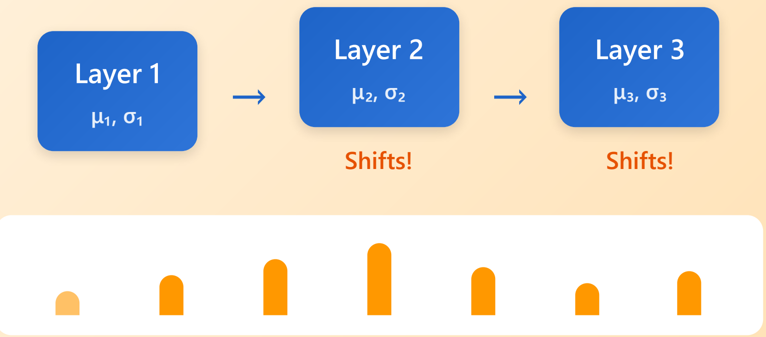
Internal Covariate Shift

Distribution Shifts During Training



Key Problems

- **Distribution changes** as previous layers update during training
- Forces layers to **continuously adapt** to new distributions
- **Slows down training** and reduces stability
- More **pronounced in deeper** networks

⚠ Training Impact

Requires lower learning rates for stable training

✓ Solution

Normalization techniques address this problem directly by stabilizing distributions

Mathematical Explanation

Input Distribution Changes per Layer

$$x^{(l)} = f(x^{(l-1)};\ \theta^{(l-1)})$$

Batch Normalization Formula

$$\mu_B = (1/m)\ \Sigma\ x_i$$

Input of layer l depends on parameters $\theta^{(l-1)}$ of previous layer

$$\theta^{(l-1)} \rightarrow \theta^{(l-1)} + \Delta\theta$$

When parameters update, input distribution $P(x^{(l)})$ changes

Quantifying Covariate Shift

$$D_{KL}(P_{old}(x) \;||\; P_{new}(x))$$

Measure shift magnitude using **KL-Divergence** between old and new distributions

Calculate batch mean

$$\sigma^2_B = (1/m)\ \Sigma\ (x_i - \mu_B)^2$$

Calculate batch variance

$$\hat{x_i} = (x_i - \mu_B)\ /\ \sqrt{(\sigma^2_B + \varepsilon)}$$

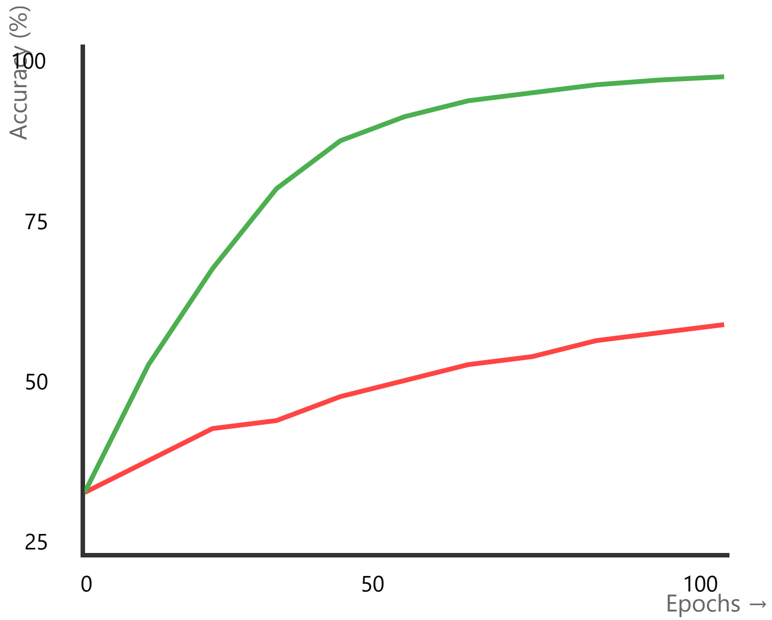
Normalized value (mean 0, variance 1)

$$y_i = \gamma\hat{x_i} + \beta$$

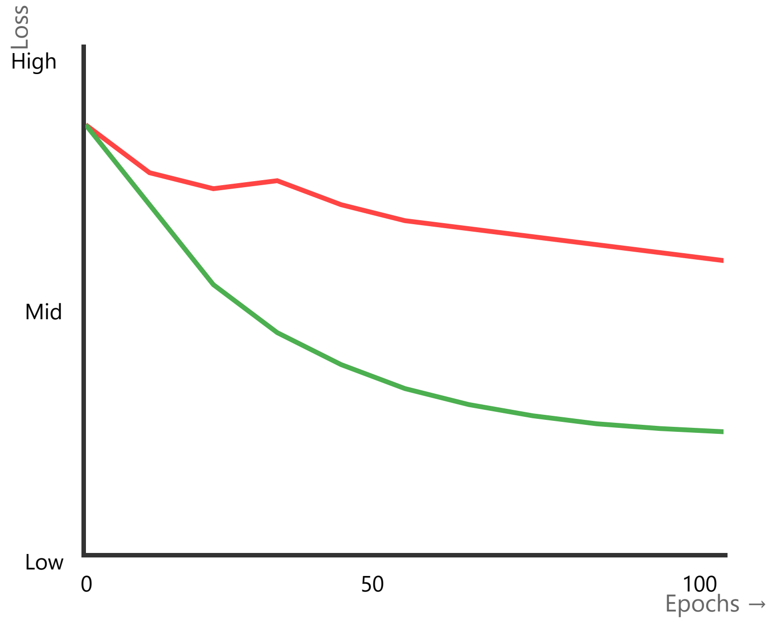
Scale and shift with learnable γ, β parameters

Experimental Results

Training Accuracy Over Epochs



Loss Over Epochs



Without Normalization
With Batch Normalization

Without Normalization
With Batch Normalization

Before vs After Comparison

❌ Without Normalization

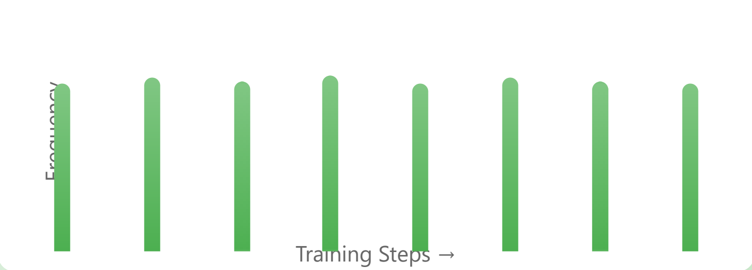
Learning Rate	0.001 (Very Low)
Convergence Time	~200 epochs
Training Stability	Unstable
Gradient Flow	Poor

Distribution Behavior

✓ With Batch Normalization

Learning Rate	0.01-0.1 (Higher)
Convergence Time	~50 epochs
Training Stability	Stable
Gradient Flow	Excellent

Distribution Behavior



💡 **Key Insight**

Batch Normalization allows **10-100x higher learning rates** and reduces training time by **~75%**