MIDDLE EAST TECHNICAL UNIVERSITY

SEMESTER I EXAMINATION 2024-2025

CENG 403 – Deep Learning - CNN Architectures & RNN Introduction - ANSWERED

January 2025 TIME ALLOWED: 3 HOURS

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **SIX (6)** questions and comprises **EIGHT (8)** printed pages.
2. Answer all questions. The marks for each question are indicated at the beginning of each question.
3. Answer each question beginning on a **FRESH** page of the answer book.
4. This **IS NOT an OPEN BOOK** exam.
5. Show clear reasoning for your answers, especially intuitive explanations.
6. For architectural diagrams, draw clear components and explain design choices.
7. Connect concepts to examples discussed in lectures where relevant.
8. Explain the practical implications of design decisions.

Question 1. ResNet and Skip Connections (25 marks)

Based on the professor's explanation: "The identity function doesn't have to be learned as part of the solution, we are giving identity as part of the solution."

(a) The professor mentioned that ResNet researchers "noticed that up to a certain number of layers actually performance degrades." Explain why this was counterintuitive and how skip connections solve this problem. Include the professor's explanation of why networks have difficulty learning the identity function. (8 marks)

Answer: The degradation problem was counterintuitive because adding more layers should theoretically allow networks to learn at least as good a representation as the shallower network by learning identity mappings in the additional layers.

The professor explained this counterintuitive finding in detail:

Why degradation was unexpected:

- Deeper networks should have at least the same representational capacity as shallower ones
- If additional layers learn identity functions, performance should not degrade
- The issue wasn't overfitting (since both training and test error increased)

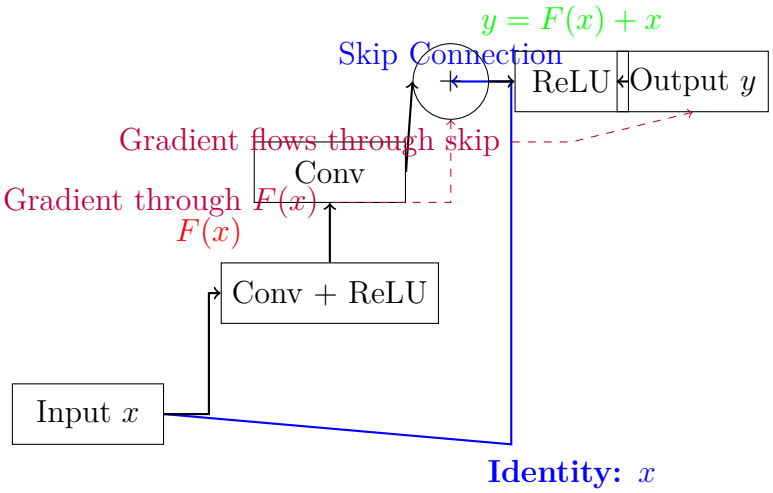
The core problem: Networks have difficulty learning the identity function because:

- Identity mapping $H(x) = x$ seems simple but is actually hard to learn through composition of ReLU and linear layers
- Multiple non-linear layers struggle to approximate the identity transformation
- Gradient vanishing makes it difficult for deep layers to receive useful learning signals

How skip connections solve this:

- Instead of learning $H(x)$, the network learns residual $F(x) = H(x) - x$
- Then $H(x) = F(x) + x$, where identity is built into the architecture
- If the optimal function is close to identity, $F(x)$ approaches zero, which is easier to learn
- The identity path is always preserved, providing a baseline performance guarantee

(b) Draw a ResNet block as described by the professor, showing how "in forward pass the identity can be implemented as part of the solution" and "in backward pass through these residual connections gradient can flow without any vanishing issues." (10 marks)



Forward Pass (Identity Implementation):

- Input x splits into two paths: identity path and weight layers path
- Weight layers compute $F(x)$ (the residual function)
- Identity path preserves original input x unchanged
- Output: $y = F(x) + x$ combines both paths

Backward Pass (Gradient Flow):

- Gradient $\frac{\partial L}{\partial y}$ reaches the addition node
- Splits into two paths: $\frac{\partial L}{\partial F(x)}$ and $\frac{\partial L}{\partial x}$
- Identity path: $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y} \cdot 1 = \frac{\partial L}{\partial y}$
- This ensures at least one gradient path is never vanishing (always has magnitude 1)
- Even if $F(x)$ gradients vanish, identity gradients flow unimpeded

(c) The professor explained that ResNet "smooths the loss surface" and provides "robustness to slight changes in the weights." Explain this concept using the professor's reasoning about how identity transform provides robustness when weights change slightly. (7 marks)

Answer: The identity transform in ResNet provides inherent robustness because even if the learned weights $F(x)$ become suboptimal due to slight changes, the identity path x always preserves the original input, creating a safety net that smooths the loss landscape.

Loss Surface Smoothing:

- Traditional deep networks have sharp, non-convex loss surfaces
- Small weight changes can lead to large performance drops
- ResNet's identity connections create "highways" in the loss landscape
- These highways provide consistent paths even when weights change

Robustness Mechanism:

- If weights in $F(x)$ become slightly perturbed: $F'(x) = F(x) + \delta F(x)$
- Output becomes: $y' = F'(x) + x = F(x) + \delta F(x) + x$
- The identity term x remains unchanged, providing stability
- Even if $\delta F(x)$ is large, the identity ensures reasonable output

Mathematical Insight:

- The Jacobian $\frac{\partial y}{\partial x} = \frac{\partial F(x)}{\partial x} + I$ always has identity component
- This prevents the Jacobian from becoming arbitrarily small or large

- Results in more stable training dynamics and better generalization
- Creates multiple viable paths through the network, increasing robustness

Question 2. ResNeXt and Multiple Pathways (20 marks)
The professor described ResNeXt as extending ResNet "by utilizing multiple paths acting on the same input layer."

(a) Compare ResNeXt with the inception module from GoogleNet as the professor discussed. Explain the key difference: "in inception module the filter sizes are different in different paths, whereas here in ResNeXt the sizes are the same." (8 marks)

Answer: ResNeXt and Inception modules both use multiple parallel paths, but ResNeXt uses identical transformations across all paths while Inception uses different filter sizes to capture multi-scale information.

Inception Module Approach:

- Multiple paths with different receptive field sizes: 1×1, 3×3, 5×5 convolutions
- Each path captures information at different scales
- Designed to handle the "what size should my filter be?" problem
- Concatenates outputs from different-sized filters

ResNeXt Approach:

- Multiple paths with identical transformations: same filter sizes across all paths
- Each path learns different feature representations using same architecture
- Focuses on "how many transformations should I apply?" rather than "what size?"
- Aggregates outputs from same-sized but differently-trained filters

Key Philosophical Difference:

- Inception: *Diversity through different scales*
- ResNeXt: *Diversity through multiple identical transformations*
- Inception targets multi-scale feature extraction
- ResNeXt targets ensemble-like behavior within a single block

(b) The professor noted that ResNeXt "can provide better performance than ResNet however because of the multiple paths it is slower." Explain this trade-off and when you would choose ResNeXt over ResNet according to the professor's guidance. (7 marks)

Answer: ResNeXt provides better accuracy through ensemble-like multiple pathways but requires more computation. Choose ResNeXt when accuracy is critical and computational resources are available; choose ResNet when speed and efficiency are priorities.

Performance Advantage:

- Multiple paths create ensemble-like behavior within single blocks
- Each path can specialize in different aspects of feature learning
- Aggregation of multiple transformations reduces variance
- Better generalization through implicit regularization

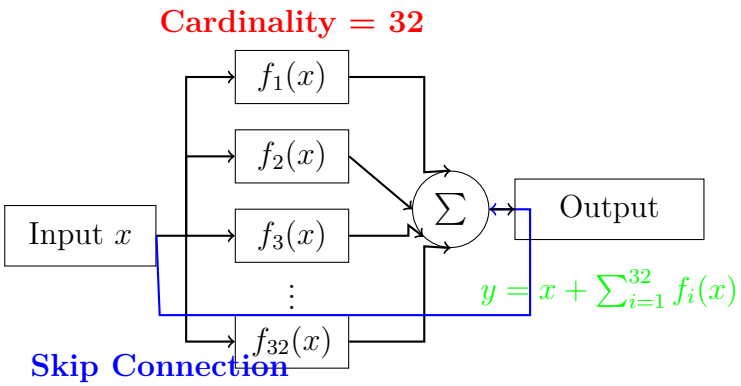
Computational Cost:

- ResNeXt requires computing multiple parallel transformations
- Memory usage increases proportional to cardinality (number of paths)
- Cannot fully utilize parallel computing due to aggregation step
- Approximately C times slower than ResNet (where C = cardinality)

When to Choose Each:

- **Choose ResNeXt:** High-accuracy applications (medical imaging, autonomous vehicles), sufficient computational budget, offline processing
- **Choose ResNet:** Real-time applications, mobile/edge devices, limited computational resources, online inference

(c) Draw a ResNeXt block showing multiple functions $f_1(x), f_2(x), \ldots, f_{32}(x)$ acting on the same input as the professor illustrated, resulting in $x + f_1(x) + f_2(x) + \ldots + f_{32}(x)$. (5 marks)



ResNeXt Block Structure:

- Input x is fed to 32 identical transformation functions
- Each $f_i(x)$ has same architecture but different learned parameters
- All transformations are computed in parallel
- Outputs are aggregated through element-wise addition
- Skip connection adds original input to aggregated result
- Final output: $y = x + \sum_{i=1}^{32} f_i(x)$

Question 3. DenseNet Architecture (22 marks)

Based on the professor's explanation: "We have residual connections not only skipping the next block but skipping to and making connections to all following blocks."

(a) Explain DenseNet's approach to skip connections as described by the professor. Why does having "skip connections for all following layers" make it "a very dense network" and potentially "more difficult to implement"? (8 marks)

Answer: DenseNet creates skip connections from each layer to ALL subsequent layers, creating a dense connectivity pattern where layer ℓ receives feature maps from all preceding layers $0, 1, 2, \ldots, \ell - 1$, making implementation complex due to the quadratic growth in connections.

DenseNet Connectivity Pattern:

- Layer ℓ receives inputs from layers $0, 1, 2, \ldots, \ell - 1$
- Each layer's input: $x_\ell = H_\ell([x_0, x_1, \ldots, x_{l-1}])$
- $[x_0, x_1, \ldots, x_{l-1}]$ represents concatenation of all previous feature maps
- Creates $\frac{L(L+1)}{2}$ connections for L layers (quadratic growth)

Why it's "Very Dense":

- Every layer connects to every other layer in a feed-forward fashion
- Information flows directly from any layer to all subsequent layers
- Creates maximum information flow density possible
- Results in rich feature reuse and gradient flow

Implementation Difficulties:

- Memory complexity: Each layer must store and access all previous feature maps
- Concatenation operations become increasingly expensive
- GPU memory management becomes challenging

- Requires careful memory optimization to avoid out-of-memory errors
- Debugging and visualization become complex due to dense connections

(b) The professor mentioned that DenseNet "can provide better performance compared to ResNet" but wasn't sure "why it didn't become as popular as ResNet." Analyze the potential reasons for this based on implementation complexity and computational requirements. (8 marks)

Answer: Despite better performance, DenseNet's popularity was limited by implementation complexity, memory requirements, computational overhead of concatenation operations, and the practical simplicity advantage of ResNet's architecture.

Performance Advantages of DenseNet:

- Better parameter efficiency (fewer parameters for same performance)
- Stronger gradient flow due to direct connections
- Implicit regularization through feature reuse
- Better feature propagation and reuse

Reasons for Limited Adoption:

1. Memory Constraints:

- Memory usage grows quadratically with depth
- GPU memory limitations restrict practical network sizes
- Batch size limitations reduce training efficiency

2. Implementation Complexity:

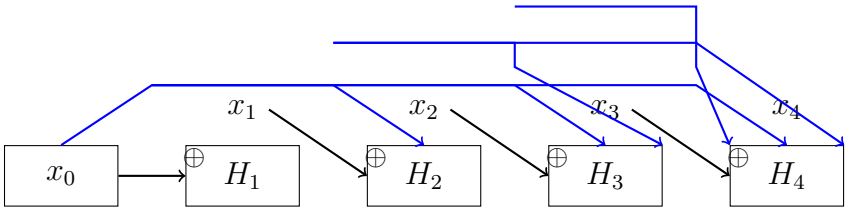
- Requires sophisticated memory management
- Concatenation operations are computationally expensive
- More complex to implement efficiently in frameworks
- Harder to optimize for different hardware

3. Practical Considerations:

- ResNet's simplicity makes it easier to modify and extend
- ResNet has cleaner mathematical formulation
- Industry adoption favored simpler, more reliable architectures
- Transfer learning and fine-tuning are simpler with ResNet

(c) Draw a DenseNet block showing how "from this layer we have skip connections to all following layers" as the professor described. Show at least 4 layers with their dense connections. (6 marks)

Dense Skip Connections



$$x_\ell = H_\ell([x_0, x_1, \ldots, x_{\ell-1}])$$

Dense Connection Pattern:

- $x_1 = H_1(x_0)$ - Layer 1 receives only x_0
- $x_2 = H_2([x_0, x_1])$ - Layer 2 receives concatenation of x_0 and x_1
- $x_3 = H_3([x_0, x_1, x_2])$ - Layer 3 receives all previous feature maps
- $x_4 = H_4([x_0, x_1, x_2, x_3])$ - Layer 4 receives all previous feature maps

Key Features:

- Blue arrows show skip connections bypassing multiple layers
- $\oplus$ symbols represent concatenation operations
- Each layer's input grows with network depth
- Maximum information reuse and gradient flow

Question 4. Highway Networks and Gating (23 marks)

The professor explained highway networks as providing "more flexibility to the network where we can modulate the skip connection."

(a) Explain the highway network concept as described by the professor: "multiply the skip connection by a learnable function" and "multiply this by another function." Write the mathematical formulation the professor presented. (10 marks)

Answer: Highway networks introduce learnable gating mechanisms that control information flow through skip connections using transform gates $T(x)$ and carry gates $C(x) = 1 - T(x)$.

Mathematical Formulation:

The highway network output is:

$$y = H(x, W_H) \cdot T(x, W_T) + x \cdot C(x, W_C)$$

Where:

- $H(x, W_H)$ is the transformation function (e.g., affine transform + activation)
- $T(x, W_T) = \sigma(W_T x + b_T)$ is the transform gate (learnable)
- $C(x, W_C) = 1 - T(x, W_T)$ is the carry gate (complementary)
- σ is the sigmoid function, ensuring $T(x), C(x) \in [0, 1]$

Gating Mechanism Explanation:

- $T(x)$ controls how much transformed information $H(x)$ passes through
- $C(x)$ controls how much original information x passes through
- $T(x) + C(x) = 1$ ensures information conservation
- When $T(x) \approx 1$: mostly transformed information flows
- When $T(x) \approx 0$: mostly original information flows (like identity)

Professor's "Multiply by Functions" Explanation:

- "Multiply the skip connection by a learnable function": $x \cdot C(x)$

- "Multiply this by another function": $H(x) \cdot T(x)$
- Both multiplications are element-wise operations
- Creates adaptive blending of transformed and original information

(b) The professor showed how highway networks use gating: "t controls the information coming through here and with this we are also controlling how much x is propagated to the next layer." Explain this gating mechanism and how it differs from simple residual connections. (8 marks)

Answer: Highway network gating provides adaptive, learnable control over information flow, unlike ResNet's fixed additive skip connections. The gates $T(x)$ and $C(x)$ dynamically decide the mixing ratio between transformed and original information based on input content.

Gating Control Mechanism:

- $T(x)$ (transform gate) determines transformation strength
- $C(x) = 1 - T(x)$ (carry gate) determines identity strength
- Gates are input-dependent: $T(x) = \sigma(W_T x + b_T)$
- Values in $[0, 1]$ allow smooth interpolation between extremes

Adaptive Information Control:

- When input needs transformation: $T(x) \to 1, C(x) \to 0$
- When input should be preserved: $T(x) \to 0, C(x) \to 1$
- Intermediate values create custom mixing ratios
- Network learns optimal gating strategy during training

Differences from ResNet Skip Connections:

Highway Networks	**ResNet**
Learnable, adaptive gates	Fixed additive skip
$y = H(x) \cdot T(x) + x \cdot C(x)$	$y = H(x) + x$
Input-dependent behavior	Input-independent behavior
Multiplicative gating	Additive skip connection
Can completely block paths	Always preserves both paths
More parameters (gate weights)	Fewer parameters

(c) The professor noted that "highway networks were not as common as ResNets but the idea of gating is actually utilized in many different architectures." Give examples of where this gating concept is used and why it's important. (5 marks)

Answer: Highway network gating mechanisms are fundamental to LSTM cells, GRU units, attention mechanisms, and modern transformer architectures, where they provide selective information flow control and memory management.

Applications of Gating Mechanisms:

1. LSTM (Long Short-Term Memory):

- Input gate: controls new information storage
- Forget gate: controls old information deletion
- Output gate: controls information output to next layer
- Same sigmoid-based gating as highway networks

2. GRU (Gated Recurrent Unit):

- Update gate: similar to transform gate in highway networks
- Reset gate: controls how much past information to use
- Simplified version of LSTM gating

3. Attention Mechanisms:

- Attention weights act as soft gates
- Control information flow from different input positions
- Self-attention in transformers uses similar principles

Why Gating is Important:

- **Selective Processing:** Only relevant information passes through
- **Gradient Flow:** Maintains gradients for long sequences/deep networks
- **Adaptability:** Network learns optimal information routing
- **Memory Management:** Controls what to remember and forget

Question 5. Binary Networks and Efficiency (25 marks)
Based on the professor's discussion: "In binary networks people are trying to replace real valued weights with binary weights and this would get rid of multiplication."

(a) Explain the motivation for binary networks as the professor described: the need to "run these networks on low resource devices edge devices" where "it will be very difficult to run and execute deep architectures requiring a lot of memory." (8 marks)

Answer: Binary networks address the critical need for deploying deep learning on resource-constrained edge devices by dramatically reducing memory requirements and computational complexity through binary weight quantization.

Edge Computing Challenges:

- Mobile phones, IoT devices, embedded systems have limited memory (MB range)
- Standard deep networks require GB of memory for weights
- Limited computational power (CPU-only, low power consumption)
- Real-time inference requirements with strict latency constraints

Traditional Network Limitations:

- 32-bit floating point weights: 4 bytes per parameter
- Large networks (ResNet-50: 25M parameters = 100MB just for weights)
- Floating point multiplications are computationally expensive
- Memory bandwidth becomes bottleneck for inference speed

Binary Network Solutions:

- Weights constrained to $\{-1, +1\}$: only 1 bit per parameter
- Massive memory reduction: 32× smaller weight storage
- Replace expensive multiplications with simple additions/subtractions
- Enable deployment on severely resource-constrained devices

Application Scenarios:

- Smartphone on-device AI (privacy-preserving inference)
- IoT sensors with minimal power budgets
- Real-time embedded vision systems
- Offline inference where cloud connectivity is unavailable

(b) The professor mentioned specific trade-offs: "we get 30 times around 30 times gain in terms of memory but we lose some accuracy." Calculate the memory savings for a network with 10 million 32-bit parameters when converted to binary, and explain when this trade-off is acceptable. (10 marks)

Answer: Converting 10 million 32-bit parameters to binary reduces memory from 40MB to 1.25MB (32× reduction), closely matching the professor's "30 times gain" statement.

Memory Calculation:

Original Network (32-bit float):

- Parameters: 10,000,000
- Bits per parameter: 32 bits = 4 bytes
- Total memory: $10^7 \times 4 = 40,000,000$ bytes = 40 MB

Binary Network (1-bit):

- Parameters: 10,000,000
- Bits per parameter: 1 bit = 0.125 bytes
- Total memory: $10^7 \times 0.125 = 1,250,000$ bytes = 1.25 MB

Memory Reduction:

$$\text{Reduction Factor} = \frac{40 \text{ MB}}{1.25 \text{ MB}} = 32$$

When Trade-off is Acceptable:

Scenarios where accuracy loss is tolerable:

- **Mobile Applications:** Object detection for photo organization

- **IoT Sensors:** Basic classification tasks (sound/motion detection)
- **Real-time Systems:** Where speed > accuracy (video games, AR filters)
- **Privacy-critical:** On-device processing preferred over cloud

Scenarios where accuracy loss is unacceptable:

- Medical diagnosis systems
- Autonomous vehicle perception
- Financial fraud detection
- Safety-critical applications

Typical Accuracy Trade-offs:

- CIFAR-10: 2-5% accuracy drop
- ImageNet: 10-15% accuracy drop
- Simple tasks: minimal accuracy loss
- Complex tasks: significant accuracy loss

(c) The professor explained that "if you change the input to binary and work only with binary values throughout the network then actually in addition to obtaining significant memory gain you can actually get very large gain in running time." Explain why binary operations are faster and when this approach is practical. (7 marks)

Answer: Binary operations are faster because they replace expensive floating-point multiplications with simple bit operations, enable parallel XNOR computations, and utilize optimized hardware instructions, but require careful consideration of accuracy requirements.

Computational Speed Advantages:

1. Multiplication Replacement:

- Standard: $w \times x$ (floating-point multiplication)
- Binary: $\text{sign}(w) \times \text{sign}(x) = \text{XNOR} + \text{popcount}$
- XNOR operation is single CPU cycle vs. multiple cycles for FP multiplication

- Elimination of complex arithmetic logic units

2. Parallel Processing:

- 64 binary operations can be computed in parallel using 64-bit registers
- Bitwise operations are inherently vectorizable
- GPU threads can process multiple binary values simultaneously
- SIMD (Single Instruction, Multiple Data) optimization

3. Memory Access Patterns:

- Reduced memory bandwidth requirements
- Better cache utilization due to compact representation
- Fewer memory transfers between CPU/GPU

When Fully Binary Approach is Practical:

Suitable Applications:

- **Edge Computing:** Ultra-low power requirements
- **FPGA Implementations:** Custom hardware can be optimized for binary ops
- **Large-scale Inference:** Server farms processing millions of requests
- **Real-time Applications:** Where latency is more important than accuracy

Limitations:

- Significant accuracy degradation for complex tasks
- Training remains challenging (requires gradient approximations)
- Not suitable for tasks requiring high precision
- Limited support in current deep learning frameworks

Question 6. Introduction to Sequence Problems (35 marks)
The professor introduced sequence modeling: "We have many problems where we have a sequence we need to process that sequence sequentially."

(a) Classify the following problems according to the professor's framework (one-to-one, one-to-many, many-to-one, many-to-many) and explain your reasoning: (15 marks)

- Image captioning (which the professor said is "a very good example" of one-to-many)
- Spam detection (professor's example of many-to-one)
- Language translation (professor's example with "Turkish" to "English")
- Online speech recognition ("live speech recognition")
- Character recognition in a sequence

Answer: Classification based on input-output sequence relationships:

1. Image Captioning: One-to-Many

- **Input:** Single image (one)
- **Output:** Sequence of words forming caption (many)
- **Example:** Image → "A dog playing in the park"
- **Professor's Note:** "A very good example" of one-to-many
- **Reasoning:** Single visual input generates variable-length text sequence

2. Spam Detection: Many-to-One

- **Input:** Sequence of words in email (many)
- **Output:** Single classification (spam/not spam) (one)
- **Example:** "Free money click here now" → Spam
- **Professor's Note:** Direct example of many-to-one
- **Reasoning:** Variable-length text input produces single decision

3. Language Translation: Many-to-Many (Sequence-to-Sequence)

- **Input:** Sequence in source language (many)
- **Output:** Sequence in target language (many)
- **Example:** "Merhaba dünya" (Turkish) → "Hello world" (English)
- **Professor's Note:** Example with Turkish to English
- **Reasoning:** Both input and output are variable-length sequences

4. Online Speech Recognition: Many-to-Many (Streaming)

- **Input:** Continuous audio stream (many)
- **Output:** Continuous text stream (many)
- **Example:** Audio waveform → Real-time transcription
- **Professor's Note:** "Live speech recognition"
- **Reasoning:** Continuous input-output with temporal alignment

5. Character Recognition in Sequence: Many-to-Many

- **Input:** Sequence of character images (many)
- **Output:** Sequence of recognized characters (many)
- **Example:** Handwritten word image → "HELLO"
- **Reasoning:** Each input character position maps to output character

(b) The professor emphasized that in sequence problems "the information at a certain point might depend on the information we have seen so far." Explain why this context dependency makes traditional CNNs and MLPs insufficient for sequence modeling. (10 marks)

Answer: Traditional CNNs and MLPs lack memory mechanisms to maintain context across sequence positions, making them unable to capture temporal dependencies that are essential for sequence understanding.

Context Dependency in Sequences:

- **Language:** "The bank" can mean financial institution or river shore
- **Context determines meaning:** "money bank" vs. "river bank"

- **Temporal dependencies:** Current word meaning depends on previous words
- **Long-range dependencies:** Information from beginning affects end interpretation

Limitations of CNNs for Sequences:

- **Fixed receptive field:** Can only see limited local context
- **No temporal state:** Each convolution operation is independent
- **Translation invariance:** Doesn't distinguish between positions in sequence
- **Limited memory:** Cannot remember information from distant positions

Limitations of MLPs for Sequences:

- **Fixed input size:** Cannot handle variable-length sequences
- **No position awareness:** Treats all positions equally
- **No memory between samples:** Each input processed independently
- **Explosive parameters:** Would need separate weights for each sequence position

What Sequence Models Need:

- **Memory mechanism:** To maintain information across time steps
- **Variable-length processing:** Handle sequences of different lengths
- **Temporal state:** Internal state that updates as sequence progresses
- **Context integration:** Ability to combine current input with historical context

(c) Compare feedforward networks with RNNs as the professor explained: "In feedforward network we don't have that option but the network has seen for the previous input we don't know." Draw both architectures showing how RNNs provide "memory capacity" through recurrent connections. (10 marks)

- **END OF PAPER**