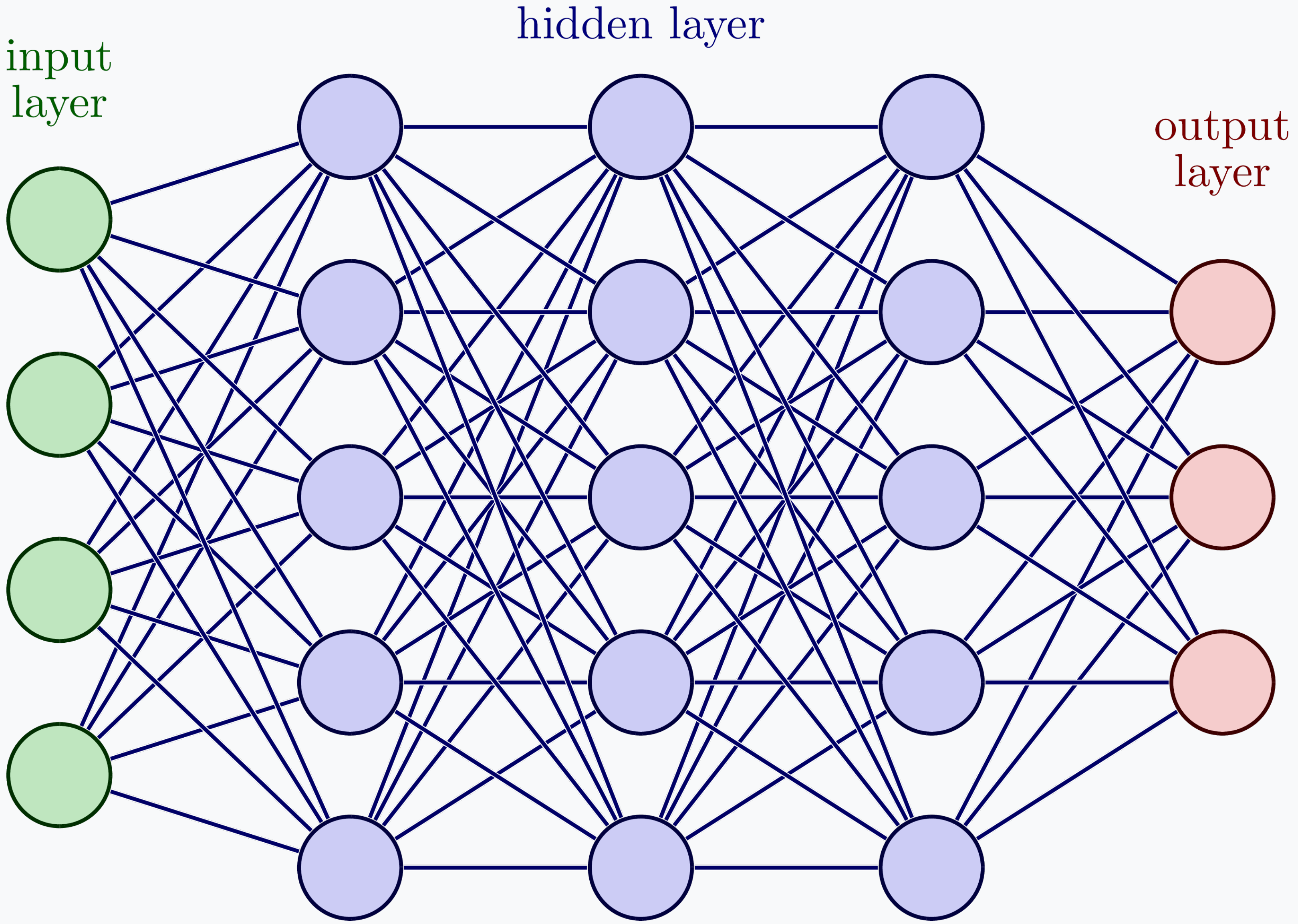
Neural Networks Class 4

Advanced Architectures & Regularization

Learning Objectives:

- Explore deep network architectures and residual connections
- Master advanced regularization techniques mathematically
- Understand various loss functions and their applications


input
layer
hidden layer
output
layer

Deep Networks - Depth vs Width

Classical Result (Width):

Single hidden layer with sufficient neurons can approximate any continuous function

Modern Understanding (Depth):

- **Exponential advantage:** Deep networks can represent certain functions exponentially more efficiently
- **Hierarchical features:** Each layer learns increasingly complex representations
- **Computational efficiency:** Depth reduces required parameters

Mathematical Insight:

Function with k layers and n neurons per layer: $O(kn^2)$ parameters
Same function with 1 layer: $O(n^k)$ parameters potentially needed

Residual Connections

The Vanishing Gradient Solution

Problem with Very Deep Networks:

$$\frac{\partial L}{\partial \mathbf{W}^{(1)}} = \frac{\partial L}{\partial \mathbf{A}^{(L)}} \prod_{l=2}^{L} \mathbf{W}^{(l)} \sigma'(\mathbf{Z}^{(l-1)})$$

As L increases, this product tends to vanish or explode.

Residual Block Mathematical Formulation:

$$\mathbf{A}^{(l+1)} = \sigma(\mathbf{A}^{(l)} + F(\mathbf{A}^{(l)}, \mathbf{W}^{(l)}))$$

Where $F(\mathbf{A}^{(l)}, \mathbf{W}^{(l)})$ is a residual function (typically 2-3 layers)

Key Innovation: Identity mapping $\mathbf{A}^{(l)}$ + learned residual $F(\cdot)$

Residual Connections - Gradient Flow Analysis

Forward Pass:

$$\mathbf{A}^{(l+1)} = \mathbf{A}^{(l)} + F(\mathbf{A}^{(l)}, \mathbf{W}^{(l)})$$

Backward Pass:

$$\frac{\partial L}{\partial \mathbf{A}^{(l)}} = \frac{\partial L}{\partial \mathbf{A}^{(l+1)}} \left(1 + \frac{\partial F(\mathbf{A}^{(l)}, \mathbf{W}^{(l)})}{\partial \mathbf{A}^{(l)}}\right)$$

Key Insight: The "+1" term ensures gradient flow even if $\frac{\partial F}{\partial \mathbf{A}^{(l)}} \to 0$

Multi-layer Gradient:

$$\frac{\partial L}{\partial \mathbf{A}^{(l)}} = \frac{\partial L}{\partial \mathbf{A}^{(L)}} \prod_{k=l+1}^{L} \left(1 + \frac{\partial F^{(k)}}{\partial \mathbf{A}^{(k-1)}}\right)$$

Skip Connections - Dense and Highway Networks

Dense Connections (DenseNet):

$$\mathbf{A}^{(l)} = \sigma([\mathbf{A}^{(0)}, \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(l-1)}]\mathbf{W}^{(l)} + \mathbf{b}^{(l)})$$

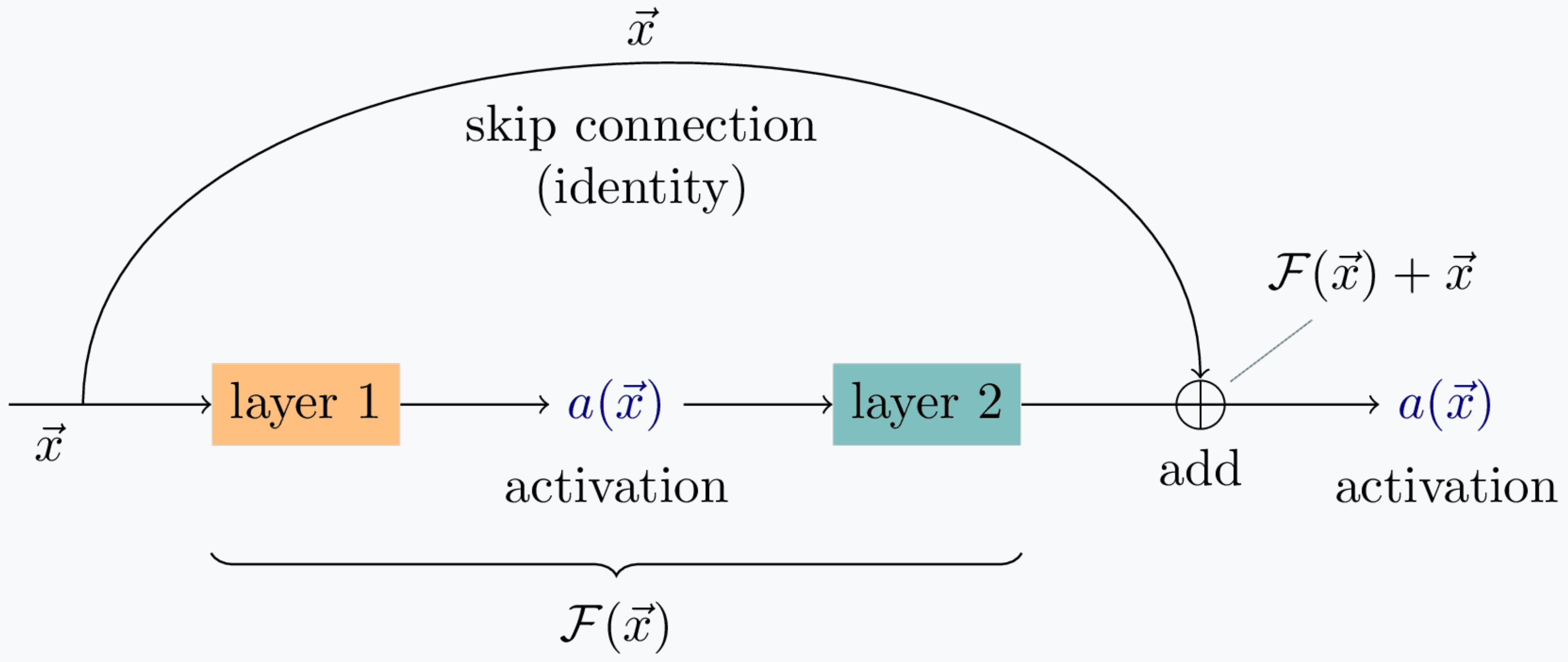
Where $[\cdot]$ denotes concatenation.

Highway Networks:

$$\mathbf{A}^{(l+1)} = \mathbf{T}(\mathbf{A}^{(l)}) \odot \sigma(\mathbf{W}_H^{(l)}\mathbf{A}^{(l)} + \mathbf{b}_H^{(l)}) + (1 - \mathbf{T}(\mathbf{A}^{(l)})) \odot \mathbf{A}^{(l)}$$

Where:

- $\mathbf{T}(\mathbf{A}^{(l)}) = \sigma(\mathbf{W}_T^{(l)}\mathbf{A}^{(l)} + \mathbf{b}_T^{(l)})$: Transform gate
- $1 - \mathbf{T}(\mathbf{A}^{(l)})$: Carry gate


$\vec{x}$
skip connection
(identity)
$\mathcal{F}(\vec{x}) + \vec{x}$
$\vec{x}$
layer 1
$a(\vec{x})$
activation
layer 2
add
$a(\vec{x})$
activation
$\mathcal{F}(\vec{x})$

Dropout Deep Dive - Mathematical Analysis

Training Phase: For layer l with dropout probability p:

$$\mathbf{r}^{(l)} \sim \text{Bernoulli}(1-p)$$

$$\tilde{\mathbf{A}}^{(l)} = \frac{1}{1-p}\mathbf{r}^{(l)} \odot \mathbf{A}^{(l)}$$

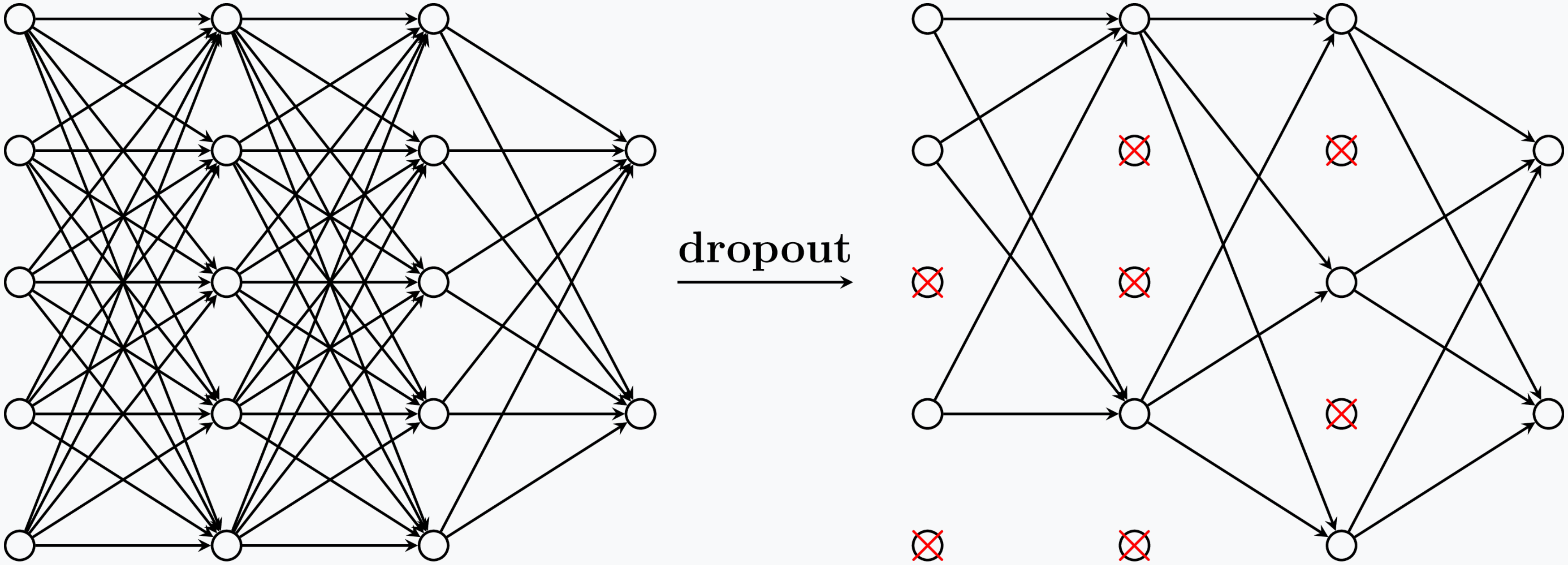
Expected Value Preservation:

$$\mathbb{E}[\tilde{A}_i^{(l)}] = \frac{1}{1-p} \cdot (1-p) \cdot A_i^{(l)} = A_i^{(l)}$$

Variance:

$$\text{Var}[\tilde{A}_i^{(l)}] = \frac{p}{1-p}(A_i^{(l)})^2$$

Inference Phase: No scaling needed due to expectation preservation


dropout

Dropout Variants

Spatial and Structured Dropout

Standard Dropout: Independent for each neuron

$$\mathbf{r}_i \sim \text{Bernoulli}(1-p) \text{ independently}$$

Spatial Dropout (for CNNs): Entire feature maps

$$\mathbf{r}_c \sim \text{Bernoulli}(1-p) \text{ for channel } c$$

$$\tilde{\mathbf{A}}_{i,j,c} = \frac{1}{1-p}\mathbf{r}_c \cdot \mathbf{A}_{i,j,c}$$

DropConnect: Drop connections instead of neurons

$$\mathbf{M}_{i,j} \sim \text{Bernoulli}(1-p)$$

$$\mathbf{A}^{(l+1)} = \sigma((\mathbf{M} \odot \mathbf{W}^{(l)})\mathbf{A}^{(l)} + \mathbf{b}^{(l)})$$

Early Stopping - Mathematical Formulation

Validation Loss Monitoring:

Let $L_{\text{val}}(t)$ be validation loss at epoch t

Stopping Criterion:

Stop training when:

$$L_{\text{val}}(t) > L_{\text{val}}(t-k) \text{ for } k \text{ consecutive epochs}$$

Patience Parameter: Number of epochs to wait before stopping

Mathematical Justification:

Minimizes expected test error by finding optimal bias-variance trade-off point

Optimal Stopping Time:

$$t^* = \arg\min_t \mathbb{E}[L_{\text{test}}(t)]$$

Data Augmentation - Mathematical Transformations

Geometric Transformations:

- **Rotation:** $\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
- **Translation:** $\mathbf{x}' = \mathbf{x} + \mathbf{t}$
- **Scaling:** $\mathbf{x}' = s \cdot \mathbf{x}$

Statistical Transformations:

- **Gaussian Noise:** $\mathbf{x}' = \mathbf{x} + \mathcal{N}(0, \sigma^2)$
- **Normalization:** $\mathbf{x}' = \frac{\mathbf{x}-\mu}{\sigma}$

Theoretical Foundation:

Augmentation increases effective dataset size and reduces overfitting by making model invariant to transformations

Loss Functions for Classification - Cross-Entropy

Binary Cross-Entropy: For probability $p = \sigma(\mathbf{w}^T\mathbf{x} + b)$ and true label $y \in \{0, 1\}$:

Likelihood:

$$P(y|\mathbf{x}) = p^y(1-p)^{1-y}$$

Log-Likelihood:

$$\log P(y|\mathbf{x}) = y \log p + (1-y)\log(1-p)$$

Cross-Entropy Loss:

$$L = -[y \log p + (1-y)\log(1-p)]$$

Derivative:

$$\frac{\partial L}{\partial p} = \frac{p-y}{p(1-p)}$$

Loss Functions for Classification - Multi-class CE

Softmax Function:

$$p_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

Properties:

- $\sum_{k=1}^{K} p_k = 1$, $p_k > 0$ for all k, Differentiable everywhere

Multi-class Cross-Entropy: $L = -\sum_{k=1}^{K} y_k \log p_k$

Derivative:

$$\frac{\partial L}{\partial z_k} = p_k - y_k$$

Loss Functions for Regression - Beyond MSE

Mean Squared Error (MSE):

$$L_{\text{MSE}} = \frac{1}{2}(\hat{y} - y)^2$$

- **Sensitive to outliers, Derivative:** $\hat{y} - y$

Mean Absolute Error (MAE):

$$L_{\text{MAE}} = |\hat{y} - y|$$

- **Robust to outliers, Non-differentiable at 0**

Huber Loss (Smooth MAE):

$$L_{\text{Huber}} = \begin{cases} \frac{1}{2}(\hat{y} - y)^2 & \text{if } |\hat{y} - y| \leq \delta \\ \delta(|\hat{y} - y| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}$$

Multi-task Learning - Mathematical Framework

Shared Representation: $\mathbf{h} = f_{\text{shared}}(\mathbf{x}; \boldsymbol{\theta}_{\text{shared}})$

Task-Specific Heads: $\hat{y}_i = f_i(\mathbf{h}; \boldsymbol{\theta}_i)$ for task i

Combined Loss: $L_{\text{total}} = \sum_{i=1}^{T} \lambda_i L_i(\hat{y}_i, y_i)$

Gradient Update:

$$\boldsymbol{\theta}_{\text{shared}} \leftarrow \boldsymbol{\theta}_{\text{shared}} - \eta \sum_{i=1}^{T} \lambda_i \frac{\partial L_i}{\partial \boldsymbol{\theta}_{\text{shared}}}$$

Challenge: Balancing task weights λ_i

Multi-label Classification - Mathematical Differences

Multi-class: One label per example, $\sum_k p_k = 1$

- Output layer: Softmax, Loss: Categorical cross-entropy

Multi-label: Multiple labels per example, Each p_k independent

- Output layer: Multiple sigmoids, Loss: Binary cross-entropy per label

Multi-label Loss: $L = -\sum_{k=1}^{K} [y_k \log p_k + (1 - y_k) \log(1 - p_k)]$

Threshold Decision:

$$\hat{y}_k = \begin{cases} 1 & \text{if } p_k > \tau \\ 0 & \text{otherwise} \end{cases}$$

Advanced Regularization

Spectral Normalization

Weight Matrix Singular Value Decomposition: $\mathbf{W} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$

Spectral Norm: Largest singular value

$$\sigma(\mathbf{W}) = \max_{\mathbf{h}:\|\mathbf{h}\|=1} \|\mathbf{W}\mathbf{h}\|$$

Spectral Normalization:

$$\mathbf{W}_{\text{SN}} = \frac{\mathbf{W}}{\sigma(\mathbf{W})}$$

Effect: Constrains Lipschitz constant of network

$$\|f(\mathbf{x}_1) - f(\mathbf{x}_2)\| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|$$

Layer Normalization

Batch Normalization: Normalize across batch dimension

$\hat{x}_i = \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}}$

Layer Normalization: Normalize across feature dimension

$\hat{x}_j = \frac{x_j - \mu_L}{\sqrt{\sigma_L^2 + \epsilon}}$

Where:

$$\mu_L = \frac{1}{H}\sum_{j=1}^{H} x_j, \quad \sigma_L^2 = \frac{1}{H}\sum_{j=1}^{H}(x_j - \mu_L)^2$$

Advantages:

- **No batch dependency:** Works with batch size = 1, **RNN friendly:** Better for sequential

Attention Mechanisms Preview

Attention Weights:

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{n} \exp(e_{i,k})}$$

Where $e_{i,j} = a(\mathbf{s}_i, \mathbf{h}_j)$ is attention energy

Context Vector: $\mathbf{c}_i = \sum_{j=1}^{n} \alpha_{i,j} \mathbf{h}_j$

Self-Attention:

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

Foundation for Transformers: Modern architecture revolutionizing NLP and beyond

Practical Architecture Design

Choosing Network Depth and Width

Depth Considerations: Shallow networks: Fast training, limited expressiveness, **Deep networks:** More expressive, harder to train, need residual connections

Width Considerations: Narrow networks: Few parameters, may underfit, **Wide networks:** Many parameters, may overfit, expensive

Rule of Thumb: Start with proven architectures (ResNet, DenseNet) and adapt

Architecture Search:

$$\alpha^* = \arg\min_{\alpha} \mathcal{L}_{\text{val}}(\mathbf{w}^*(\alpha), \alpha)$$

Where $\mathbf{w}^*(\alpha) = \arg\min_{\mathbf{w}} \mathcal{L}_{\text{train}}(\mathbf{w}, \alpha)$

Summary: Class 4 Key Concepts

Deep Network Architectures:

- **Residual connections:** $\mathbf{A}^{(l+1)} = \mathbf{A}^{(l)} + F(\mathbf{A}^{(l)})$ solve vanishing gradients
- **Skip connections:** Enable very deep networks, improve gradient flow

Advanced Regularization:

- **Dropout variants:** Spatial, DropConnect with mathematical formulations
- **Data augmentation:** Mathematical transformations for robustness
- **Spectral normalization:** Control Lipschitz constant

Loss Functions:

- **Classification:** Binary/multi-class cross-entropy derivations
- **Regression:** MSE, MAE, Huber loss trade-offs