Nuts and Bolts in Training Transformers and Diffusion Models - Background Notes

Zihan Ding

April 2025

1 Transformers

1.1 Infini-attention

We follow the standard attention projection scheme with $k_t = \theta_K x_t$, $v_t = \theta_V x_t$, $q_t = \theta_Q x_t$ where $k_t, v_t, q_t \in \mathbb{R}^{d'}$.

The key components of Infini-attention are:

1. Memory Retrieval:

$$A_{\text{mem}} = \frac{\sigma(Q) M_{s-1}}{\sigma(Q) z_{s-1}}, \tag{1}$$

where $\sigma(\cdot)$ is $\text{ELU}(x) + 1$.

2. Memory Update (Delta variant):

$$M_s = M_{s-1} + \sigma(K)^\top \left(V - \frac{\sigma(K) M_{s-1}}{\sigma(K) z_{s-1}} \right), \tag{2}$$

$$z_s = z_{s-1} + \sum_{t=1}^{N} \sigma(k_t). \tag{3}$$

3. Output Aggregation:

$$A = \text{sigmoid}(\beta) \odot A_{\text{mem}} + (1 - \text{sigmoid}(\beta)) \odot A_{\text{dot}}. \tag{4}$$

The memory update equations are differentiable with respect to all parameters, allowing backpropagation through time (BPTT) to be applied:

1. The memory update equation is a linear combination of the previous memory state and an update term:

$$M_s = M_{s-1} + \sigma(K)^\top \Delta_s \tag{5}$$

where $\Delta_s = V - \frac{\sigma(K) M_{s-1}}{\sigma(K) z_{s-1}}$ is the update term.

2. The gradient can flow through:

- The linear combination operation
- The ELU activation $\sigma(\cdot)$ which is differentiable
- The division operation in Δ_s which is differentiable when the denominator is non-zero (guaranteed by $\sigma(K)z_{s-1} > 0$)

3. Similarly, the counter update $z_s = z_{s-1} + \sum_{t=1}^{N} \sigma(k_t)$ is a simple differentiable accumulation.

Therefore, gradients can be propagated backwards through the memory updates across multiple timesteps, allowing the model to learn long-term dependencies through the memory mechanism.

1.2 Normalization-Free Networks

Layer Normalization (LayerNorm) normalizes activations across feature dimensions for each sample independently:

$$\text{LayerNorm}(x) = \gamma \cdot \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{6}$$

where μ and σ^2 are the mean and variance computed across features.

Root Mean Square Normalization (RMSNorm) simplifies LayerNorm by only using the RMS statistic:

$$\text{RMSNorm}(x) = \frac{x}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2 + \epsilon}} \cdot \gamma, \tag{7}$$

where n is the number of features and γ is a learnable scale parameter.

Dynamic Tanh (DyT) is a novel activation function that introduces learnable parameters to adapt the shape and scale of the traditional hyperbolic tangent function. The key idea is to allow the network to learn optimal activation characteristics for different layers:

$$\text{DyT}(x) = \gamma \cdot \tanh(\alpha x) + \beta \tag{8}$$

where:

- α controls the steepness of the activation curve
- γ scales the output range
- β shifts the activation function vertically

These learnable parameters enable the network to:

- Adjust the sensitivity to input values through α

- Control the magnitude of activations via γ
- Optimize the bias through β

This adaptive activation function helps achieve better training stability and performance compared to fixed activation functions, while maintaining the bounded output properties of tanh.

2 Diffuison Models

2.1 Position Embedding

Rotary Position Embedding (RoPE) is a method for encoding positional information directly into the attention mechanism. It applies rotation transformations to query and key vectors based on their positions:

$$\text{RoPE}(x, m) = xe^{im\theta} \tag{9}$$

where x is the input vector, m is the position, θ is a frequency parameter. This formulation allows the model to naturally capture relative positional relationships while maintaining translational equivariance.

For a query q and key k at positions m and n respectively:

$$\begin{align} \langle \text{RoPE}(q, m), \text{RoPE}(k, n) \rangle &= \langle qe^{im\theta}, ke^{in\theta} \rangle \tag{10} \\ &= \text{RoPE}(\langle q, k \rangle, m - n) \tag{11} \end{align}$$

where $\text{RoPE}(x, p)$ applies the rotary transformation with position p. This formulation shows that:

- The attention score depends only on the relative position $(m - n)$
- The absolute positions are encoded but cancel out in the dot product

The choice of maximum frequency θ_{max} in RoPE affects the model's ability to distinguish positions at different scales:

- Lower θ_{max} allows better generalization to longer sequences but reduces positional precision
- Higher θ_{max} gives finer position discrimination but may not extrapolate well

The optimal choice depends on the expected sequence lengths and required positional resolution for the specific task.

Property	**Absolute PE**	**Sinusoidal PE**	**RoPE**
Formula	$x + p_m$ where p_m is learned	$x + [\sin(m\omega^k), \cos(m\omega^k)]$	$xe^{im\theta}$
Attention	$\langle x_m + p_m, x_n + p_n \rangle$	$\langle x_m + \text{PE}(m), x_n + \text{PE}(n) \rangle$	$\langle qe^{im\theta}, ke^{in\theta} \rangle = f(m-n)$
Relative Position	Not preserved	Implicit	Explicit via $m - n$
Where Applied	Input embeddings only	Input embeddings only	Every attention block

Table 1: Comparison of positional encoding methods. m, n are positions, d is dimension.

2.2 Classifier-Free Guidance

Classifier-free guidance (CFG) combines unconditional and conditional generation using a weighted average:

$$p_{CFG}(x_t|x_{t+1}, c) \propto p(x_t|x_{t+1}, c)^w \cdot p(x_t|x_{t+1})^{1-w} \tag{12}$$

This can be rewritten in terms of log probabilities, showing the connection to conditional probability:

$$\log p_{CFG}(x_t|x_{t+1}, c) = w \log p(x_t|x_{t+1}, c) + (1 - w) \log p(x_t|x_{t+1}) + const \tag{13}$$

where $w > 1$ increases the influence of the conditional signal c, while $w = 0$ recovers unconditional generation.

2.3 Flow Matching

Flow matching provides a more direct approach to learning continuous normalizing flows compared to diffusion models. The loss is simpler and requires only a single forward pass:

$$\mathcal{L}_{FM} = \mathbb{E}_{x_0,t,\epsilon} \left[\|\mathbf{v}(x_t, t) - \mathbf{v}_\theta(x_t, t)\|_2^2 \right] \tag{14}$$

where $x_t = x_0 + t(x_1 - x_0)$ is a point along the straight-line path between data x_0 and noise x_1, $\mathbf{v}(x_t, t)$ is the ground truth velocity field, and $\mathbf{v}_\theta(x_t, t)$ is the learned velocity field. This contrasts with diffusion models which require multiple denoising steps and typically use MSE loss on noise predictions.

2.4 FasterDiT

FasterDiT improve DiT efficiency by three key tricks: standard deviation scaling, log-time weighting and v-direction loss.

1. Extended SNR The traditional Signal-to-Noise Ratio (SNR) is defined as:

$$\mathrm{SNR}_{\mathrm{prev}}(t) = \frac{\alpha_t^2}{\sigma_t^2}$$

To account for data-dependent scaling, the authors extend this as:

$$\mathrm{SNR}_{\mathrm{adj}}(t) = C(I) \cdot \frac{\alpha_t^2}{\sigma_t^2}$$

2. Multi-Step Balance via Scaling and Logit-Normal Sampling To better shape the training SNR profile:

- **Data scaling:** Adjust the input signal variance to match desired SNR distribution.
- **Timestep sampling:** Use the logit-normal distribution:

  $$t \sim \mathrm{LogitNormal}(\mu, \sigma)$$

  to focus training on effective SNR ranges.

3. Velocity Direction Loss for Enhanced Supervision In addition to the velocity MSE loss:

$$\mathcal{L}_{\mathrm{MSE}} = \|v - \hat{v}\|^2$$

they introduce directional supervision:

$$\mathcal{L}_{\mathrm{dir}} = \lambda \cdot (1 - \cos(\theta)) = \lambda \cdot \left(1 - \frac{v \cdot \hat{v}}{\|v\|\|\hat{v}\|}\right)$$

where θ is the angle between v and $\hat{v}$. The total loss becomes:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{MSE}} + \mathcal{L}_{\mathrm{dir}}$$

2.5 Shortcut Model

Shortcut Models: A single neural network s_θ learns to denoise data in variable step sizes by conditioning on noise level and step size d.

The model builds on flow matching principles, where given data $x_0 \sim p_{\mathrm{data}}$ and noise $x_1 \sim \mathcal{N}(0, I)$, we have:

$$x_t = (1 - t)x_0 + tx_1$$

$$v_t = \frac{dx_t}{dt} = x_1 - x_0$$

$$v_\theta(x_t, t) \approx \mathbb{E}_{x_0, x_1}[x_1 - x_0 \mid x_t]$$

The key components of Shortcut model are:

1. A shortcut function that predicts updates between timesteps:

$$x_{t+d} = x_t + s_\theta(x_t, t, d) \tag{1}$$

2. A self-consistency property for efficient long-range predictions:

$$s_\theta(x_t, t, 2d) = \frac{1}{2}s_\theta(x_t, t, d) + \frac{1}{2}s_\theta(x_t + s_\theta(x_t, t, d), t, d) \tag{2}$$

3. A combined loss incorporating both flow matching and self-consistency:

$$\mathcal{L}^S(\theta) = \mathbb{E}_{x_0,x_1,t,d}\left[\|s_\theta(x_t, t, 0) - (x_1 - x_0)\|^2 + \|s_\theta(x_t, t, 2d) - s_{\text{target}}\|^2\right] \tag{3}$$

where:

$$s_{\text{target}} = \frac{1}{2}s_\theta(x_t, t, d) + \frac{1}{2}s_\theta(x'_{t+d}, t, d), \quad x'_{t+d} = x_t + s_\theta(x_t, t, d)$$

3 Tokenization

Image Encoding In diffusion transformers, images are first divided into non-overlapping patches and then tokenized. Given an input image $X \in \mathbb{R}^{H\times W\times C}$, where H, W are height and width, and C is the number of channels:

1. **Patchification:** The image is divided into N non-overlapping patches of size $P \times P$:

$$X_p \in \mathbb{R}^{N\times(P^2\cdot C)}, \quad \text{where } N = \frac{H \times W}{P^2}$$

2. **Linear Projection:** Each patch is flattened and projected to D dimensions using a learnable linear projection E:

$$Z = X_pE, \quad \text{where } E \in \mathbb{R}^{(P^2\cdot C)\times D}$$

3. **Position Embedding:** Learnable position embeddings $E_{pos} \in \mathbb{R}^{N\times D}$ are added to provide spatial information:

$$Z_{final} = Z + E_{pos}$$

The resulting tokens $Z_{final} \in \mathbb{R}^{N\times D}$ serve as input to the transformer architecture. During training, these tokens are conditioned on timesteps t and noise levels to learn the denoising process.

Text Encoding For text-guided image generation, the model also processes text inputs through a separate tokenization pipeline:

1. **Text Tokenization:** Input text is first tokenized using a pretrained tokenizer (e.g., CLIP, T5):

$$T = \text{Tokenizer}(text) \in \mathbb{R}^{L\times V}$$

where L is the sequence length and V is the vocabulary size.

2. **Text Embedding:** Tokens are embedded into a continuous space using pretrained text encoder:

$$T_e = \text{TextEncoder}(T) \in \mathbb{R}^{L \times D_t}$$

where D_t is the text embedding dimension.

D_t **does not necessarily to be the same as** D**.**

Cross-Attention for Image-Text Conditioning The image tokens Z_{final} and text embeddings T_e are combined through cross-attention layers in the transformer:

1. **Cross-Attention Mechanism:**

$$\text{CrossAttn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

where Q are queries from image tokens, and K, V are keys/values from text embeddings.

2. **Conditioning Integration:** Each transformer block includes cross-attention:

$$Z'_{final} = \text{CrossAttn}(Z_{final}W_Q, T_eW_K, T_eW_V)$$

where W_Q, W_K, W_V are learnable projection matrices.

This allows the image generation process to be guided by the semantic information in the text condition, enabling text-to-image generation capabilities.