin tưởng), là một cô gái, và nó có ARG1 (điều được tin tưởng), chính là cậu bé vừa được nhắc đến. Ở đây, cậu bé đóng hai vai trò: (1) là ARG0 của "want-01", và (2) là ARG1 của "believe-01". AMR thể hiện điều này bằng hai cạnh có hướng trở đến cùng một nút. (Theo OntoNotes, các giác quan của vị ngữ được đánh dấu bằng hậu tố như -01 và -02, trong khi ARG0, ARG1, v.v., chỉ các vai trò cốt lõi, cụ thể của vị ngữ.)

```
(w / want-01
  :ARG0 (b / boy)
  :ARG1 (b2 / believe-01
    :ARG0 (g / girl)
    :ARG1 b))
```

Hình 4.29: Biểu diễn AMR dạng text của ví dụ trong hình 4.28

Kể từ khi ra đời, AMR đã được áp dụng rộng rãi trong nhiều lĩnh vực xử lý ngôn ngữ tự nhiên. Các ứng dụng phổ biến bao gồm dịch máy, tóm tắt văn bản, trả lời câu hỏi, và phân tích cảm xúc. Sự đa dạng trong ứng dụng này chứng tỏ tính linh hoạt và hữu ích của AMR trong việc nắm bắt và biểu diễn ý nghĩa ngôn ngữ.

Song song với sự phát triển về ứng dụng, nhiều công cụ đã được tạo ra để hỗ trợ việc tự động hóa quá trình tạo và phân tích AMR. Các công cụ này bao gồm các bộ phân tích cú pháp tiên tiến và các mô hình học máy phức tạp, giúp tăng cường khả năng xử lý AMR ở quy mô lớn.

Mặc dù ban đầu AMR được phát triển chủ yếu cho tiếng Anh, nhưng theo thời gian, nó đã được mở rộng để hỗ trợ nhiều ngôn ngữ khác. Điều này đã mở ra cơ hội cho việc áp dụng AMR trong các nhiệm vụ xử lý ngôn ngữ đa ngôn ngữ và liên ngôn ngữ.

Quá trình phát triển của AMR vẫn đang tiếp diễn. Các nhà nghiên cứu liên tục cải tiến và mở rộng khả năng của AMR để xử lý các hiện tượng ngôn ngữ ngày

càng phức tạp hơn, đồng thời tăng độ chính xác trong biểu diễn ngữ nghĩa. Những nỗ lực này nhằm đảm bảo rằng AMR vẫn là một công cụ mạnh mẽ và linh hoạt trong lĩnh vực xử lý ngôn ngữ tự nhiên, có khả năng đáp ứng các thách thức mới nổi trong việc hiểu và biểu diễn ngôn ngữ tự nhiên.

Trong lĩnh vực dịch máy, AMR đóng vai trò quan trọng như một biểu diễn trung gian. Bằng cách chuyển đổi văn bản nguồn thành AMR, sau đó từ AMR sang ngôn ngữ đích, các hệ thống dịch có thể nắm bắt được ý nghĩa sâu sắc của văn bản, giúp tạo ra bản dịch chính xác và tự nhiên hơn. Phương pháp này đặc biệt hiệu quả khi xử lý các cặp ngôn ngữ có cấu trúc khác biệt lớn.

AMR cũng được sử dụng rộng rãi trong tác vụ tóm tắt văn bản. Bằng cách biểu diễn nội dung chính của văn bản dưới dạng đồ thị AMR, các hệ thống có thể dễ dàng xác định và trích xuất thông tin quan trọng, từ đó tạo ra các bản tóm tắt ngắn gọn và đầy đủ nội dung.

Trong lĩnh vực trả lời câu hỏi, AMR giúp phân tích cả câu hỏi và các đoạn văn bản tiềm năng chứa câu trả lời. Việc so sánh cấu trúc AMR của câu hỏi và văn bản cho phép hệ thống xác định chính xác thông tin cần thiết để đưa ra câu trả lời phù hợp.

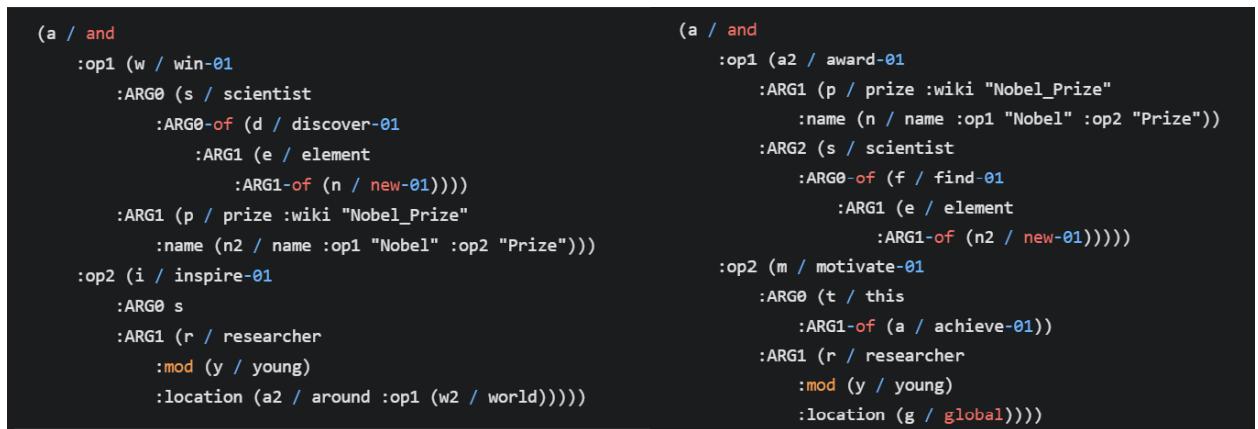
AMR cũng được ứng dụng trong phân tích cảm xúc và khai thác ý kiến. Bằng cách biểu diễn cấu trúc ngữ nghĩa của câu, AMR giúp hệ thống nắm bắt được các sắc thái tinh tế trong cảm xúc và quan điểm được thể hiện, vượt ra ngoài phân tích đơn thuần dựa trên từ khóa.

Trong lĩnh vực sinh văn bản tự động, AMR được sử dụng như một khuôn mẫu ngữ nghĩa. Các hệ thống có thể tạo ra văn bản mạch lạc và có ý nghĩa bằng cách sinh câu từ biểu diễn AMR, đảm bảo rằng nội dung được tạo ra phù hợp với ý định ngữ nghĩa mong muốn.

Ngoài ra, AMR còn được áp dụng trong nhiều lĩnh vực khác như phân tích diễn ngôn, trích xuất quan hệ, và hệ thống đối thoại. Trong mỗi trường hợp, khả năng nắm bắt ý nghĩa sâu sắc của AMR giúp cải thiện hiệu suất và độ chính xác của các hệ thống xử lý ngôn ngữ tự nhiên.

Tiếp theo, báo cáo sẽ đưa ra hai ví dụ về AMR của hai câu đồng nghĩa sau:

- "The scientist who discovered the new element won the Nobel Prize and inspired young researchers around the world."
- "The Nobel Prize was awarded to the scientist who found the new element, and this achievement motivated young researchers globally."



Hình 4.30: Biểu diễn AMR của hai câu đồng nghĩa

Từ hình 4.30 có thể thấy rằng cả hai AMR đều thể hiện được hai sự kiện chính: việc một nhà khoa học được trao giải Nobel vì phát hiện ra nguyên tố mới, và việc thành tựu này truyền cảm hứng/thúc đẩy các nhà khoa học trẻ. Bên cạnh đó, các thực thể chính (nhà khoa học, giải Nobel, nguyên tố mới) đều được xác định rõ ràng trong cả hai AMR. Mọi quan hệ ngữ nghĩa giữa các thực thể cũng được thể hiện rõ ràng bằng các nhãn vai trò ngữ nghĩa (ARG0, ARG1, ARG0-of, ARG1-of, mod, location).

Tuy nhiên, hai AMR có sự khác biệt về cấu trúc. AMR bên trái sử dụng cấu trúc phức tạp hơn với nhiều mệnh đề con lồng nhau, trong khi AMR bên phải có cấu trúc đơn giản hơn với hai mệnh đề chính được nối với nhau bởi "and".

Sự lựa chọn động từ trung tâm để diễn tả sự kiện được trao giải Nobel cũng khác nhau. AMR bên trái sử dụng động từ "win" (chiến thắng), trong khi AMR bên phải sử dụng "award" (trao thưởng). Việc lựa chọn động từ này ảnh hưởng đến cấu trúc và cách thức thể hiện vai trò ngữ nghĩa trong mỗi AMR.

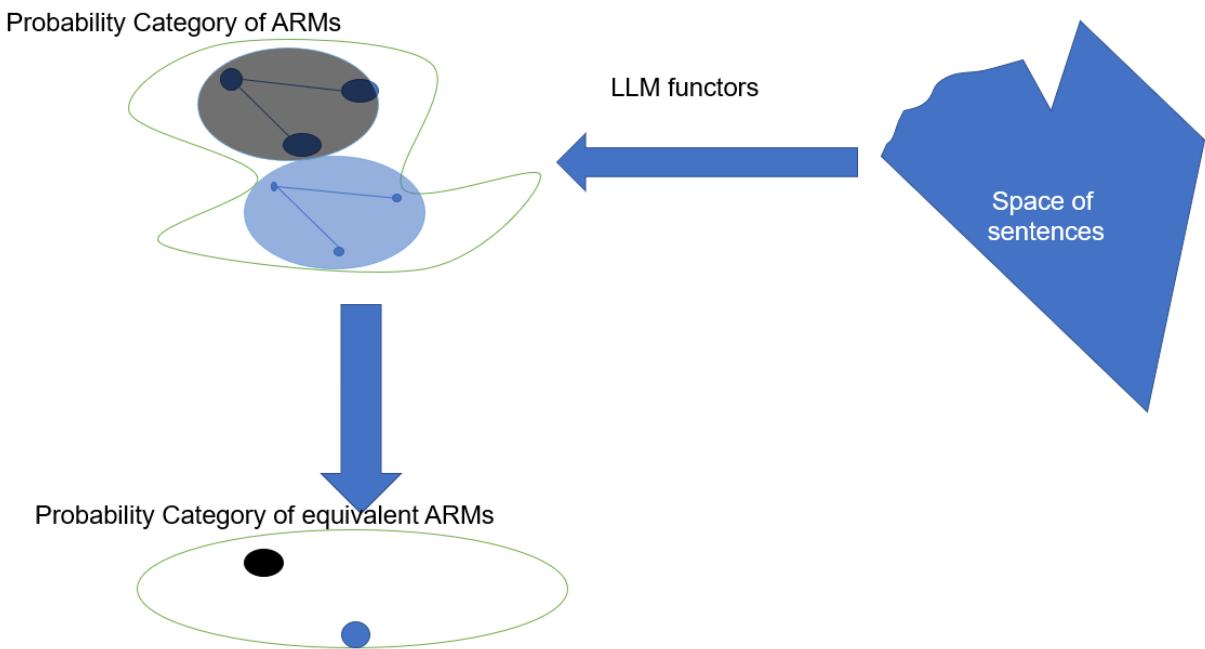
Phạm vi ảnh hưởng của sự kiện "truyền cảm hứng/thúc đẩy" cũng được thể hiện khác nhau. Trong khi AMR bên trái thể hiện sự kiện "truyền cảm hứng" có phạm vi ảnh hưởng là "các nhà khoa học trẻ trên toàn thế giới", thì AMR bên phải lại thể hiện sự kiện "thúc đẩy" có phạm vi ảnh hưởng là "tất cả" (không chỉ giới hạn ở các nhà khoa học trẻ).

Cuối cùng, vị trí của thông tin về phạm vi ảnh hưởng cũng khác nhau. AMR bên trái đặt thông tin về phạm vi ảnh hưởng ("around the world") ở cuối câu, trong khi AMR bên phải đặt thông tin này ("globally") gần với động từ "motivate".

Như vậy, có thể thấy rằng với hai câu khác nhau nhưng cùng ý nghĩa thì khi ta chuyển sang AMR thì sẽ cho ta hai đồ thị AMR khác nhau. Hệ quả, nên tồn tại một quan hệ xác suất giữa hai đồ thị AMR trong một văn cảnh nào đấy.

Do đó, không gian các đồ thị AMR có thể đưa ra ý tưởng về một cấu trúc mới. Hãy xem xét khái niệm tương đương (equivalence) như một tập hợp các mũi tên: có thể đảo ngược và có thể kết hợp (composable). Điều này dẫn đến việc hình thành một cấu trúc groupoid (groupoid structure). Tuy nhiên, ngoài cấu trúc groupoid, còn có nhiều cấu trúc khác có thể áp dụng. Ví dụ, ta có thể định nghĩa một hàm trên đồ thị AMR để mô tả tần suất sử dụng các phần tử trong

đồ thị.

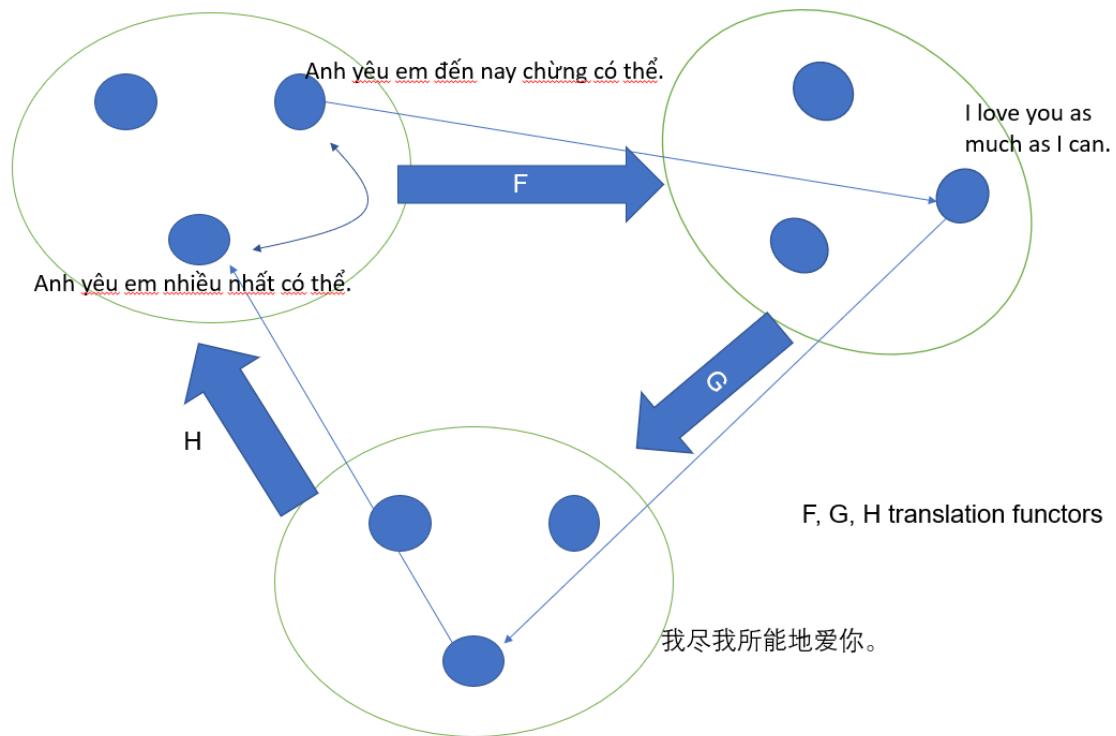


Hình 4.31: Ý tưởng cấu trúc mới về AMR. (LLM functors nghĩa là sử dụng LLM để chuyển một câu sang AMR)

Ngoài ra, việc so sánh các đồ thị AMR từ các câu đồng nghĩa cũng mở ra hướng nghiên cứu về các phép biến đổi giữa các đồ thị này. Chúng ta có thể xem xét cách các đồ thị AMR biến đổi khi có sự thay đổi trong cách diễn đạt ngôn ngữ tự nhiên, từ đó xây dựng các mô hình dự đoán hoặc phân loại dựa trên các biến đổi này. Điều này không chỉ giúp hiểu rõ hơn về cấu trúc ngữ nghĩa mà còn có thể ứng dụng trong việc cải thiện các mô hình ngôn ngữ, như mô hình dịch máy hoặc tóm tắt văn bản, bằng cách khai thác sự tương quan giữa các biểu diễn AMR khác nhau trong ngữ cảnh cụ thể.

Sự kết hợp (compose) của các hàm tử (functors) tạo ra các dạng tương đương AMR mới chưa được ghi nhận trong tài liệu hiện có, và những tương đương này có thể được tính toán một cách tự động thông qua mô hình ngôn ngữ lớn (Large Language Model).

Tiếp theo, báo cáo sẽ lấy một ví dụ dịch một câu sang các ngôn ngữ khác nhau rồi dịch trở lại ngôn ngữ ban đầu (hình 4.32).



Hình 4.32: Ví dụ minh họa bằng cách dịch một câu sang ngôn ngữ khác. Những điểm trong hình tròn là các câu khác nhau nhưng cùng nghĩa. F, G, H là các phép dịch một ngôn ngữ này sang ngôn ngữ khác.

Từ hình 4.32 cho thấy trong quá trình dịch và biểu diễn nghĩa của câu qua nhiều ngôn ngữ. Khi một câu được dịch qua các ngôn ngữ khác nhau (được biểu thị bằng các hàm ánh xạ F, G, H) rồi quay trở lại ngôn ngữ ban đầu, kết quả thu được thường là một câu mới nhưng vẫn giữ nguyên ý nghĩa cốt lõi của câu gốc.

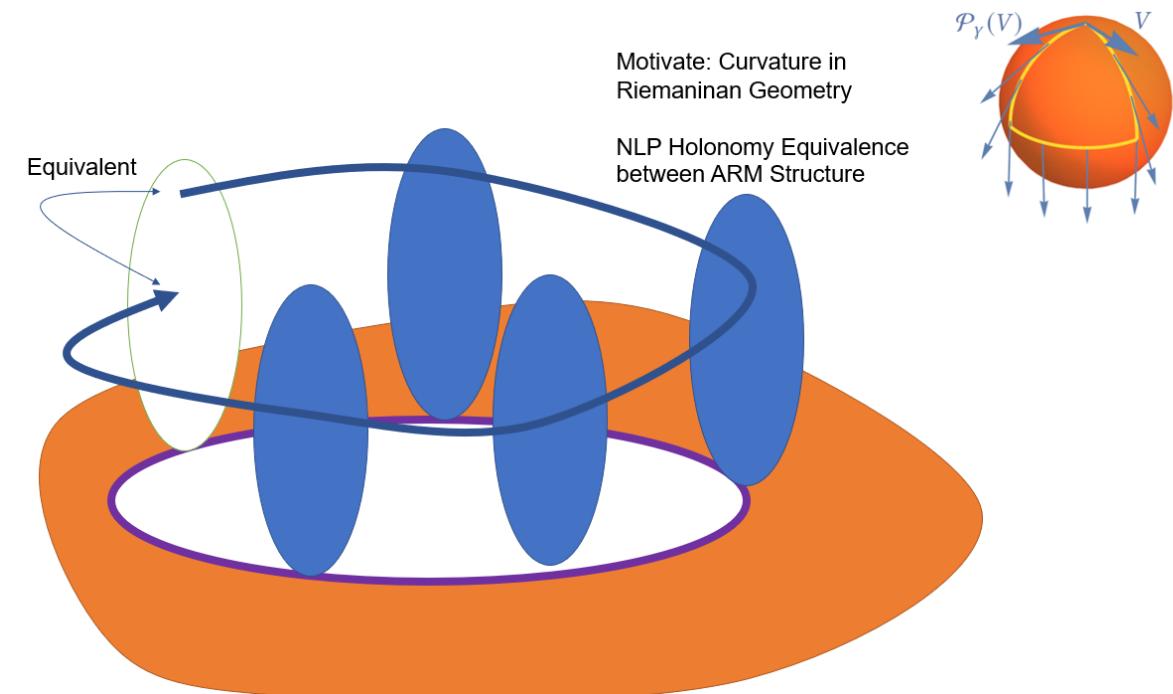
Các hàm ánh xạ F, G, H được coi là các phép biến đổi 1-1 giữa các không gian ngôn ngữ. Tuy nhiên, khi xét đến không gian biểu diễn ngữ nghĩa trừu tượng như AMR (Abstract Meaning Representation), phép nâng (lifting) không còn là phép đồng nhất (identity) mà chỉ còn là phép tương đương (equivalence).

Điều này có nghĩa là các đồ thị AMR của các câu đồng nghĩa, dù có thể khác

nhau về cấu trúc cụ thể, nhưng vẫn nên có một mức độ tương đồng nhất định. Ta có thể quantify mức độ tương đồng này bằng một xác suất hay độ đo tương đương nào đó.

Hiện tượng này phản ánh bản chất phức tạp của ngôn ngữ tự nhiên, nơi cùng một ý nghĩa có thể được diễn đạt bằng nhiều cách khác nhau, không chỉ trong cùng một ngôn ngữ mà còn qua các ngôn ngữ khác nhau. Nó cũng cho thấy sự cần thiết của các phương pháp biểu diễn ngữ nghĩa linh hoạt như AMR, có khả năng nắm bắt được bản chất ngữ nghĩa của câu mà không bị ràng buộc quá chặt vào cấu trúc cú pháp cụ thể.

Cuối cùng, nhận xét này gợi ý rằng trong các ứng dụng xử lý ngôn ngữ tự nhiên, đặc biệt là trong lĩnh vực dịch máy và so sánh ngữ nghĩa giữa các câu, ta nên tập trung vào việc phát triển các phương pháp đánh giá độ tương đồng ngữ nghĩa dựa trên các biểu diễn trừu tượng như AMR, thay vì chỉ dựa vào sự giống nhau về mặt từ vựng hay cú pháp.



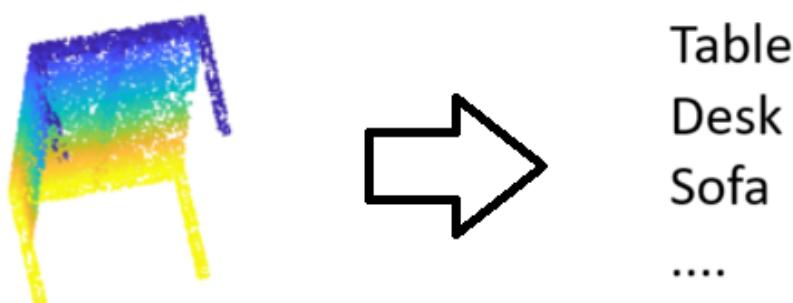
Hình 4.33: Motivation từ Curvature in Riemannian Geometry cho cấu trúc AMR trong NLP

4.3 Phân loại hình 3D sử dụng TFN

4.3.1 Tổng quan bài toán

Bài toán của báo cáo là phân loại hình 3D. Cụ thể hơn thì báo cáo sẽ làm việc về bài toán Point Cloud classification. Point Cloud Classification là một bài toán trong lĩnh vực Machine Learning và Computer Vision, đặc biệt liên quan đến việc xử lý dữ liệu 3D. Point Cloud là một tập hợp các điểm trong không gian ba chiều (3D), mỗi điểm được xác định bởi tọa độ (x, y, z) và đôi khi kèm theo các thông tin khác như màu sắc, cường độ ánh sáng, hay các đặc điểm khác. Các điểm này được thu thập từ các cảm biến 3D như LiDAR, máy quét 3D, hoặc từ các hình ảnh 2D thông qua các kỹ thuật như Structure from Motion (SfM).

Mục tiêu của bài toán Point Cloud Classification là phân loại toàn bộ đám mây điểm thành các nhãn (labels) khác nhau, chẳng hạn như xác định một đám mây điểm thuộc về một loại đối tượng nào đó (ví dụ: ô tô, người, cây cối, tòa nhà).



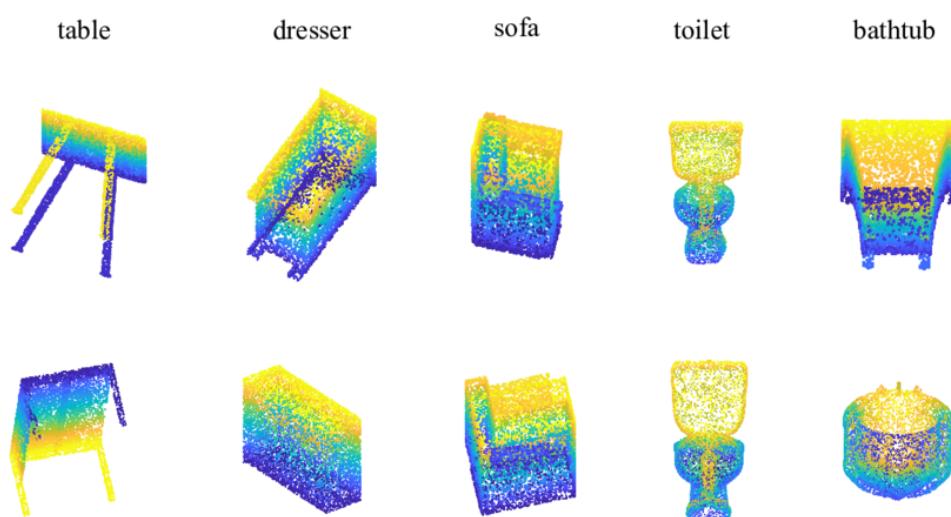
Một số thách thức chính trong Point Cloud Classification bao gồm:

- Không gian đầu vào có kích thước lớn: Đám mây điểm thường chứa hàng nghìn đến hàng triệu điểm, gây khó khăn cho việc xử lý và tính toán.

- Thiếu thông tin cấu trúc: Khác với hình ảnh 2D có cấu trúc lưới cố định, các điểm trong point cloud không có thứ tự cố định và không có cấu trúc mạng lưới rõ ràng, làm cho việc áp dụng các phương pháp truyền thống như Convolutional Neural Networks (CNN) trở nên khó khăn.
- Tính không đồng nhất: Point cloud có thể có mật độ không đồng nhất, nghĩa là một số vùng có thể chứa nhiều điểm hơn các vùng khác, tạo ra sự bất đối xứng trong dữ liệu.

4.3.2 Dữ liệu sử dụng

Ở đây, báo cáo sẽ sử dụng bộ dữ liệu về Point Cloud là bộ dữ liệu ModelNet10[14]. ModelNet10 là một bộ dữ liệu chuẩn trong lĩnh vực xử lý dữ liệu 3D, được sử dụng phổ biến trong các nghiên cứu và thí nghiệm liên quan đến Point Cloud Classification và 3D Object Recognition. Bộ dữ liệu này được giới thiệu lần đầu tiên bởi các nhà nghiên cứu tại Princeton University trong bài báo "3D ShapeNets: A Deep Representation for Volumetric Shapes"[14] vào năm 2015.



Hình 4.34: Ảnh minh họa một số dữ liệu trong tập dữ liệu ModelNet10

ModelNet10 bao gồm khoảng **3991** dữ liệu 3D được trích xuất từ các đối tượng trong thế giới thực và được sắp xếp thành 10 danh mục khác nhau. Những danh mục này bao gồm: Bed (Giường), Chair (Ghế), Desk (Bàn làm việc), Dresser (Tủ ngăn kéo), Monitor (Màn hình), Nightstand (Tủ đầu giường), Sofa (Ghế sofa), Table (Bàn), Toilet (Bồn cầu), Bathtub (Bồn tắm)

4.3.3 Phương pháp thực nghiệm

Ở đây, báo cáo sẽ chỉ thực nghiệm để thấy được sự hữu ích của các mạng *equivariant deep learning* trong bài toán liên quan đến dữ liệu 3D. Cụ thể hơn, báo cáo chỉ sử dụng TFN để cho thấy rằng dữ liệu được training là dữ liệu chưa xoay đi một góc nào đấy. Do đó, sau khi training xong, báo cáo sẽ xay dữ liệu một cách ngẫu nhiên rồi sau đó cho mô hình dự đoán.

Về phần dữ liệu, do hạn chế về mặt phần cứng là 15GB VRAM của Google Colab, báo cáo không thể training hết được toàn bộ dữ liệu do có một số dữ liệu có rất nhiều điểm (khoảng hơn 1000 điểm). Do đó, với mỗi loại nhãn của dữ liệu, báo cáo sẽ lấy một điểm dữ liệu cho mỗi loại nhãn rồi sau đó training mô hình TFN với dữ liệu này.

Về phần mô hình, báo cáo sử dụng thư viện E3NN để xây dựng mô hình TFN. TFN sẽ có hai layer do số lượng dữ liệu ít (10 dữ liệu). Giữa hai layer này sẽ có một hàm kích hoạt của thư viện E3NN có tên là Gate. Kích thước đầu ra cuối cùng sẽ là 10 với hàm kích hoạt là Softmax tương ứng cho xác suất mà một dữ liệu thuộc vào từng class.

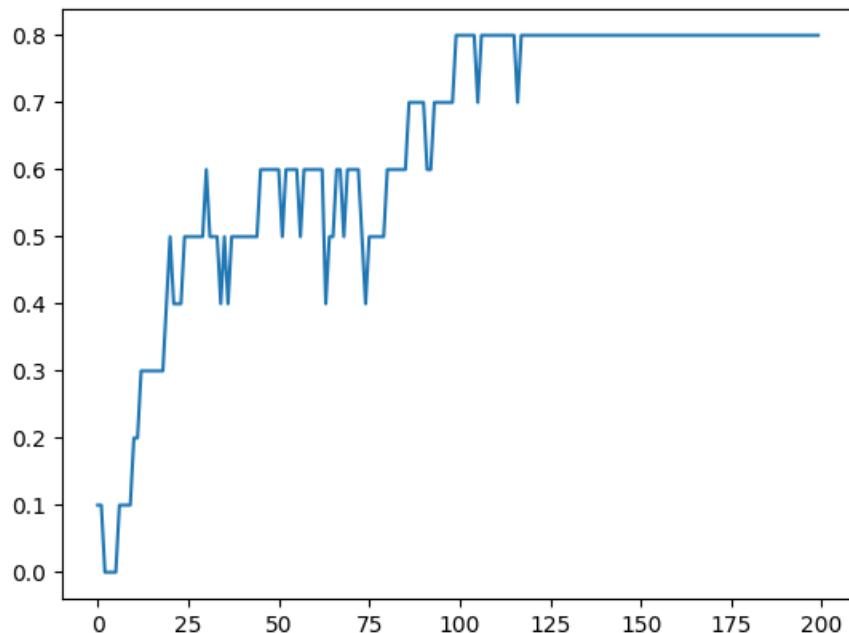
Mô hình sẽ được huấn luyện với 200 epochs với kích thước batch là 10. Hàm mất mát được sử dụng là Cross-Entropy loss và thuật toán tối ưu là Adam với learning rate là 10^{-3} .

Thiết bị huấn luyện	T4 GPU (Google Colab)
Số epochs	200
Kích thước batch	10
Hàm mất mát	Cross - Entropy
Thuật toán tối ưu	Adam
Hệ số học	10^{-3}

Bảng 4.1: Bảng thống kê các thông số huấn luyện mô hình

4.3.4 Kết quả thực nghiệm và đánh giá

Sau quá trình training, mô hình đạt được độ chính xác là 80% . Điều này cho thấy rằng mô hình vẫn chưa phân loại tốt hoàn toàn vì một phần là thiếu dữ liệu hoặc có thể là do có một số dữ liệu có cấu trúc khá là tương đồng nên mô hình chưa thể nhận biết được. Hình 4.35 dưới đây là biểu đồ theo dõi chất lượng dự đoán của mô hình.

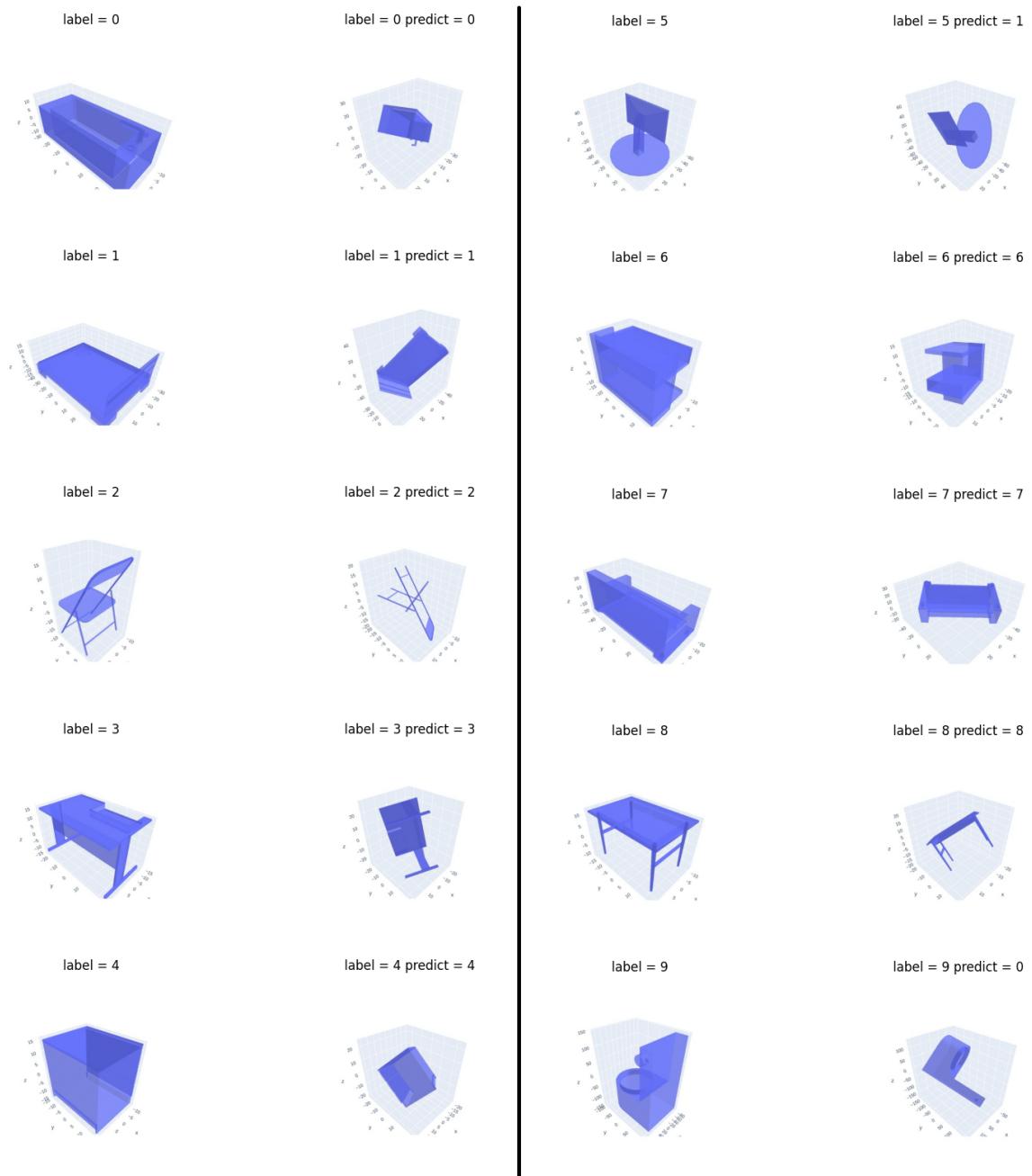


Hình 4.35: Biểu đồ theo dõi tính độ chính xác của mô hình trong quá trình training

Từ biểu đồ có thể thấy độ chính xác ban đầu khá thấp nhưng tăng dần qua

các epoch, cho thấy mô hình đang học và cải thiện dần. Tuy nhiên, trong quá trình training, độ chính xác dao động khá nhiều, đặc biệt là ở giai đoạn giữa, điều này có thể chỉ ra rằng mô hình phải tinh chỉnh khá nhiều để khớp với dữ liệu hơn. Cuối quá trình training, độ chính xác dần ổn định và đạt mức khoảng 0.8, nghĩa là mô hình có khả năng dự đoán đúng khoảng 80% dữ liệu.

Để thấy được độ hiệu quả của mô hình khi dự đoán dữ liệu đối với tác động của phép xoay, báo cáo sẽ xoay từng dữ liệu một góc bất kỳ trong không gian 3 chiều rồi cho mô hình dự đoán. Hình 4.36 cho thấy cái nhìn tổng quan về hiệu suất của mô hình với các hình có tiêu đề chỉ có chữ label là dữ liệu gốc còn đối với dữ liệu còn lại là dữ liệu đã được xoay.



Hình 4.36: Hình ảnh thống kê kết quả dự đoán của mô hình

Có thể thấy mô hình hoạt động khá tốt khi dự đoán chính xác 8/10 dữ liệu. Đối với những dữ liệu dự đoán sai thì chúng ta sẽ cần phải thêm dữ liệu đối với mô hình hoặc là thay đổi cấu trúc của mô hình để đạt được hiệu quả cao hơn.

Chương 5

KẾT LUẬN

Tài liệu tham khảo

- [1] Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. *Spherical CNNs*. 2018. arXiv: [1801.10130 \[cs.LG\]](https://arxiv.org/abs/1801.10130). URL: <https://arxiv.org/abs/1801.10130>.
- [2] AMR Guidelines Contributors. *AMR Guidelines*. Accessed on August 17, 2024. n.d. URL: <https://github.com/amrisi/amr-guidelines>.
- [3] *Da tap*. Accessed on April 23, 2024. 2024. URL: https://vi.wikipedia.org/wiki/%C4%90a_t%E1%BA%A1p.
- [4] *Geometric Deep Learning*. First Italian School. July 25–28, Pescara, Italy. 2022.
- [5] Robert Gilmore. *Lie Groups, Physics, and Geometry: An Introduction for Physicists, Engineers and Chemists*. Cambridge University Press, 2008.
- [6] Roe Goodman and Nolan R. Wallach. *Representations and Invariants of the Classical Groups*. Vol. 68. Cambridge University Press, 1998.
- [7] David J. Griffiths and Darrell F. Schroeter. *Introduction to Quantum Mechanics*. 3rd ed. Cambridge University Press, 2018.
- [8] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: [1609.02907 \[cs.LG\]](https://arxiv.org/abs/1609.02907). URL: <https://arxiv.org/abs/1609.02907>.

- [9] Kristof T. Schütt, Pieter-Jan Kindermans, Huziel E. Saucedo, Stefan Chmiela, Alexandre Tkatchenko, and Klaus-Robert Müller. *SchNet: A continuous-filter convolutional neural network for modeling quantum interactions*. 2017. arXiv: [1706.08566 \[stat.ML\]](https://arxiv.org/abs/1706.08566). URL: <https://arxiv.org/abs/1706.08566>.
- [10] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. *Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds*. 2018. arXiv: [1802.08219 \[cs.LG\]](https://arxiv.org/abs/1802.08219). URL: <https://arxiv.org/abs/1802.08219>.
- [11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: [1706.03762 \[cs.CL\]](https://arxiv.org/abs/1706.03762). URL: <https://arxiv.org/abs/1706.03762>.
- [12] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. *Graph Attention Networks*. 2018. arXiv: [1710.10903 \[stat.ML\]](https://arxiv.org/abs/1710.10903). URL: <https://arxiv.org/abs/1710.10903>.
- [13] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. *Harmonic Networks: Deep Translation and Rotation Equivariance*. 2017. arXiv: [1612.04642 \[cs.CV\]](https://arxiv.org/abs/1612.04642). URL: <https://arxiv.org/abs/1612.04642>.
- [14] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. *3D ShapeNets: A Deep Representation for Volumetric Shapes*. 2015. arXiv: [1406.5670 \[cs.CV\]](https://arxiv.org/abs/1406.5670). URL: <https://arxiv.org/abs/1406.5670>.
- [15] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. *Deep Sets*. 2018. arXiv: [1703.06114 \[cs.LG\]](https://arxiv.org/abs/1703.06114). URL: <https://arxiv.org/abs/1703.06114>.