AF2 Structure Module and IPA: A Self-Contained Walkthrough

Study Notes

December 8, 2025

Prompt (Context)

The goal of this note is to provide a self-contained, didactic explanation of the AlphaFold 2 structure module, in particular the invariant point attention (IPA) mechanism. The requirements are:

- Target the structure module and explain all relevant pieces (frames, IPA, BackboneUpdate, FAPE, torsions, etc.).
- For each algorithm line, explain what it does at a high level, what each term means, and the mathematical basis.
- When introducing a mathematical concept, explain it in plain English and from basic linear algebra/Transformer knowledge.
- Treat this as self-contained learning material, not just a brief summary.

1 What the Structure Module Does

By the time the structure module begins, the Evoformer trunk has produced two main tensors:

- A *single representation* $s_i^{\text{initial}} \in \mathbb{R}^{c_s}$ for each residue i,
- A *pair representation* $z_{ij} \in \mathbb{R}^{c_z}$ for each residue pair (i, j).

The structure module turns these into:

- a rigid frame (R_i, t_i) for each residue's backbone (one rotation matrix and one translation vector per residue),
- side-chain and backbone torsion angles,
- final 3D coordinates of all atoms $\tilde{x}_i^a$.

Algorithm 20 (structure module) can be summarized as:

1. Normalize and project the input features.

2. Initialize each residue's frame to identity rotation at the origin.
3. Run 8 identical structure blocks (sharing parameters). Each block:
   (a) Updates the single representation with IPA,
   (b) Applies a feedforward Transition MLP on the single representation,
   (c) Predicts a small frame update (quaternion + translation) per residue and composes it into each backbone frame,
   (d) Predicts torsion angles and applies auxiliary losses (per-iteration FAPE and torsion losses).
4. Finally, convert the final frames and torsion angles into rigid-group frames and then coordinates for all atoms, and apply the final FAPE and confidence losses.

Everything geometric in the structure module is expressed through *rigid frames* and *points expressed in frames*. IPA is how residues "look at each other" using these frames in a geometry-respecting way.

2 Frames and Rigid-Body Transformations

2.1 Rigid Motions and SE(3)

A rigid motion in 3D is a rotation followed by a translation. Mathematically:

- A rotation $R \in \mathbb{R}^{3\times 3}$ is an orthogonal matrix with $\det R = 1$ and $R^\top R = I$,
- A translation $t \in \mathbb{R}^3$ is a vector.

The rigid transform $T = (R, t)$ acts on a point $x \in \mathbb{R}^3$ by

$$T \circ x = Rx + t.$$

Composition of rigid transforms is

$$(R_2, t_2) \circ (R_1, t_1) = (R_2R_1, R_2t_1 + t_2).$$

The set of all such transforms, with this composition, is the Lie group SE(3).
AlphaFold uses exactly this representation: a *frame* for residue i is

$$T_i = (R_i, t_i).$$

2.2 Local vs. Global Coordinates

Think of each residue's frame as a small coordinate system attached to that residue.

- A point in global coordinates is $x \in \mathbb{R}^3$,
- The local coordinates of the same point in frame $T_i = (R_i, t_i)$ are

$$T_i^{-1} \circ x = R_i^{-1}(x - t_i).$$

Thus:

- Local $\to$ global: $x_{\text{global}} = T_i \circ x_{\text{local}}$,
- Global $\to$ local: $x_{\text{local}} = T_i^{-1} \circ x_{\text{global}}$.

The FAPE loss and IPA both rely on constantly mapping points between local and global frames.

3 Constructing Frames from Coordinates

During training, we have ground-truth atom positions from the PDB. For each residue, we need a canonical backbone frame so that:

- The origin is at Cα,
- Axes are aligned with the backbone geometry in a consistent, right-handed way.

3.1 Algorithm 21: `rigidFrom3Points`

A small routine `rigidFrom3Points`(x1, x2, x3) takes three 3D points and returns a frame (R, t) with:

1. $v_1 = x_3 - x_2$,
2. $v_2 = x_1 - x_2$,
3. $e_1 = v_1/\|v_1\|$,
4. $u_2 = v_2 - e_1(e_1^\top v_2)$,
5. $e_2 = u_2/\|u_2\|$,
6. $e_3 = e_1 \times e_2$,
7. $R = \text{concat}(e_1, e_2, e_3)$ as columns,
8. $t = x_2$,
9. Return (R, t).

This is Gram–Schmidt orthogonalization: from two non-collinear vectors with a common origin it builds an orthonormal basis. For backbone frames they choose

$$x_1 = N, \quad x_2 = \text{C}_\alpha, \quad x_3 = C,$$

so the backbone frame has origin at Cα.

3.2 Reflections and Chirality

Because e_3 comes from the cross product $e_1 \times e_2$, the frame behaves like a pseudo-vector frame under reflection. If we reflect all coordinates through the origin ($x \mapsto -x$), the frame does not simply become $(R, -t)$; the rotation part also picks up a reflection matrix. This is why FAPE and IPA can distinguish chiral pairs.

4 Structure Module Loop (Algorithm 20)

Here is the core of Algorithm 20, paraphrased.

1. Normalize the initial single representation s_i^{initial} and pair representation z_{ij}.
2. Project s_i^{initial} into the working single representation s_i.
3. Initialize all frames $T_i = (I, 0)$.
4. For $l = 1$ to $N_{\text{layer}} = 8$:
   (a) Update the single representation with IPA:
   $$\{s_i\} \mathrel{+}= \text{InvariantPointAttention}(\{s_i\}, \{z_{ij}\}, \{T_i\}).$$
   (b) Apply LayerNorm and dropout.
   (c) Apply a 3-layer ReLU MLP (Transition) with residual connection.
   (d) Update backbone frames:
   $$T_i \leftarrow T_i \circ \text{BackboneUpdate}(s_i).$$
   (e) Predict torsion angles and apply auxiliary per-layer losses:
   - Compute Cα positions from translations t_i,
   - Compute a backbone-only FAPE loss between current backbone frames and ground truth,
   - Compute a torsion-angle loss.

   (f) If $l < N_{\text{layer}}$, stop gradients through rotations R_i (but not translations) for stability.
5. After the loop, average per-layer auxiliary losses, compute rigid-group frames and coordinates for all atoms via `computeAllAtomCoordinates`, handle symmetric atom renaming, and compute final FAPE and confidence terms.

Each iteration thus has a two-stage update:

- IPA: updates hidden features s_i using geometry but does not move frames,
- BackboneUpdate: uses the updated s_i to move the frames themselves.

This is the "two-stage" update described in the main paper: first invariant feature updates via IPA, then an equivariant frame update.

5 Background: Standard Self-Attention

In standard Transformer self-attention, for each position i and head h:

- Query: $q_i^h = W_q^h s_i$,
- Key: $k_j^h = W_k^h s_j$,

- Value: $v_j^h = W_v^h s_j$.

Attention logits (before softmax) are

$$\ell_{ij}^h = \frac{1}{\sqrt{c}} (q_i^h)^\top k_j^h,$$

and weights are

$$a_{ij}^h = \mathrm{softmax}_j(\ell_{ij}^h).$$

The output for residue i and head h is

$$o_i^h = \sum_j a_{ij}^h v_j^h.$$

This interaction is purely in feature space; it has no explicit notion of 3D space.

6 Invariant Point Attention: Intuition

IPA preserves the basic self-attention structure but augments it with:

- biases from the pair representation z_{ij},
- point-based 3D distances defined using frames T_i.

Key ideas:

- For each residue i and head h, the network predicts several 3D query points in i's local frame,
- It also predicts key points and value points per residue and head,
- Using current frames T_i, it maps these points to global 3D, measures squared distances between query and key points, and uses those distances in the attention logits,
- Large distances give negative contributions to logits, so residues far apart in 3D get low attention regardless of feature similarity,
- The construction is invariant to a global rigid motion of all frames.

Intuitively, each head carries a small constellation of points per residue. Residue i prefers to attend to residues j whose constellations lie near its own when viewed in global 3D. At the same time, IPA still uses standard dot-product attention and pair bias, mixing:

1. feature similarity between residues i and j,
2. pair information z_{ij},
3. geometric proximity of learned points.

7 IPA: Algorithm 22 Line-by-Line

We follow Algorithm 22 and explain each object. Defaults: $N_{\text{head}} = 12$, $c = 16$, $N_{\text{query points}} = 4$, $N_{\text{point values}} = 8$.

7.1 Project Single Representations to Queries, Keys, Values

Line 1:

> Use a linear map (no bias) on s_i to produce three vectors per head: $q_i^h, k_i^h, v_i^h \in \mathbb{R}^c$.

For each residue i:

- apply a learned linear layer to s_i producing $3 \times N_{\text{head}} \times c$ numbers,
- split them into q_i^h, k_i^h, v_i^h for each head h.

This is exactly standard attention with specific head and channel sizes.

7.2 Predict Query and Key Points in Local Frames

Line 2:

> Apply another linear map (no bias) to s_i to produce 3D vectors $\tilde{q}_i^{hp}$ and $\tilde{k}_i^{hp}$ for each head h and point index $p = 1, \ldots, N_{\text{query points}}$.

These satisfy:

- $\tilde{q}_i^{hp} \in \mathbb{R}^3$: p-th query point for residue i and head h, in local coordinates of i,
- $\tilde{k}_i^{hp} \in \mathbb{R}^3$: p-th key point for residue i and head h, also in local coordinates.

Each head uses four local points per residue as geometric anchors encoding expected neighbor positions.

7.3 Predict Value Points

Line 3:

> Apply another linear map to s_i to produce 3D value points $\tilde{v}_i^{hp}$ for each head h and point index $p = 1, \ldots, N_{\text{point values}}$.

These are again in local coordinates. They will be:

- moved to global space,
- averaged across attended residues,
- mapped back to local coordinates,

yielding updated local points that are invariant/equivariant in the correct way.

7.4 Pair Bias from Pair Representation

Line 4:

> Compute a bias b_{ij}^h from the pair representation z_{ij} using a linear layer with no bias.

For each head h and pair (i, j) we obtain a scalar bias. This is analogous to the pair bias used throughout the Evoformer, allowing pair information to modulate attention.

7.5 Weight Factors w_C and w_L

Lines 5–6:

Define $w_C = \sqrt{2/(9N_{\text{query points}})}$ and $w_L = \sqrt{1/3}$.

These constants are shared across heads. The supplement chooses them by assuming all query/key components are unit Gaussians and computing variances of different contributions to the logits, aiming for:

- comparable scales for the dot-product term, pair-bias term, and point-distance term,
- overall logit variance of order 1.

7.6 Logits with Geometry

Line 7:

$$a_{ij}^h = \text{softmax}_j\Big(w_L\big[\sqrt{1/c}(q_i^h)^\top k_j^h + b_{ij}^h - \gamma_h w_C^2 \sum_p \|T_i \circ \tilde{q}_i^{hp} - T_j \circ \tilde{k}_j^{hp}\|^2\big]\Big).$$

Interpretation:

- The first term is standard scaled dot-product attention,
- b_{ij}^h is a learned pair bias,
- The last term is a geometric penalty: if learned point constellations for i and j are far apart in 3D, the penalty is large and the logit is reduced.

Thus the score combines feature similarity, pair information, and 3D proximity in a way that remains invariant to global rigid motions.

7.7 Outputs Per Head

Once we have a_{ij}^h:

1. **Pair-weighted pair representation** (line 8):

$$\tilde{o}_i^h = \sum_j a_{ij}^h z_{ij}.$$

2. **Standard value aggregation** (line 9):

$$o_i^h = \sum_j a_{ij}^h v_j^h.$$

3. **Point aggregation in 3D and mapping back to local** (line 10):

$$\tilde{o}_i^{hp} = T_i^{-1} \circ \Big(\sum_j a_{ij}^h (T_j \circ \tilde{v}_j^{hp})\Big).$$

This:

- maps value points from local to global via T_j,
- averages them in global space using attention weights,
- maps the result back to residue i's local frame with T_i^{-1}.

Algorithm 22 then concatenates $\tilde{o}_i^h$, o_i^h, $\tilde{o}_i^{hp}$, and their norms, and applies a linear layer to obtain $\tilde{s}_i$, which is added to s_i as a residual update.

8 Invariance vs. Equivariance in the Structure Module

We clarify two key notions:

- Invariance: transforming the input does not change the output,
- Equivariance: transforming the input transforms the output in a predictable way.

In the structure module:

- IPA is (approximately) invariant to global SE(3) transformations: if we apply a global rigid motion to all frames T_i and leave s_i unchanged, then
  - logits and weights a_{ij}^h are unchanged,
  - outputs $\tilde{o}_i^{hp}$, $\tilde{o}_i^h$, o_i^h in local coordinates are unchanged.
- BackboneUpdate is SE(3)-equivariant: if we globally transform all frames initially, the final frames are transformed by the same global transform.

Thus "IPA + BackboneUpdate" yields an overall block that is SE(3)-equivariant: invariant feature updates followed by an equivariant frame update.

9 BackboneUpdate: How Frames Move

Algorithm 23 (Backbone update) in the supplement predicts a quaternion and a translation per residue from s_i. The first quaternion component is fixed to 1, ensuring valid rotations and biasing updates toward small rotations.

Paraphrased behavior:

1. From s_i, apply a linear layer to get scalars (b_i, c_i, d_i) and a translation vector $\tilde{t}_i \in \mathbb{R}^3$,
2. Build the non-unit quaternion $(1, b_i, c_i, d_i)$,
3. Normalize it to unit length:

$$q_i = \frac{(1, b_i, c_i, d_i)}{\|(1, b_i, c_i, d_i)\|},$$

4. Convert q_i to a rotation matrix ΔR_i using the standard quaternion-to-matrix formula,
5. Define $\Delta T_i = (\Delta R_i, \tilde{t}_i)$,

6. Update the frame via composition:

$$T_i \leftarrow T_i \circ \Delta T_i.$$

Because the identity rotation corresponds to quaternion $(1, 0, 0, 0)$, small (b_i, c_i, d_i) produce quaternions close to identity, so early updates are small rotations. Since the update is expressed in each residue's current local frame, and composition with a global transform preserves relationships, this rule is equivariant under global rigid motions.

10 FAPE: Frame Aligned Point Error

IPA and BackboneUpdate operate on frames and points; FAPE is the loss that measures geometric error in a frame-consistent way. It is defined in Section 1.9.2 and Algorithm 28.

Given:

- predicted frames $\{T_i\}$ for each rigid group,
- predicted atom positions $\{\tilde{x}_j\}$,
- ground-truth frames $\{T_i^{\text{true}}\}$,
- ground-truth atom positions $\{\tilde{x}_j^{\text{true}}\}$,

Algorithm 28 computes:

1. Predicted atom j in local coordinates of frame i:

$$x_{ij} = T_i^{-1} \circ \tilde{x}_j.$$

2. True atom j in local coordinates of true frame i:

$$x_{ij}^{\text{true}} = (T_i^{\text{true}})^{-1} \circ \tilde{x}_j^{\text{true}}.$$

3. Distance per frame and atom:

$$d_{ij} = \sqrt{\|x_{ij} - x_{ij}^{\text{true}}\|^2 + \varepsilon},$$

4. Final scalar FAPE loss:

$$L_{\text{FAPE}} = \frac{1}{Z} \,\text{mean}_{i,j}\big(\min(d_{\text{clamp}}, d_{ij})\big),$$

with clamping to avoid large gradients.

Intuition:

- FAPE compares relative positions of each atom in each local frame between prediction and ground truth,
- It is invariant to global rigid motions of both predicted and ground truth structures, since local coordinates x_{ij}, x_{ij}^{true} are unchanged by a common global transform.

In the structure module:

- At each iteration they compute a low-cost FAPE using only backbone frames and Cα atoms,
- At the end they compute a full FAPE over all frames and atoms.

FAPE and IPA share the same foundation: points expressed relative to frames.

11 Torsion Angles, Rigid Groups, and All-Atom Reconstruction

11.1 Torsion-Angle Representation

From s_i and s_i^{initial}, a small residual network predicts, for each residue i and torsion type f (e.g. ω, ϕ, ψ, χ_1–χ_4), a vector $\tilde{\alpha}_i^f \in \mathbb{R}^2$.

These vectors are normalized to lie on the unit circle when used, effectively representing torsion angles as $(\cos\theta, \sin\theta)$. This:

- avoids discontinuities at 2π,
- allows constructing rotations without explicit trigonometric function calls.

An auxiliary loss encourages $\|\tilde{\alpha}_i^f\|$ to be close to 1.

11.2 Rigid Groups and Atom Templates

Each amino acid is decomposed into a small number of rigid groups (backbone and side-chain torsion segments). For each residue i and group f:

- there is a frame T_i^f obtained by rotating about the relevant bond axis by the torsion angle, starting from T_i and chaining down the side-chain tree,
- there is a fixed "template" position for each atom a in the local coordinates of group f.

Algorithm `computeAllAtomCoordinates` uses these to produce coordinates:

1. Build T_i^f for each group f using T_i and torsions α_i^f,
2. For each atom a belonging to group f, compute global coordinates

$$\tilde{x}_i^a = T_i^f \circ x_i^{a,\text{template}}.$$

Symmetric groups (e.g. certain χ_2 groups) are handled with special renaming procedures to avoid penalizing symmetry-equivalent side-chain orientations.

12 Putting It All Together

A single pass of the structure module can be viewed as follows.

12.1 Residue-Gas Initialization

We start from:

- single representations s_i,
- pair representations z_{ij},
- initial frames $T_i = (I, 0)$ (a "residue gas" with frames at the origin).

12.2 Structure Blocks

For each of the 8 structure blocks:

1. **IPA:**
   - converts s_i into queries, keys, values and point features per head,
   - uses frames T_i to compute distance-based attention biases in 3D,
   - mixes feature similarity, pair information, and geometry,
   - outputs updated s_i and local point features invariant to global transforms.
2. **Transition MLP:**
   - further refines s_i via a residual MLP, with LayerNorm and dropout.
3. **BackboneUpdate:**
   - predicts small frame updates ΔT_i via quaternions and translations,
   - composes these into T_i.
4. **Intermediate supervision:**
   - uses current backbone frames to compute a backbone-only FAPE and torsion losses.
5. **Stability trick:**
   - if not the last block, stop gradients through rotations R_i.

Over 8 iterations, this behaves like an SE(3)-equivariant message-passing network built explicitly on frames and attention.

12.3 Final Prediction and Loss

After the last block:

1. Use final frames and torsion angles to compute rigid-group frames and all-atom coordinates,
2. Rename symmetric atoms in the ground truth to best align with predicted symmetric groups,
3. Compute full FAPE over all rigid frames and atoms,
4. Compute per-residue lDDT-Cα to train the confidence predictor pLDDT.

Because FAPE strongly penalizes frame misalignment, and frames themselves are constructed from N–Cα–C and torsions, the loss encourages correct backbone geometry and side-chain packing.

13 Mathematical Background Recap

13.1 Dot Product and Norm

For $u, v \in \mathbb{R}^d$:

- dot product: $u^\top v = \sum_{k=1}^{d} u_k v_k$,
- norm: $\|u\| = \sqrt{u^\top u}$,
- squared Euclidean distance: $\|u - v\|^2 = \|u\|^2 + \|v\|^2 - 2u^\top v$.

These appear in:

- standard attention logits via $q^\top k$,
- IPA distance terms $\|T_i \circ \tilde{q} - T_j \circ \tilde{k}\|^2$,
- FAPE's local distances inside frames.

13.2 Softmax

For $\ell \in \mathbb{R}^n$,

$$\text{softmax}(\ell)_j = \frac{e^{\ell_j}}{\sum_k e^{\ell_k}}.$$

This maps arbitrary scores to positive weights summing to 1. Attention uses softmax over the key index j for each query i and head h.

13.3 Gram–Schmidt Orthogonalization

Given non-collinear $v_1, v_2 \in \mathbb{R}^3$:

1. $e_1 = v_1/\|v_1\|$,
2. remove the component of v_2 along e_1:

$$u_2 = v_2 - e_1(e_1^\top v_2),$$

3. normalize: $e_2 = u_2/\|u_2\|$,
4. define $e_3 = e_1 \times e_2$.

Then (e_1, e_2, e_3) is an orthonormal basis. This is what `rigidFrom3Points` implements.

13.4 Cross Product

For $a, b \in \mathbb{R}^3$, the cross product $a \times b$ is perpendicular to both, with length $\|a\|\,\|b\| \sin\theta$ (where θ is the angle between them) and direction given by the right-hand rule. It ensures right-handed frame orientation and is crucial for chirality.

13.5 Quaternions for Rotations

A unit quaternion $q = (a, b, c, d)$ with $a^2 + b^2 + c^2 + d^2 = 1$ represents a 3D rotation. The corresponding rotation matrix is

$$R = \begin{pmatrix} a^2 + b^2 - c^2 - d^2 & 2bc - 2ad & 2bd + 2ac \\ 2bc + 2ad & a^2 - b^2 + c^2 - d^2 & 2cd - 2ab \\ 2bd - 2ac & 2cd + 2ab & a^2 - b^2 - c^2 + d^2 \end{pmatrix}.$$

Quaternions are convenient for learning rotations because:

- normalizing a quaternion guarantees a valid rotation,
- small deviations from $(1, 0, 0, 0)$ correspond to small rotations.

13.6 Invariance vs. Equivariance, Again

Summarizing:

- IPA is invariant to global SE(3) because it uses relative distances expressed via frames and maps value points back to local frames,
- BackboneUpdate composes local transforms, so a global transform applied to input frames yields outputs transformed in the same way.

14 Big-Picture Intuition for IPA

One can picture each IPA head as:

- defining a learned pattern of points in each residue's local frame,
- when residue i looks at residue j, using current frames to map these point sets into global space and checking how well they align, while also using feature similarity and pair information,
- assigning higher attention to residues whose patterns line up and whose features agree,
- aggregating both value features and value points from neighbors, then mapping aggregated points back into residue i's frame.

Thus each head is a geometry-aware attention mechanism that:

- biases attention towards geometrically compatible residues,
- transports geometric information in a frame-invariant way.

This design allows the structure module to iteratively refine structure in 3D while being robust to arbitrary choices of global coordinate system.