Advanced Topics - Deep Learning Basics

Introduction to Deep Networks, Activation Functions, and Common Architectures

Sarwan Ali

Department of Computer Science
Georgia State University

Understanding Deep Learning

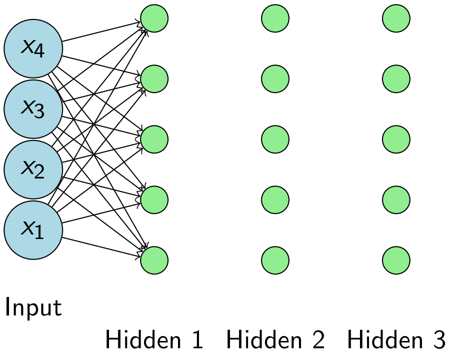
Today's Learning Journey

What is Deep Learning?

Deep Learning is a subset of machine learning that uses artificial neural networks with multiple layers (deep networks) to model and understand complex patterns in data.

Key Characteristics:

- Multiple hidden layers (typically 3+ layers)
- Automatic feature extraction
- Non-linear transformations
- End-to-end learning



Deep Learning vs Traditional ML

Traditional Machine Learning

- Manual feature engineering
- Shallow learning algorithms
- Limited representation power
- Good for structured data
- Interpretable models

Examples: Linear Regression, SVM, Decision Trees, Random Forest

Deep Learning

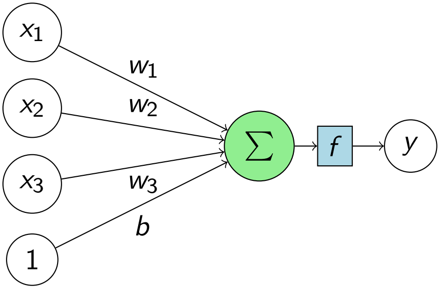
- Automatic feature learning
- Deep neural networks
- High representation power
- Excellent for unstructured data
- Black-box models

Examples: CNNs, RNNs, Transformers, GANs

The Perceptron: Building Block

Single Perceptron:

$$
\begin{aligned}
y &= f\left(\sum_{i=1}^{n} w_i x_i + b\right) && (1)\\
&= f(\mathbf{w}^T\mathbf{x} + b) && (2)
\end{aligned}
$$



Where:

- $\mathbf{x}$: input vector
- $\mathbf{w}$: weight vector
- b: bias term
- f: activation function

Multi-Layer Perceptron (MLP)

Forward Propagation:

$$\mathbf{h}^{(1)} = f^{(1)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) \tag{3}$$

$$\mathbf{h}^{(2)} = f^{(2)}(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}) \tag{4}$$

$$\vdots \tag{5}$$

$$\mathbf{y} = f^{(L)}(\mathbf{W}^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}) \tag{6}$$

Where:

- $\mathbf{h}^{(l)}$: hidden layer l activations
- $\mathbf{W}^{(l)}$: weight matrix for layer l
- $\mathbf{b}^{(l)}$: bias vector for layer l
- $f^{(l)}$: activation function for layer l
- L: total number of layers

Why Activation Functions?

Purpose of Activation Functions:

- Introduce **non-linearity** into the network
- Enable learning of complex patterns
- Without activation functions, deep networks collapse to linear models

Mathematical Insight:

$$\begin{aligned}
\text{Without activation: } \mathbf{y} &= \mathbf{W}^{(2)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)} && (7)\\
&= \mathbf{W}^{(2)}\mathbf{W}^{(1)}\mathbf{x} + \mathbf{W}^{(2)}\mathbf{b}^{(1)} + \mathbf{b}^{(2)} && (8)\\
&= \mathbf{W}'\mathbf{x} + \mathbf{b}' \quad \text{(Linear transformation)} && (9)
\end{aligned}$$

Common Activation Functions

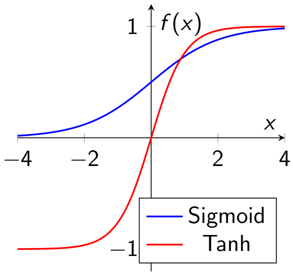
1. Sigmoid Function $\sigma(x) = \frac{1}{1+e^{-x}}$

Properties:

- Range: $(0, 1)$
- Smooth, differentiable
- Prone to vanishing gradients

2. Hyperbolic Tangent (Tanh)

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{10}$$



Properties:

- Range: $(-1, 1)$
- Zero-centered
- Still suffers from vanishing gradients

3. ReLU (Rectified Linear Unit)

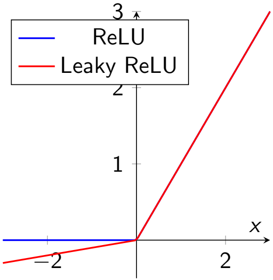
$$\mathrm{ReLU}(x) = \max(0, x) \tag{11}$$

Advantages:

- Simple computation
- No vanishing gradient problem
- Sparse activation

Disadvantages:

- Dying ReLU problem
- Not differentiable at $x = 0$



4. Leaky ReLU

$$\mathrm{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases} \tag{12}$$

5. Swish/SiLU (Sigmoid Linear Unit)

$$\text{Swish}(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}} \tag{13}$$

6. GELU (Gaussian Error Linear Unit)

$$\text{GELU}(x) = x \cdot \Phi(x) = \frac{x}{2}[1 + \text{erf}(\frac{x}{\sqrt{2}})] \tag{14}$$

Choosing Activation Functions:

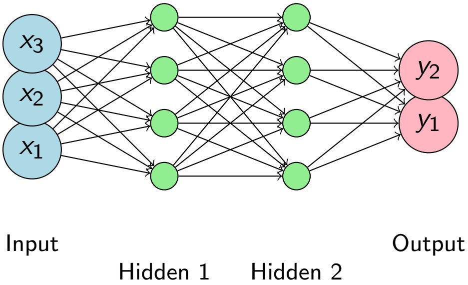
- **Hidden layers:** ReLU (default choice), Swish, GELU
- **Output layer:**
  - Binary classification: Sigmoid
  - Multi-class classification: Softmax
  - Regression: Linear (no activation)

Feedforward Neural Networks

Architecture: Information flows in one direction from input to output

Characteristics:

- Fully connected layers
- No cycles or loops
- Each neuron connects to all neurons in next layer
- Universal function approximators



Applications:

- Classification tasks
- Regression problems
- Feature learning
- Function approximation

Convolutional Neural Networks (CNNs)

Designed for processing grid-like data (images, time series)

Key Components:

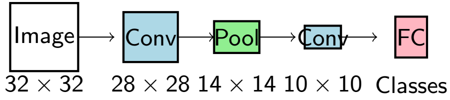
- **Convolutional layers:** Feature extraction
- **Pooling layers:** Dimensionality reduction
- **Fully connected layers:** Classification

Advantages:

- Translation invariance
- Parameter sharing
- Local connectivity
- Hierarchical feature learning

Applications:

- Image classification, Object detection
- Medical imaging, Computer vision



Convolution Operation:

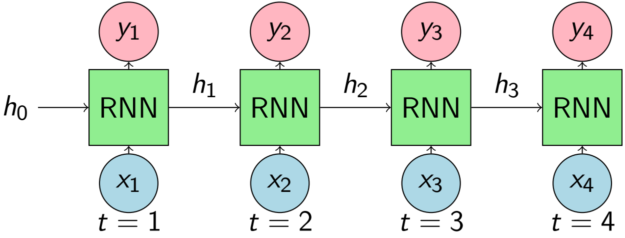
$$(f * g)(i, j) = \sum_m \sum_n f(m, n) \cdot g(i - m, j - n) \tag{15}$$

Recurrent Neural Networks (RNNs)

Designed for sequential data processing

Key Characteristics:

- Memory through hidden states
- Process sequences of variable length
- Share parameters across time steps
- Can model temporal dependencies



RNN Equations:

$$\mathbf{h}_t = f(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h) \tag{16}$$

$$\mathbf{y}_t = \mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y \tag{17}$$

Variants:

- **LSTM:** Long Short-Term Memory
- **GRU:** Gated Recurrent Unit

Applications:

- Natural Language Processing
- Time series forecasting
- Speech recognition
- Machine translation

Backpropagation Algorithm

Core idea: Compute gradients of loss function with respect to network parameters using chain rule

Forward Pass: Compute predictions

$$\mathbf{a}^{(l)} = f^{(l)}(\mathbf{z}^{(l)}) \quad \text{where} \quad \mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \tag{18}$$

Backward Pass: Compute gradients

$$\delta^{(l)} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{(l)}} \tag{19}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \delta^{(l)}(\mathbf{a}^{(l-1)})^T \tag{20}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \delta^{(l)} \tag{21}$$

$$\delta^{(l-1)} = (\mathbf{W}^{(l)})^T\delta^{(l)} \odot f'^{(l-1)}(\mathbf{z}^{(l-1)}) \tag{22}$$

Where $\mathcal{L}$ is the loss function and $\odot$ denotes element-wise multiplication.

Gradient Descent Optimization

Parameter Update Rule:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta \mathcal{L}(\theta_t) \tag{23}$$

Where η is the learning rate and θ represents all network parameters.

Variants:

- **Batch Gradient Descent:** Uses entire dataset
- **Stochastic Gradient Descent (SGD):** Uses single examples
- **Mini-batch Gradient Descent:** Uses small batches

Advanced Optimizers:

- **Adam:** Adaptive moment estimation
- **RMSprop:** Root mean square propagation
- **AdaGrad:** Adaptive gradient algorithm

Vanishing and Exploding Gradients

Vanishing Gradients:

- Gradients become very small in early layers
- Common with sigmoid/tanh activations
- Deep networks fail to learn

Solutions:

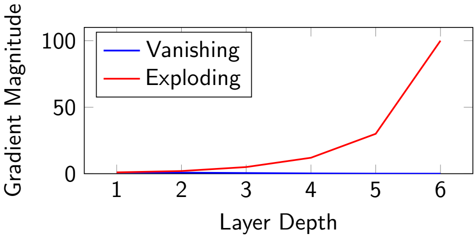
- Use ReLU activations
- Proper weight initialization
- Batch normalization
- Residual connections

Exploding Gradients:

- Gradients become very large
- Unstable training
- Parameters oscillate wildly

Solutions:

- Gradient clipping
- Proper weight initialization
- Lower learning rates
- Batch normalization



Overfitting in Deep Networks

Overfitting: Model performs well on training data but poorly on unseen data

Causes:

- Too many parameters
- Insufficient training data
- Complex model architecture
- Training for too long

Detection:

- Large gap between training and validation loss
- Validation accuracy decreases while training accuracy increases

Regularization Techniques:

- **Dropout:** Randomly set neurons to zero
- **L1/L2 Regularization:** Add penalty terms
- **Early Stopping:** Stop when validation loss increases
- **Data Augmentation:** Increase training data
- **Batch Normalization:** Normalize layer inputs

Dropout Mathematical Formulation:

$$\tilde{\mathbf{h}} = \mathbf{r} \odot \mathbf{h} \quad \text{where} \quad r_i \sim \text{Bernoulli}(p) \tag{24}$$

During training: p is dropout probability. During inference: scale by $(1 - p)$.

Batch Normalization

Problem: Internal covariate shift - distribution of layer inputs changes during training

Batch Normalization Algorithm:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{(Batch mean)} \tag{25}$$

$$\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \quad \text{(Batch variance)} \tag{26}$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad \text{(Normalize)} \tag{27}$$

$$y_i = \gamma \hat{x}_i + \beta \quad \text{(Scale and shift)} \tag{28}$$

Where γ and β are learnable parameters, ϵ is a small constant for numerical stability.

Benefits:

- Faster training convergence, Higher learning rates possible
- Reduces internal covariate shift, Acts as regularization

Skip Connections and ResNet

Residual Learning: Learn residual mapping instead of direct mapping

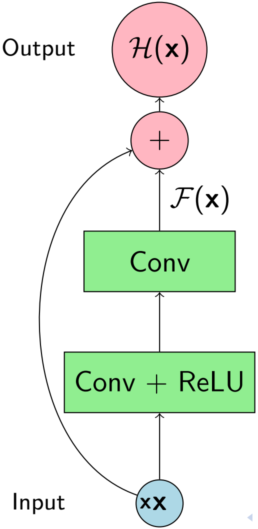
Traditional Network:

$$\mathcal{H}(\mathbf{x}) = \text{desired mapping} \tag{29}$$

Residual Network:

$$\mathcal{H}(\mathbf{x}) = \mathcal{F}(\mathbf{x}) + \mathbf{x} \tag{30}$$

$$\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x} \tag{31}$$



Advantages:

- Solves vanishing gradient problem
- Enables very deep networks (100+ layers)
- Identity mapping when $\mathcal{F}(\mathbf{x}) = 0$
- Improved gradient flow

Attention Mechanisms

Motivation: Focus on relevant parts of input sequence

Attention Formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{32}$$

Where:

- Q: Query matrix
- K: Key matrix
- V: Value matrix
- d_k: Dimension of key vectors

Self-Attention: Q, K, and V are all derived from the same input sequence

Applications:

- **Transformer Architecture:** GPT, BERT, T5
- **Computer Vision:** Vision Transformers (ViTs)
- **Multimodal:** CLIP, DALL-E

Weight Initialization

Why Important: Poor initialization can lead to vanishing/exploding gradients

Common Initialization Methods:

- **Xavier/Glorot Initialization:**

$$W \sim \mathcal{N}\left(0, \frac{2}{n_{in} + n_{out}}\right) \tag{33}$$

- **He Initialization (for ReLU):**

$$W \sim \mathcal{N}\left(0, \frac{2}{n_{in}}\right) \tag{34}$$

- **LeCun Initialization:**

$$W \sim \mathcal{N}\left(0, \frac{1}{n_{in}}\right) \tag{35}$$

Where n_{in} is number of input units and n_{out} is number of output units.

Rule of Thumb: Use He initialization for ReLU networks, Xavier for sigmoid/tanh networks.

Hyperparameter Tuning

Key Hyperparameters in Deep Learning:

Architecture:

- Number of layers
- Number of neurons per layer
- Activation functions
- Network topology

Training:

- Learning rate
- Batch size
- Number of epochs
- Optimizer choice

Regularization:

- Dropout rate
- L1/L2 regularization strength
- Early stopping patience
- Data augmentation parameters

Tuning Strategies:

- Grid search
- Random search
- Bayesian optimization
- Automated ML (AutoML)

Best Practice: Start with established architectures and fine-tune from there.

Model Evaluation and Validation

Evaluation Strategies:

- **Train/Validation/Test Split:** 60
- **Cross-Validation:** K-fold cross-validation
- **Hold-out Validation:** Simple train/test split

Metrics for Different Tasks:

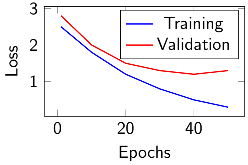
Classification:

- Accuracy
- Precision, Recall, F1-score
- ROC-AUC
- Confusion Matrix

Regression:

- Mean Squared Error (MSE)
- Mean Absolute Error (MAE)
- R-squared (R^2)
- Root Mean Squared Error (RMSE)

Learning Curves:



Real-World Applications

Computer Vision:

- Image classification
- Object detection
- Facial recognition
- Medical imaging
- Autonomous driving

Natural Language Processing:

- Machine translation
- Sentiment analysis
- Chatbots and virtual assistants
- Text summarization
- Question answering

Other Domains:

- **Speech:** Recognition, synthesis
- **Robotics:** Control, navigation
- **Gaming:** AlphaGo, game AI
- **Finance:** Fraud detection, trading
- **Healthcare:** Drug discovery, diagnosis
- **Art:** Style transfer, generation

Emerging Applications:

- Climate modeling
- Protein folding
- Scientific discovery
- Creative content generation

Current Trends and Future Directions

Current Hot Topics:

Architecture Innovations:

- **Transformers:** Attention is all you need
- **Vision Transformers:** ViT, DeiT, Swin
- **Efficient Architectures:** MobileNet, EfficientNet
- **Neural Architecture Search:** AutoML for architectures

Training Innovations:

- Self-supervised learning
- Few-shot learning
- Meta-learning
- Continual learning

Large-Scale Models:

- Large Language Models (GPT, BERT)
- Foundation models
- Multimodal models
- Scaling laws

Challenges and Opportunities:

- Interpretability and explainability
- Robustness and adversarial attacks
- Energy efficiency
- Fairness and bias
- Democratization of AI

Key Concepts Review

What We Covered Today:

1. **Deep Learning Fundamentals:**
   - Multi-layer neural networks
   - Forward and backward propagation
   - Universal function approximation
2. **Activation Functions:**
   - ReLU family (ReLU, Leaky ReLU)
   - Traditional functions (Sigmoid, Tanh)
   - Modern variants (Swish, GELU)
3. **Common Architectures:**
   - Feedforward Networks (MLPs)
   - Convolutional Neural Networks (CNNs)
   - Recurrent Neural Networks (RNNs)
4. **Training Challenges and Solutions:**
   - Vanishing/exploding gradients
   - Overfitting and regularization
   - Modern techniques (BatchNorm, ResNet, Attention)

Best Practices Summary

Architecture Design:

- Start simple, then add complexity
- Use proven architectures as baselines
- Consider computational constraints
- Match architecture to problem type

Training Strategy:

- Use appropriate initialization
- Monitor training/validation curves
- Apply regularization techniques
- Use modern optimizers (Adam, AdamW)

Debugging and Optimization:

- Overfit on small dataset first
- Check gradient magnitudes
- Visualize learned features
- Use learning rate schedules

Evaluation:

- Use proper train/validation/test splits
- Report multiple metrics
- Consider domain-specific evaluation
- Test on diverse datasets

Next Steps in Your Deep Learning Journey

Immediate Next Steps:

- Implement basic neural networks from scratch
- Experiment with different activation functions
- Try various optimization algorithms
- Practice with real datasets

Advanced Topics to Explore:

- **Specialized Architectures:** Transformers, GANs, VAEs
- **Advanced Training:** Transfer learning, multi-task learning
- **Optimization:** Learning rate scheduling, gradient clipping
- **Deployment:** Model compression, quantization, edge deployment

Recommended Resources:

- Deep Learning by Ian Goodfellow, Yoshua Bengio, Aaron Courville
- Papers With Code (paperswithcode.com)
- PyTorch/TensorFlow tutorials
- Coursera Deep Learning Specialization

Thank You!

Questions and Discussion

Remember: Deep learning is both an art and a science.
Practice, experiment, and stay curious!