Building a Transformer

David I. Inouye

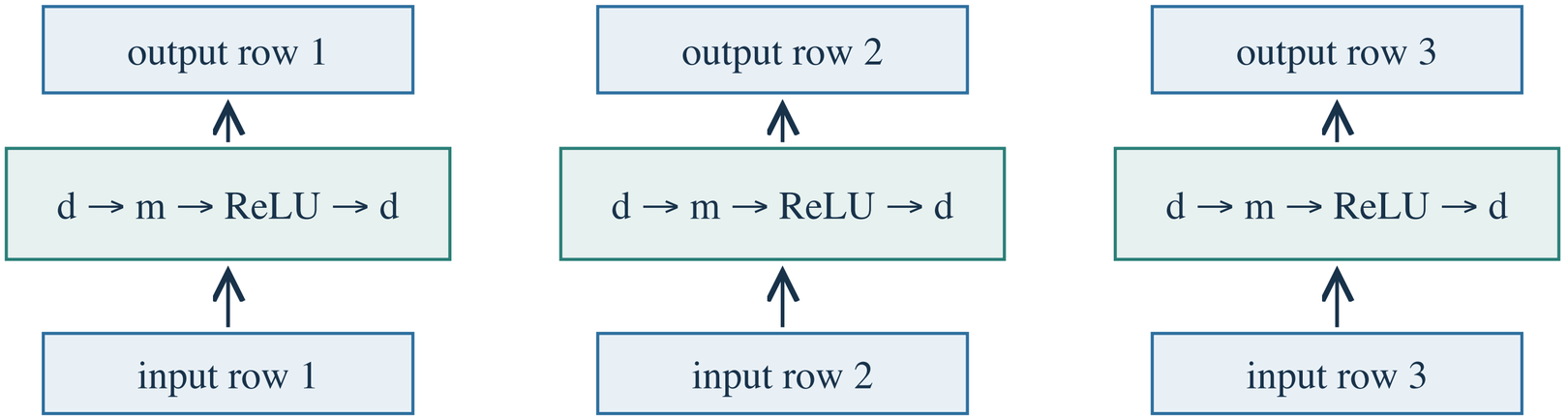
Dynamic Content · Last Updated: October 30, 2026

A transformer combines context mixing and feature transformation

Transform features → preserve a residual path → normalize → stack blocks.

Keep the same task: use an email prefix to predict its next token.

Attention mixes positions; a shared MLP transforms each row



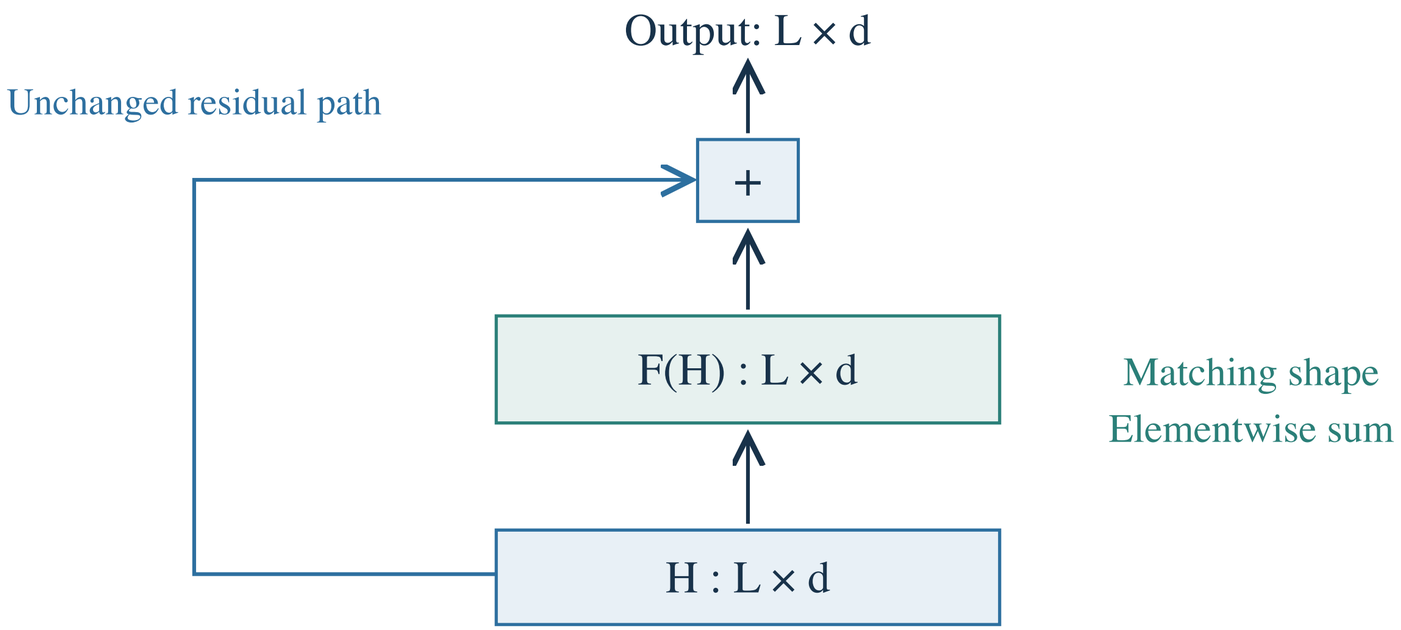
Same MLP at every position; no mixing between positions

$$\mathrm{MLP}(H) = \mathrm{ReLU}(HW_1^\top + \mathbf{1}\boldsymbol{b}_1^\top)W_2^\top + \mathbf{1}\boldsymbol{b}_2^\top$$

$$H \in \mathbb{R}^{L\times d}, \quad W_1 \in \mathbb{R}^{m\times d}, \quad W_2 \in \mathbb{R}^{d\times m}$$

Same parameters at every position; separate parameters in each block.

A residual path adds a learned update without changing the shape



$$H_{\text{out}} = H + F(H)$$

The attention output projection and the MLP's last layer both return d **features**.

LayerNorm normalizes one token's features at a time

Like BatchNorm: subtract a mean, divide by a standard deviation, then learn a scale and shift.

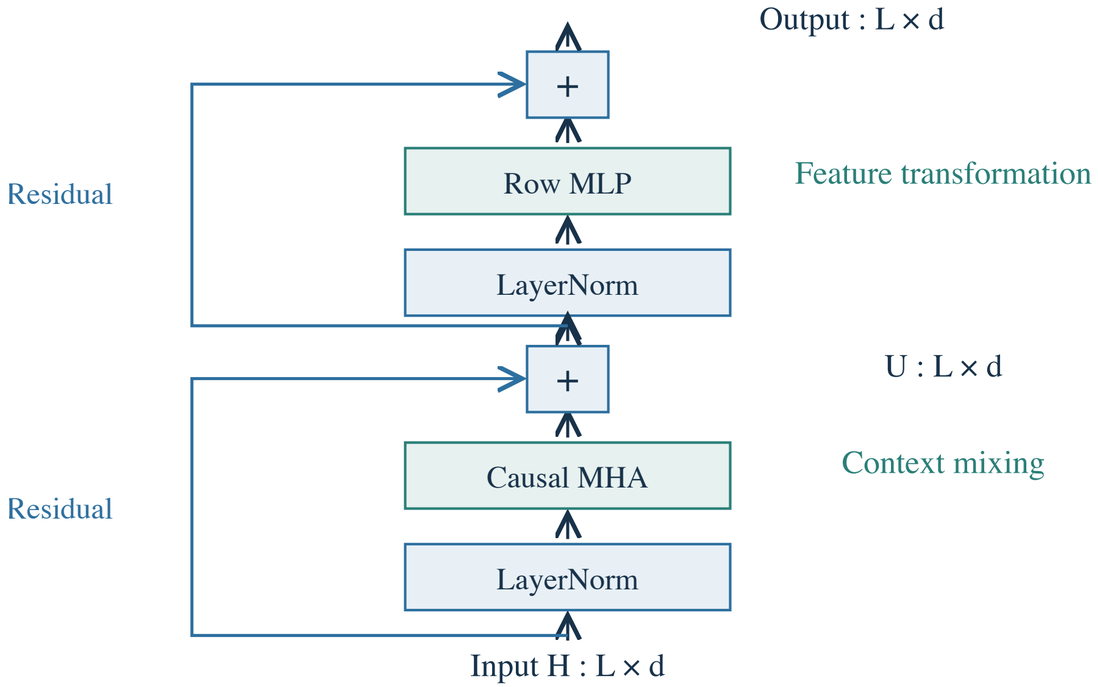
$$\mu_\ell = \frac{1}{d}\sum_{j=1}^{d} h_{\ell j}, \qquad \sigma_\ell^2 = \frac{1}{d}\sum_{j=1}^{d} (h_{\ell j} - \mu_\ell)^2$$

$$\mathrm{LN}(\boldsymbol{h}_\ell) = \boldsymbol{\gamma} \odot \frac{\boldsymbol{h}_\ell - \mu_\ell \mathbf{1}}{\sqrt{\sigma_\ell^2 + \epsilon}} + \boldsymbol{\beta}$$

- **BatchNorm (CNN):** one channel across batch examples and spatial locations.
- **LayerNorm:** one token across its d features—the last axis of `[B,L,d]`.

At evaluation, LayerNorm still uses **that token's statistics**; BatchNorm usually uses running statistics.

A pre-norm block adds two shape-preserving updates



$$U = H + \mathrm{MHA}_{\mathrm{causal}}(\mathrm{LN}_1(H))$$

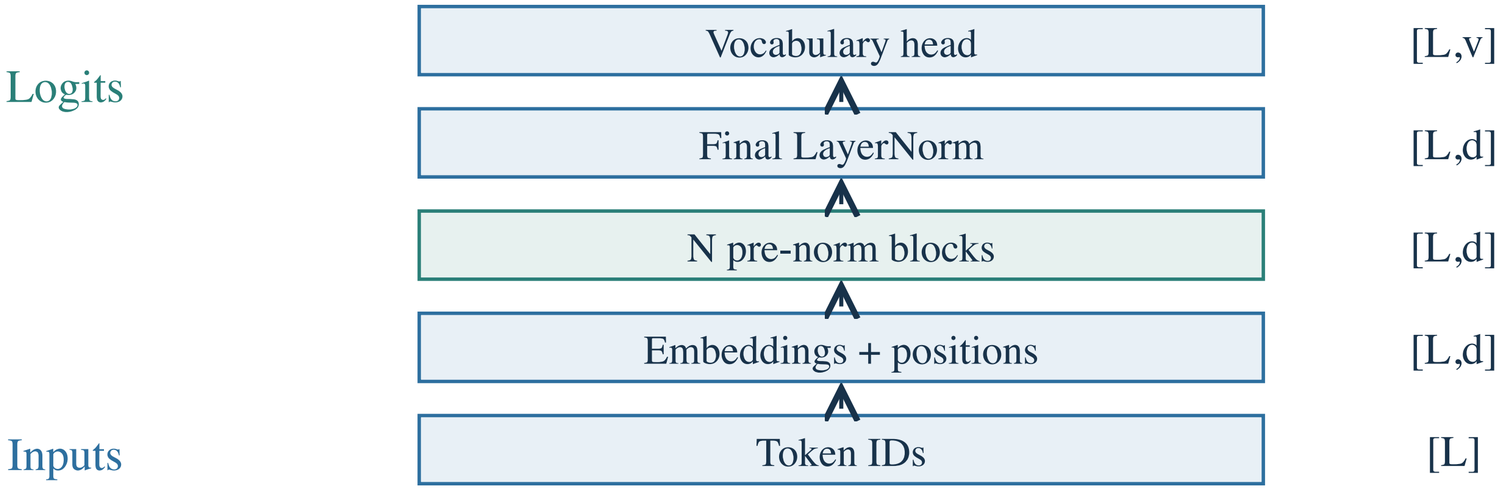
$$H_{\mathrm{out}} = U + \mathrm{MLP}(\mathrm{LN}_2(U))$$

Pre-norm is the usual modern LLM default. The original Transformer used post-norm.

Pre-norm	Post-norm
$H + F(\mathrm{LN}(H))$	$\mathrm{LN}(H + F(H))$
Normalize **before the branch**.	Normalize **after addition**.

Pre-norm leaves the bypass untouched.

Stacked blocks turn token features into next-token logits

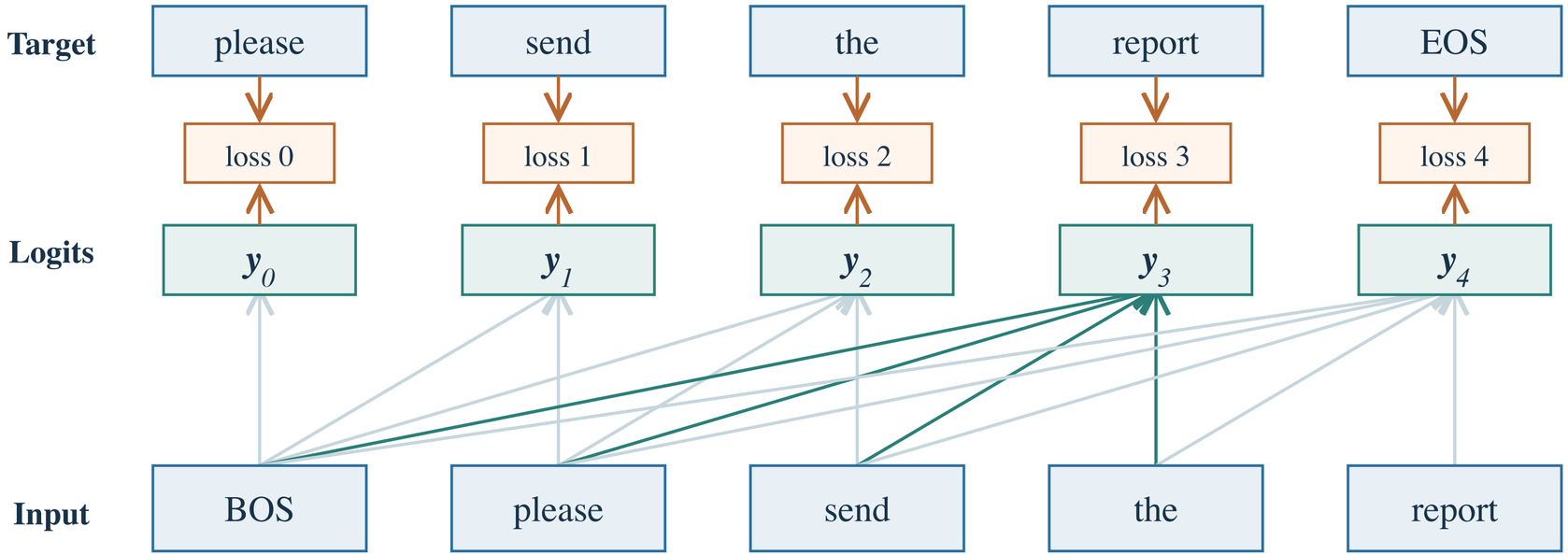


RoPE alternative: omit additive positions; rotate Q/K within each block

$$H^{(0)} = E[w_{0:L}, :] + P, \qquad Y = \mathrm{LN}_{\mathrm{final}}(H^{(N)})W_{\mathrm{vocab}}^{\mathsf{T}} + \mathbf{1}\boldsymbol{c}^{\mathsf{T}}$$

$$H^{(N)} \in \mathbb{R}^{L\times d}, \qquad W_{\mathrm{vocab}} \in \mathbb{R}^{v\times d}, \qquad Y \in \mathbb{R}^{L\times v}$$

One length-L input solves L prefix-prediction problems



Read upward: one causal pass → L logit rows → L shifted-target losses

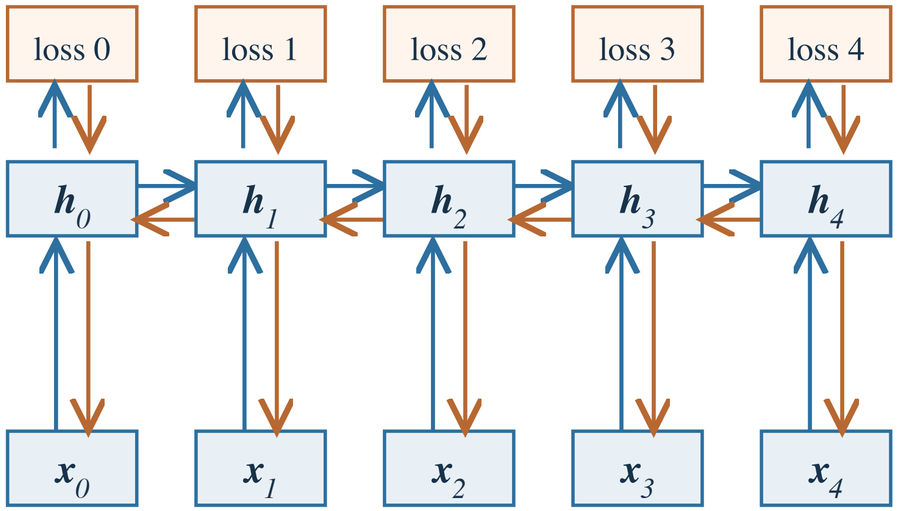
$$\mathcal{L}(\theta) = -\frac{1}{L}\sum_{\ell=0}^{L-1} \log p_\theta(w_{\ell+1} \mid w_0, \ldots, w_\ell).$$

Known inputs allow parallel work; the mask preserves each allowed prefix.

Backpropagation follows different dependency graphs

Same inputs: BOS, please, send, the, report • Same targets: please, send, the, report, EOS

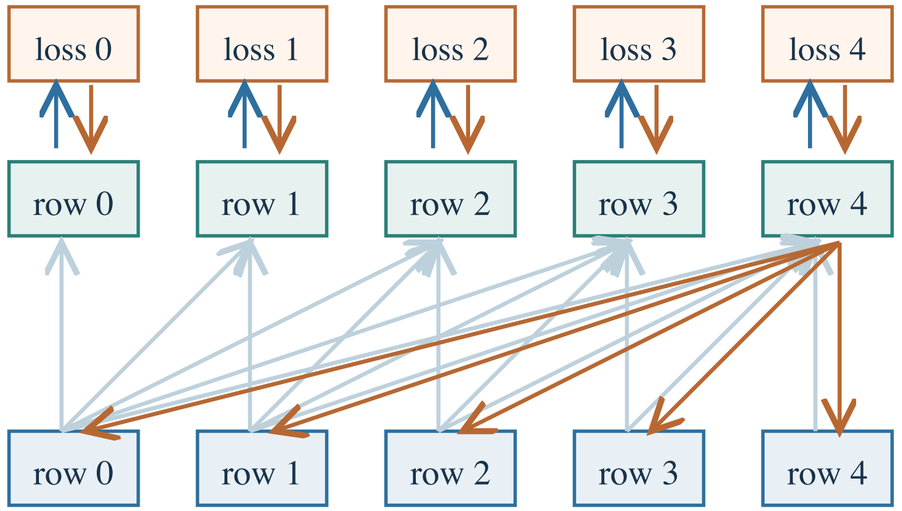
RNN: a sequential state chain



Forward: left → right; backward: right → left

Later states must wait for earlier states

Transformer: positions processed together



Shown: backward paths from the last row

Sequential across layers, not across token states

Blue: forward • Orange: backward • All losses contribute to shared parameters

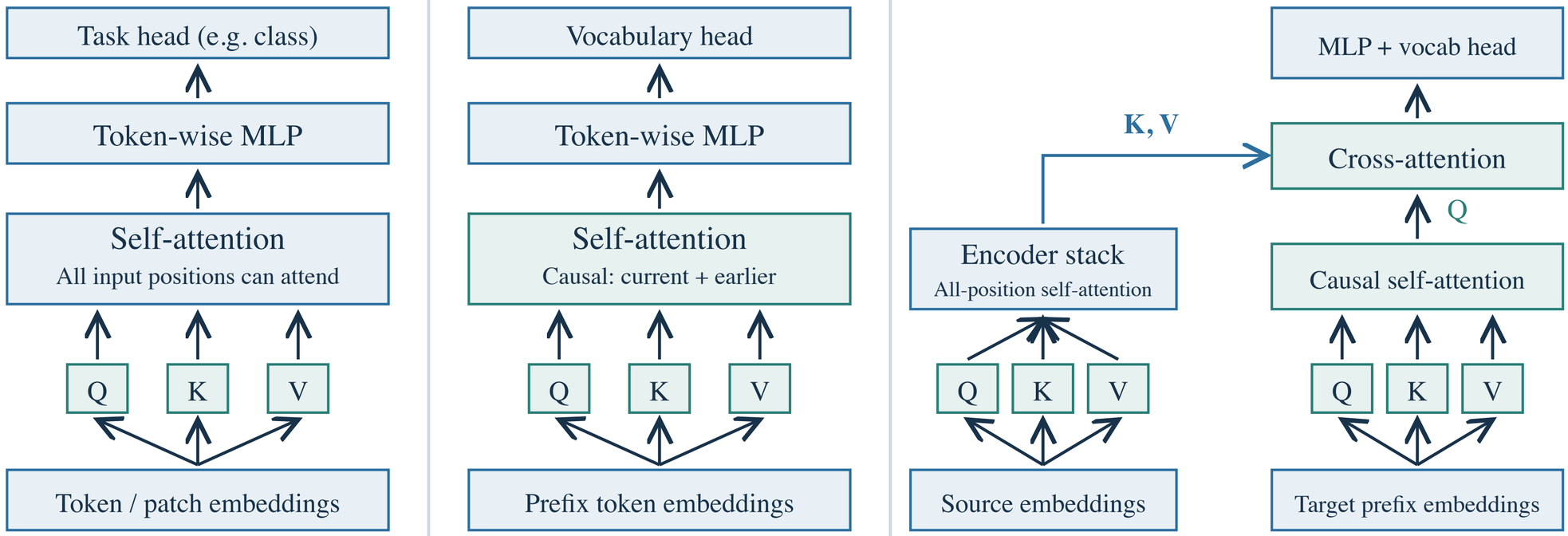
Transformer benefit: parallel work across positions in forward **and backward** computation.

Can one forward pass score all shifted targets?

> **The training batch contains each full email.**
> **Which model change would let an early prediction read its future target?**
> **A. Compute all causal output rows together**
> **B. Normalize each token over its own features**
> **C. Remove the causal attention mask**

C. Parallel computation is valid when every path respects the allowed prefix.

A causal decoder is the common decoder-only LLM architecture



Q/K/V are learned projections of the indicated stream. Layers repeat; residual paths and normalization omitted.

Our email model is a causal decoder. Other tasks can use different information paths and heads.