Expansion on Algorithm 20: Dropout, LayerNorm, and Transition MLP

Study Notes

December 8, 2025

1 Context

We focus on lines 6–9 of Algorithm 20 (the structure module loop) in the AF2 supplement, and in particular on items labeled 4.b and 4.c in a paraphrased description. From the supplement:

6: $\{s_i\}$ += InvariantPointAttention($\{s_i\}, \{z_{ij}\}, \{T_i\}$)
7: $s_i \leftarrow \text{LayerNorm}(\text{Dropout}_{0.1}(s_i))$
8: $s_i \leftarrow s_i + \text{Linear}(\text{ReLU}(\text{Linear}(\text{ReLU}(\text{Linear}(s_i)))))$
9: $s_i \leftarrow \text{LayerNorm}(\text{Dropout}_{0.1}(s_i))$.

The mapping to the paraphrased bullets is:

- 4.a: IPA step (line 6),
- 4.b: "Apply LayerNorm and dropout" (line 7),
- 4.c: "Apply a 3-layer ReLU MLP (Transition) with residual connection" (lines 8 and 9).

We now unpack 4.b and 4.c carefully.

2 Step 4.b: $s_i \leftarrow \text{LayerNorm}(\text{Dropout}_{0.1}(s_i))$

2.1 Literal Operation

After the IPA update,

$$s_i \leftarrow s_i + \text{IPA}_i(s, z, T),$$

each residue's single embedding $s_i \in \mathbb{R}^{c_s}$ is transformed as follows.

1. Dropout with rate 0.1. Each channel of s_i is independently either:

- set to zero with probability 0.1, or
- kept and scaled by $1/(1-0.1) = 1/0.9$ with probability 0.9.

If $m_i \in \{0, 1/(1-p)\}^{c_s}$ is a random mask (with $p = 0.1$), dropout yields

$$\tilde{s}_i = \text{Dropout}_{0.1}(s_i) = m_i \odot s_i.$$

2. Layer normalization. LayerNorm is applied across the channel dimension independently for each residue. Define

$$\mu_i = \frac{1}{c_s}\sum_{k=1}^{c_s} \tilde{s}_{i,k}, \qquad \sigma_i^2 = \frac{1}{c_s}\sum_{k=1}^{c_s}(\tilde{s}_{i,k} - \mu_i)^2.$$

LayerNorm produces

$$\mathrm{LayerNorm}(\tilde{s}_i)_k = \gamma_k \frac{\tilde{s}_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}} + \beta_k,$$

with learnable scale γ_k and bias β_k per channel.

Thus line 7 is exactly

$$s_i \leftarrow \mathrm{LayerNorm}(\mathrm{Dropout}_{0.1}(s_i)).$$

2.2 Conceptual Rationale

- **Dropout** injects noise into the representation. After IPA, some channels of s_i are randomly zeroed, so subsequent updates cannot depend too heavily on any single component. The supplement explicitly states that dropout is applied to the outputs of IPA and the Transition layer.
- **LayerNorm** stabilizes per-residue activations. It keeps the mean and variance of s_i under control, making learning easier and reducing the risk of exploding or vanishing activations as we iterate the structure block.
- The combination "Dropout then LayerNorm" means the normalized activations already incorporate the effect of dropout noise; the Transition MLP therefore sees a noisy but rescaled vector with controlled variance.

Intuitively, line 7 performs:

> Take the per-residue features after IPA, inject some noise via dropout, then rescale them to a standard form before feeding them into the small MLP.

3 Step 4.c: Transition — 3-Layer ReLU MLP with Residual

Lines 8 and 9 implement the Transition for the structure module's single representation.

3.1 Code Structure

Line 8:

$$s_i \leftarrow s_i + \mathrm{Linear}(\mathrm{ReLU}(\mathrm{Linear}(\mathrm{ReLU}(\mathrm{Linear}(s_i))))),$$

line 9:

$$s_i \leftarrow \mathrm{LayerNorm}(\mathrm{Dropout}_{0.1}(s_i)).$$

It is easiest to rewrite line 8 in explicit mathematical steps. Let the input to this block be the normalized vector from line 7, and denote it by x_i:

$$x_i = s_i \quad \text{(after line 7)}.$$

1. First linear layer.

$$h_i^{(1)} = W^{(1)} x_i + b^{(1)}, \qquad h_i^{(1)} \in \mathbb{R}^{c_s}.$$

2. First ReLU.

$$r_i^{(1)} = \mathrm{ReLU}(h_i^{(1)}).$$

3. Second linear layer.

$$h_i^{(2)} = W^{(2)} r_i^{(1)} + b^{(2)}, \qquad h_i^{(2)} \in \mathbb{R}^{c_s}.$$

4. Second ReLU.

$$r_i^{(2)} = \mathrm{ReLU}(h_i^{(2)}).$$

5. Third linear layer.

$$h_i^{(3)} = W^{(3)} r_i^{(2)} + b^{(3)}, \qquad h_i^{(3)} \in \mathbb{R}^{c_s}.$$

6. Residual addition.

$$s_i \leftarrow x_i + h_i^{(3)}.$$

This stack

$$x_i \mapsto x_i + W^{(3)}\,\mathrm{ReLU}\big(W^{(2)}\,\mathrm{ReLU}(W^{(1)} x_i + b^{(1)}) + b^{(2)}\big) + b^{(3)}$$

is exactly what line 8 encodes.

Two details from the algorithm text are important:

- They note "all intermediate activations $\in \mathbb{R}^{c_s}$", which tells us they do *not* expand the hidden dimension here; each layer remains of size c_s.
- They name this block the *Transition* for the structure module, analogous to the MSA Transition in the Evoformer, which is also a small MLP with a residual connection.

Line 9 then applies dropout and LayerNorm once more:

$$s_i \leftarrow \mathrm{LayerNorm}(\mathrm{Dropout}_{0.1}(s_i)).$$

Thus the full 4.c block is:

1. Apply a residual 3-layer ReLU MLP to the normalized IPA output,
2. Apply dropout and LayerNorm to the result.

3.2 Why This Pattern?

Conceptually this mirrors a Transformer block:

- attention-like update (IPA) plus residual,
- normalization + MLP + residual,
- another normalization.

The difference is that here LayerNorm and dropout appear on both sides of the MLP:

1. Normalize the IPA output,
2. Use that stable input for the Transition MLP,
3. Add the MLP output back to the normalized input (residual),
4. Normalize again, with dropout, to keep the statistics of s_i well behaved for the next structure block iteration.

Mathematically, the Transition MLP provides *non-geometric flexibility* at the per-residue feature level:

- IPA is heavily constrained by geometry and equivariance,
- The Transition allows each residue to apply a learned nonlinear transformation to its feature vector, mixing channels arbitrarily without affecting coordinates,
- The residual connection means the MLP only needs to learn a correction to the identity map rather than a completely new mapping, improving optimization.

In summary, steps 4.b and 4.c together:

> Take the feature update from IPA, regularize and normalize it, pass it through a small per-residue MLP with a residual connection, and normalize again so that the updated feature vector is numerically stable before predicting the next frame update.