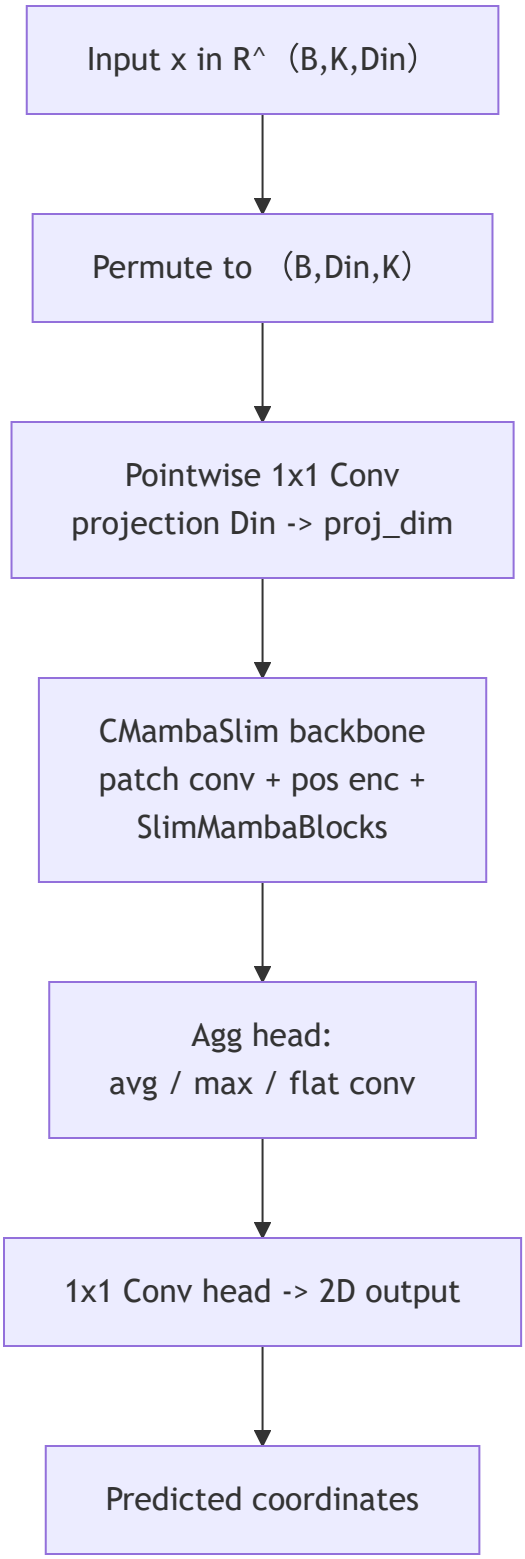
Ablation Architectures Overview

This note summarizes the three backbones that will be compared in the ablation: the refactored regression stack we currently ship, the original Channel Mamba （C-Mamba） design, and the vanilla Mamba block. All derivations below assume the pure floating-point （non-quantized） implementation so we can focus on architectural effects. For each one the main dataflow and the key equations are listed so they can be referenced when configuring experiments.

1. Current Refactored CMamba Regressor



Equations

- Permutation and projection

  `x_perm = permute（x, （0,2,1））`

  `h0 = Conv1x1（x_perm）` where `Conv1x1: R^（Din） -> R^（proj_dim）` .

- Patch embedding and positional encoding

  `z = Conv_patch（h0）`（`kernel = patch_len`，`stride = stride`）.

  `L = floor（（K - patch_len） / stride） + 1`.

  `z ∈ R^（B, proj_dim, L）` and `PE ∈ R^（L, d_model）`.

  `z = z + pe_scale * PE` if positional encoding is enabled.

- SlimMambaBlock （per token index `t`）:

```
h = RMSNorm（z）
[u_t, z_t] = W_in * h_t
if DWConv: u_t = DWConv（u_t）
u_t = SiLU（u_t）
λ_t = sigmoid（W_dt * u_t）
s_t = λ_t ⊙ s_{t-1} + （1 - λ_t） ⊙ u_t
g_t = SiLU（z_t） （skipped when gate_off）
y_t = W_out（ s_t ⊙ g_t ）
```

- **Note on EMA simplification**

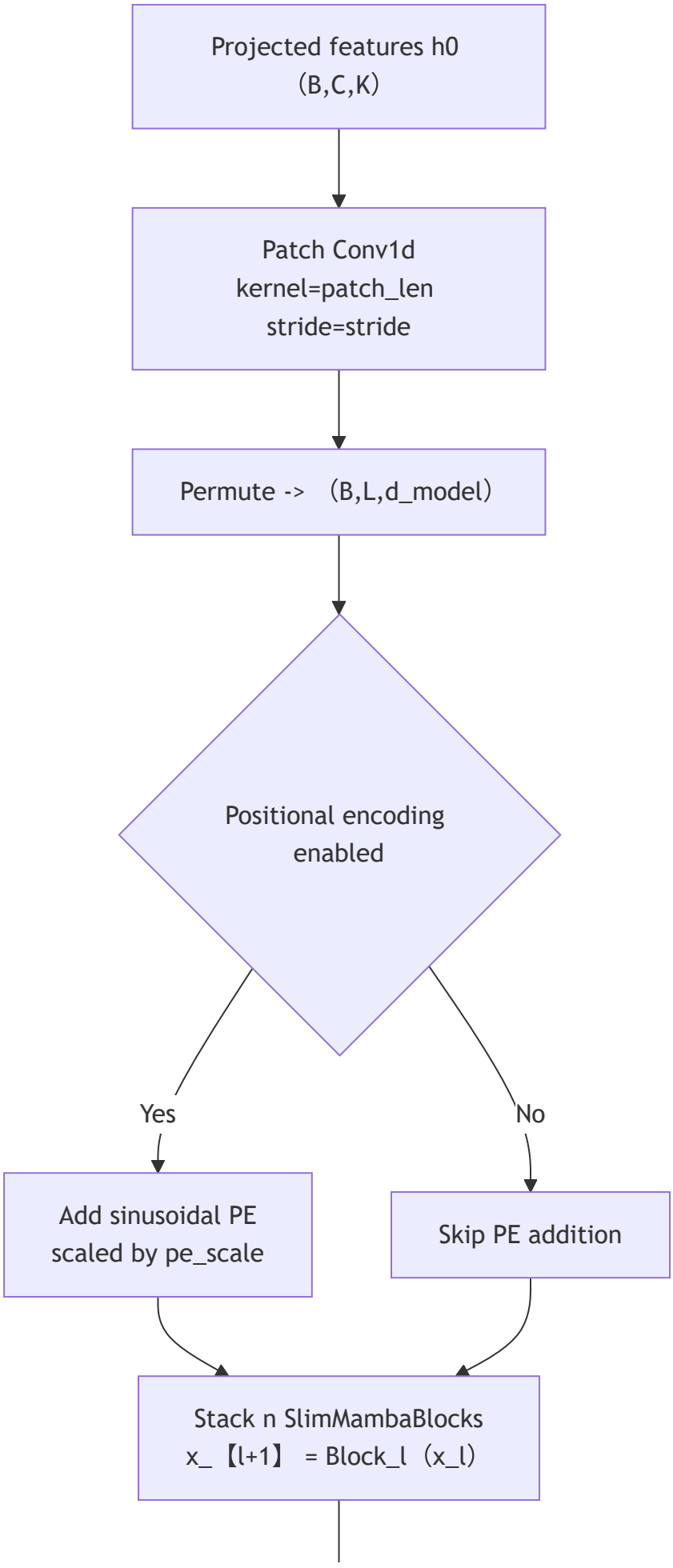
  The selective-scan block here is an EMA-style SSM with input-conditioned decay only; it does **not** implement the full Mamba formulation with learnable high-order SSM matrices `（A,B,C,D）`, discretization, low-rank `dt_rank`, or the fused CUDA parallel scan/streaming state mechanism. Any ablation that requires those official Mamba behaviors must regard them as absent in node `D` of the flowchart （the CMambaSlim backbone block）.

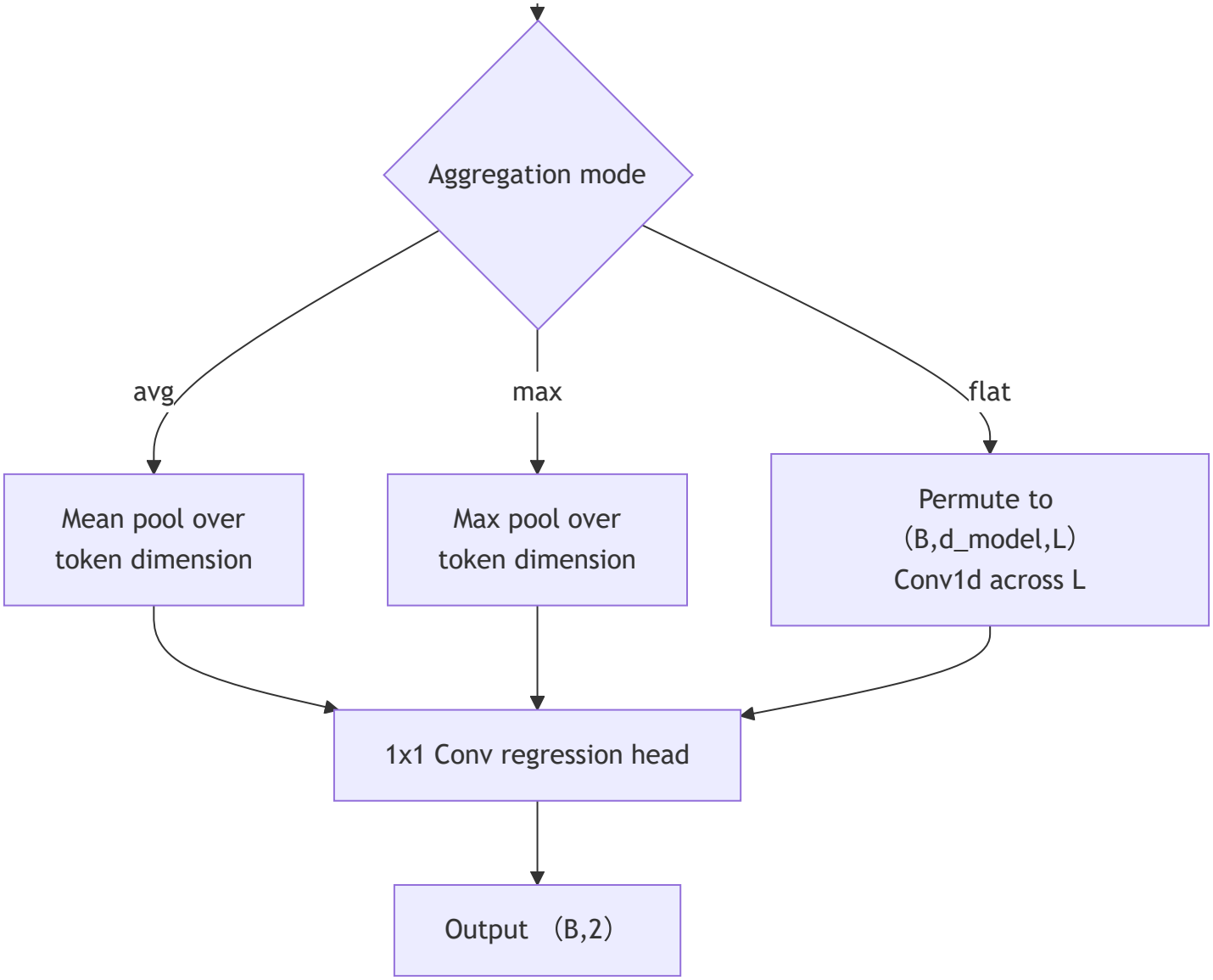
- Aggregation and regression head

  `y_flat = AggPool（y）` where AggPool ∈ {mean, max, flatten-conv}.

  `pred = Conv1x1_head（y_flat）` → `（B,2）` final positions.

Backbone （Non-Quantized SlimMamba Reference）


Projected features h0
(B,C,K)
Patch Conv1d
kernel=patch_len
stride=stride
Permute -> (B,L,d_model)
Positional encoding
enabled
Yes
No
Add sinusoidal PE
scaled by pe_scale
Skip PE addition
Stack n SlimMambaBlocks
x_【l+1】 = Block_l (x_l)



- **Patch embedding （Conv1d without quantization）**

```
x_patch = Conv1d_patch（h0）
x_patch = permute（x_patch, （0,2,1））  # （B, L, d_model）
```

  where `Conv1d_patch` uses standard weights `W_patch ∈ R^（d_model, proj_dim, patch_len）` and bias `b_patch`.

- **Sinusoidal position encoding** （when enabled）：

  `PE[i, 2j] = sin（i / 10000^（2j/d_model））`，

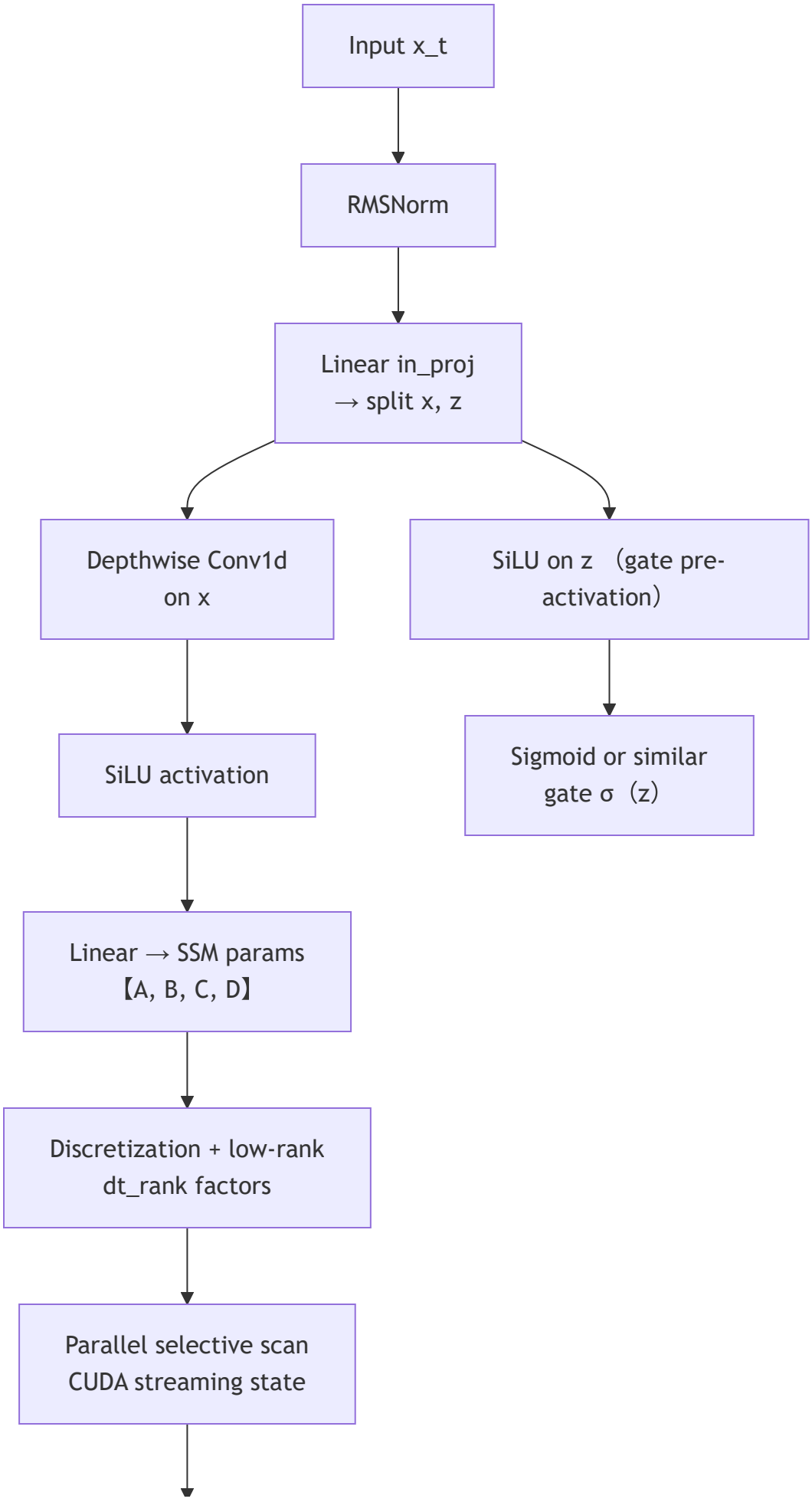
  `PE[i, 2j+1] = cos（i / 10000^（2j/d_model））`，

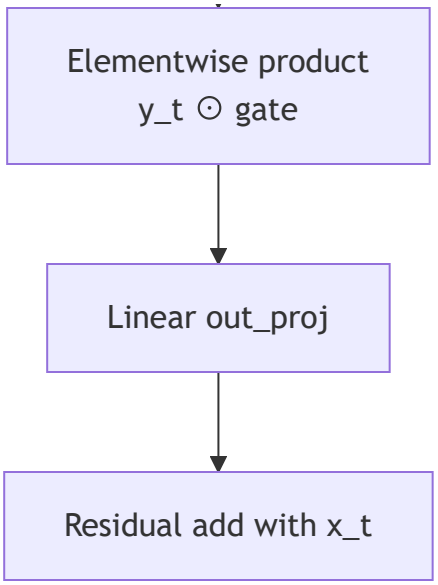
  `x_patch = x_patch + pe_scale * PE`.

- **SlimMambaBlock stack**: `x_{ℓ+1} = SlimBlock_ℓ（x_ℓ）` for `ℓ = 1..n_layer`. See Section 4 for the full set of ablation knobs inside each block.
- **Aggregation head** （non-quantized）：
  - `avg`：`y = mean（x_last, dim=1）`

    - max： y = max（x_last, dim=1）
    - flat： y = Conv1d_flat（permute（x_last, （0,2,1））） .
- **Regression head**: pred = Conv1x1_head（y） with weights W_head ∈ R^（2, proj_dim, 1） .

2. Mamba （Original）


Input x_t
RMSNorm
Linear in_proj
→ split x, z
Depthwise Conv1d
on x
SiLU on z （gate pre-
activation）
SiLU activation
Sigmoid or similar
gate σ（z）
Linear → SSM params
【A, B, C, D】
Discretization + low-rank
dt_rank factors
Parallel selective scan
CUDA streaming state



Equations

- Stem projection

  `h = Conv1x1（x）`

  `h = DWConv（h）` （optional） to mix local temporal context.
- Channel selective scan （treat `c` as sequence index）

```
[u_c, z_c] = W_c * h_c
λ_c = sigmoid（W_dt_c * u_c）
s_c = λ_c ⊙ s_{c-1} + （1 - λ_c） ⊙ u_c
g_c = SiLU（z_c）
y_c = W_o（ s_c ⊙ g_c ）
```

- Final projection

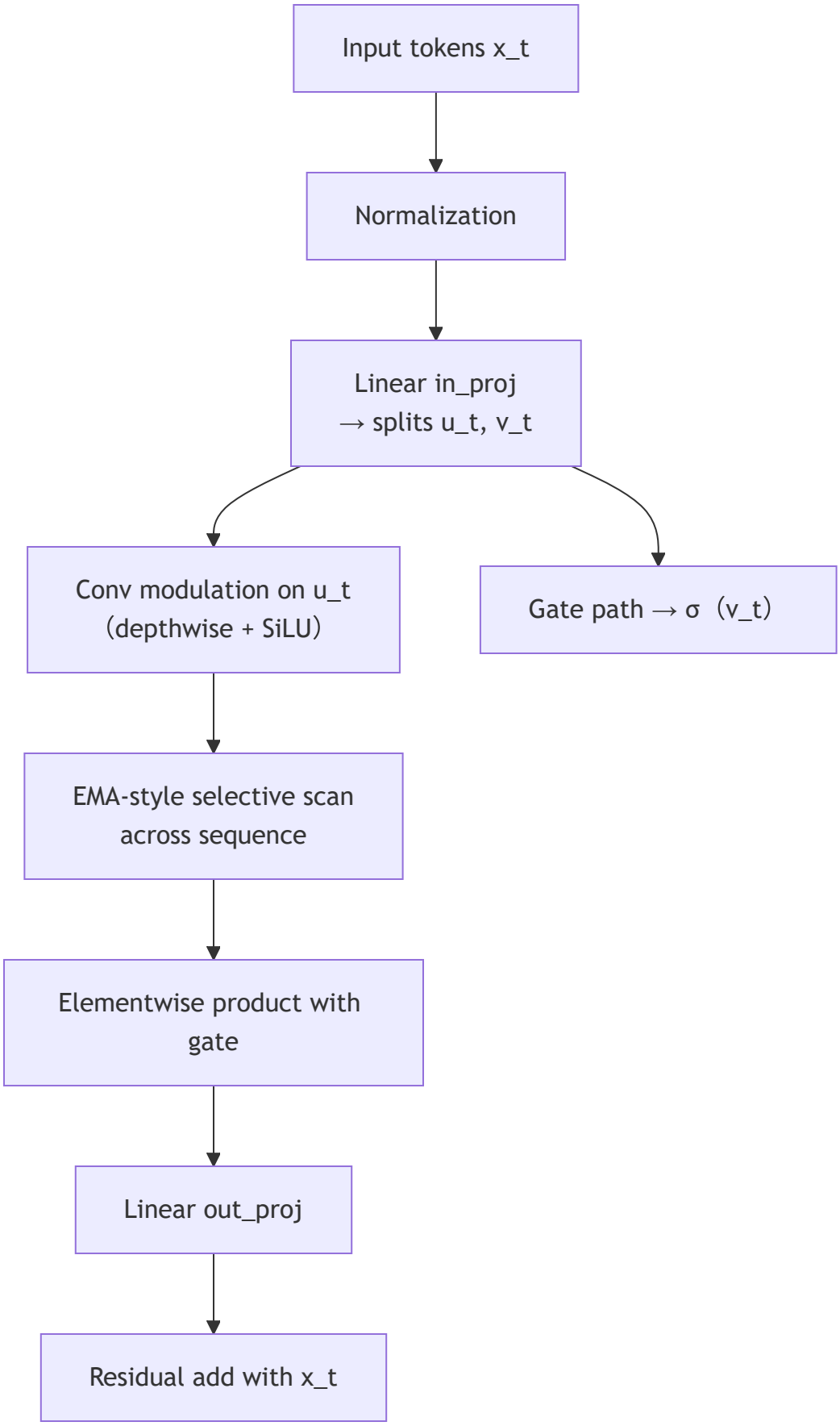
  `y = reshape（y_c back to B x C x T）`

  `pred = Head（y）` （e.g., global average over T followed by linear layer）.

The hallmark is that the recurrent state moves along channels instead of time tokens, which emphasizes cross-channel dependency before the task head.

In the flowchart above, nodes `F`, `G`, and `H` explicitly capture the official Mamba components: learnable high-order SSM parameters `【A,B,C,D】`, discretization with low-rank `dt_rank`, and the CUDA-parallel selective scan with streaming state. These are the capabilities absent from the CMambaSlim backbone block `D` in Section 1.

3. SlimMamba Block



Equations

- In-projection and convolutional modulation

  `h_t = Norm（x_t）`

  `[u_t, v_t] = W_in * h_t`

  `ũ_t = u_t + Conv1d_depthwise（u_t）` （optional conv）.
- Selective scan state update （sequence dimension）

  `λ_t = sigmoid（W_dt * ũ_t）`

  `s_t = λ_t ⊙ s_{t-1} + （1 - λ_t） ⊙ ũ_t`
- Output

  `g_t = SiLU（v_t）`

  `y_t = W_out（ s_t ⊙ g_t ）`

  `x_{t+1} = x_t + y_t`.

This block processes tokens in chronological order and is the baseline to compare against when removing the channel-oriented modifications or the simplified SlimMamba variant. Its selective scan is the simplified EMA version (no `【A,B,C,D】`, no discretization, and no CUDA streaming state), matching the implementation used inside the CMambaSlim backbone.

4. SlimMambaBlock Ablation Targets （No Quantization）

All ablations on the regression backbone should reference the non-quantized SlimMambaBlock defined below. Each toggle can be independently disabled to isolate its contribution.

1. **Depthwise Convolution （DWConv） Path**
   - Baseline: `u_t = DWConv（u_t）` with kernel size `k = d_conv` and padding `floor（k/2）` so the temporal length stays constant.
   - Ablation: bypass DWConv by setting `u_t = u_t`.
2. **Gating Branch**
   - Baseline gate: `g_t = SiLU（z_t）` and `s_t ← s_t ⊙ g_t`.
   - Ablation: disable gate （`g_t = 1`） to measure the effect of multiplicative modulation.
3. **Selective-Scan Parameters**
   - Baseline recurrence: `λ_t = sigmoid（W_dt u_t）` and `s_t = λ_t ⊙ s_{t-1} + （1 - λ_t） ⊙ u_t`.
   - Ablation ideas: replace `λ_t` with a constant decay or remove residual accumulation entirely （`s_t = u_t`）.
4. **Residual Scaling**

- Baseline: block output `y_t = W_out（s_t）` followed by residual `x_{t+1} = x_t + y_t`.
- Ablation: test without residual （`x_{t+1} = y_t`） or add learnable scaling `α`（`x_{t+1} = x_t + α y_t`）.

5. **Positional Encoding Usage**
   - Baseline inherits the global PE toggle from the backbone.
   - Ablation: disable PE specifically for selected layers to measure reliance on absolute positions.

These handles provide a concrete checklist for the upcoming experiments while keeping the analysis in the floating-point （non-quantized） regime.