Diagrams

Below, I'll provide a **mapping** of all relevant diagrams from **Module 3: Convolutional Neural Networks (CNNs)**, **Module 4: Recurrent Networks**, and **Module 5: Recursive Neural Networks** based on the provided documents, along with simple text-based (ASCII) representations of these diagrams. These are designed to be Notion-friendly and can be copied directly into your notes for your open-book assessment on March 17, 2025. Since I can't generate graphical images, I'll use ASCII art to approximate the diagrams and provide detailed descriptions with page references for you to locate them in the original documents during the exam. You can enhance these in Notion with sketches if needed.

Mapping of Relevant Diagrams

Module 3: Convolutional Neural Networks (CNNs)

1. Convolutional Filter Application (Page 19)

- **Description**: Shows a 5×5 filter (e.g., loopy pattern) sliding over an image to produce a feature map.
- **Mapping**: Filter over input image → Output feature map.
- **ASCII Diagram**:

```
Input Image (e.g., 6×6):
| a | b | c | d | e | f |
| g | h | i | j | k | l |
| m | n | o | p | q | r |
| s | t | u | v | w | x |
| y | z | A | B | C | D |
| E | F | G | H | I | J |

Filter (5×5, Page 19):
| -1 |  1 |  1 |  1 | -1 |
| -1 |  1 | -1 |  1 | -1 |
| -1 |  1 |  1 |  1 | -1 |
| -1 |  1 |  1 |  1 | -1 |
| -1 | -1 |  1 |  1 | -1 |

Output (after convolution, e.g., stride=1, no padding):
| x1 | x2 |
| x3 | x4 |
```

- **Explanation**: The filter slides over the image (e.g., starting at top-left), computes dot products, and produces a smaller output (e.g., 2×2 for 6×6 input with 5×5 filter, no padding).

---

2. ResNet Residual Block (Page 147)

- **Description**: Illustrates a residual block with a skip connection adding the input to the output of convolutional layers.
- **Mapping**: Input → Convolutional Layers → Add Input → Output.
- **ASCII Diagram**:

```
Input (x) --——⟶ [Conv Layers (F(x))] --——⟶ (+) --——⟶ Output (y)
        |                                |
        |----------------------------------| (Skip Connection)
```

- **Explanation**: The input (x) passes through conv layers (F(x)), then adds back to itself via the skip connection, producing y = F(x) + x.

---

3. AlexNet Architecture (Page 153)

- **Description**: Shows the 8-layer structure: 5 convolutional layers followed by 3 fully connected layers.
- **Mapping**: Input → Conv1 → Pool1 → Conv2 → ... → FC1 → FC2 → Output.
- **ASCII Diagram**:

```
Input (227×227×3) ⟶ [Conv1] ⟶ [Pool1] ⟶ [Conv2] ⟶ [Pool2] ⟶ [Conv3] ⟶ [Conv4] --
> [Conv5] ⟶ [FC1] ⟶ [FC2] ⟶ Output
```

- **Explanation**: Starts with a 227×227×3 image, processes through 5 conv + pooling layers, then 3 fully connected layers for classification.

---

4. Inception Module (Page 165)

- **Description**: Depicts an inception block with parallel 1×1, 3×3, 5×5 convolutions and pooling, concatenated at the end.
- **Mapping**: Input → Parallel Filters (1×1, 3×3, 5×5, Pool) → Concatenate → Output.
- **ASCII Diagram**:

```
Input --——⟶ [1×1 Conv] ----\\
    |⟶ [3×3 Conv] ----|⟶ [Concatenate] ⟶ Output
    |⟶ [5×5 Conv] ----|
    \\⟶ [3×3 Pool] ----/
```

- **Explanation**: The input splits into four paths with different filters/pooling, then recombines into a single output.

Module 4: Recurrent Networks

5. RNN Basic Structure (Page 20)

- **Description**: Shows the recurrent loop where the hidden state is passed back to the next time step.

- **Mapping**: Input (x_t) → Hidden (h_t) → Output (y_t), with h_t looping back.
- **ASCII Diagram**:

```
x_t ⟶ [h_t] ⟶ y_t
      ^  |
      |__| (Loop)
```

- **Explanation**: At each time step t, input x_t updates hidden state h_t, which produces y_t and loops back for the next step.

---

6. Bidirectional RNN (Page 4, Inferred)

- **Description**: Illustrates forward and backward RNNs combining to produce an output.
- **Mapping**: Input → Forward h_f + Backward h_b → Output.
- **ASCII Diagram**:

```
x_t ⟶ [h_f (Forward)] --\\
                        |⟶ [Combine] ⟶ y_t
x_t ⟶ [h_b (Backward)] --/
```

- **Explanation**: Processes sequence in both directions, combining forward (h_f) and backward (h_b) states.

---

7. Encoder-Decoder (Page 4)

- **Description**: Shows an encoder compressing a sequence into a context, decoded into an output sequence.
- **Mapping**: Input Sequence → Encoder → Context → Decoder → Output Sequence.
- **ASCII Diagram**:

```
Input Seq (x1, x2, x3) ⟶ [Encoder] ⟶ Context ⟶ [Decoder] ⟶ Output Seq (y1, y2, y3)
```

- **Explanation**: Encoder compresses input into a fixed context, which the decoder uses to generate output.

Module 5: Recursive Neural Networks

8. LSTM Cell (Page 207)

- **Description**: Depicts the LSTM cell with forget, input, and output gates controlling the cell state and hidden state.
- **Mapping**: Input (x_t) + Previous (h_{t-1}, C_{t-1}) → Gates → New (h_t, C_t).
- **ASCII Diagram**:

```
x_t, h_{t-1} ⟶ [f_t (Forget)] --\\
C_{t-1} ----⟶ [i_t (Input)] ---|⟶ [C_t] ⟶ [o_t (Output)] ⟶ h_t
```

```
                [tilde{C}_t] ---/
```

- **Explanation**: Gates (f_t, i_t, o_t) regulate what to forget, add, and output from cell state C_t to hidden state h_t.

---

9. GRU Cell (Page 235)

- **Description**: Shows the GRU cell with reset and update gates updating the hidden state.
- **Mapping**: Input (x_t) + Previous (h_{t-1}) → Reset (r_t) + Update (z_t) → New (h_t).
- **ASCII Diagram**:

```
x_t, h_{t-1} ⟶ [r_t (Reset)] --\\
                [z_t (Update)] |⟶ [h_t]
                [tilde{h}_t] --/
```

- **Explanation**: Reset gate (r_t) controls past influence, update gate (z_t) blends old and new info into h_t.

---

10. Echo State Network (Page 229, Inferred)

- **Description**: Illustrates a fixed reservoir with trainable output weights.
- **Mapping**: Input → Reservoir → Output.
- **ASCII Diagram**:

```
Input (x_t) ⟶ [Reservoir (r)] ⟶ [W_out] ⟶ Output (y)
```

- **Explanation**: Input drives a fixed recurrent reservoir, and only the output layer (W_out) is trained.

Integration into Notion Notes

Below is how you can integrate these diagrams into your existing Notion-friendly notes (from the previous response). I'll add them under the relevant sections with page references.

Module 3: Convolutional Neural Networks (CNNs)

1. Foundations of CNNs

- **Definition**: Feed-forward neural networks inspired by the visual cortex, designed for image processing with parameter sharing (Pages 3, 8).
- **Why CNNs?**:
  - ANNs inefficient for large images (e.g., 1920 × 1080 × 3 ≈ 6M neurons) (Page 6).
  - CNNs use local patterns, reducing parameters (Page 14).

2. Key Components

- **Convolutional Layers**:
  - Filters detect edges, textures (Page 10).
  - Formula: Output size = (n - f) / s + 1 (n = input size, f = filter size, s = stride) (Page 158).
  - **Diagram (Page 19)**:

```
Input Image (e.g., 6×6):
| a | b | c | d | e | f |
| g | h | i | j | k | l |
| m | n | o | p | q | r |
| s | t | u | v | w | x |
| y | z | A | B | C | D |
| E | F | G | H | I | J |

Filter (5×5):
| -1 |  1 |  1 |  1 | -1 |
| -1 |  1 | -1 |  1 | -1 |
| -1 |  1 |  1 |  1 | -1 |
| -1 |  1 |  1 |  1 | -1 |
| -1 | -1 |  1 |  1 | -1 |

Output (2×2):
| x1 | x2 |
| x3 | x4 |
```

5. Deep Convolutional Models

a. ResNet

- **Concept**: Residual blocks with skip connections (Page 147).
- **Formula**: y = F(x) + x (F = conv layers).
- **Diagram (Page 147)**:

```
Input (x) --——→ [Conv Layers (F(x))] --——→ (+) --——→ Output (y)
        |                                |
        |----------------------------------|
```

b. AlexNet

- **Architecture**: 5 Conv + 3 FC, ReLU, overlapping pooling (Page 153).
- **Diagram (Page 153)**:

```
Input (227×227×3) → [Conv1] → [Pool1] → [Conv2] → [Pool2] → [Conv3] → [Conv4] --
> [Conv5] → [FC1] → [FC2] → Output
```

c. InceptionNet

- **Concept**: Inception modules: 1 × 1, 3 × 3, 5 × 5 + pooling (Page 165).
- **Diagram (Page 165)**:

```
Input --——→ [1×1 Conv] ----\\
    |→ [3×3 Conv] ----|→ [Concatenate] → Output
    |→ [5×5 Conv] ----|
    \\→ [3×3 Pool] ----/
```

Module 4: Recurrent Networks

1. Recurrent Neural Networks (RNNs)

- **Definition**: Networks with loops, memory via h_t (Page 15).
- **Formula**: h_t = tanh(W_h * h_{t-1} + W_x * x_t + b) (Page 15).
- **Diagram (Page 20)**:

```
x_t → [h_t] → y_t
      ^  |
      |__|
```

2. Bidirectional RNNs

- **Concept**: Forward (h_f) + Backward (h_b) (Page 4).
- **Diagram (Page 4)**:

```
x_t → [h_f (Forward)] --\\
                        |→ [Combine] → y_t
x_t → [h_b (Backward)] --/
```

3. Encoder-Decoder & Seq2Seq

- **Encoder**: Compresses sequence (Page 4).
- **Diagram (Page 4)**:

```
Input Seq (x1, x2, x3) → [Encoder] → Context → [Decoder] → Output Seq (y1, y2, y3)
```

Module 5: Recursive Neural Networks

2. Long Short-Term Memory (LSTM)

- **Architecture**: Cell state C_t, gates: f_t, i_t, o_t (Page 207).
- **Formulae**:

- C_t = f_t * C_{t-1} + i_t * tilde{C}_t
- h_t = o_t * tanh(C_t)

- **Diagram (Page 207)**:

```
x_t, h_{t-1} ⟶ [f_t (Forget)] --\\
C_{t-1} ----⟶ [i_t (Input)] ---|⟶ [C_t] ⟶ [o_t (Output)] ⟶ h_t
            [tilde{C}_t] ---/
```

3. Gated Recurrent Unit (GRU)

- **Architecture**: Reset (r_t), Update (z_t) (Page 233).
- **Formulae**:
  - r_t = sigmoid(W_r * [h_{t-1}, x_t])
  - z_t = sigmoid(W_z * [h_{t-1}, x_t])
  - tilde{h}*t = tanh(W_h * (r_t * h*{t-1}) + W_x * x_t + b)
  - h_t = (1 - z_t) * h_{t-1} + z_t * tilde{h}_t
- **Diagram (Page 235)**:

```
x_t, h_{t-1} ⟶ [r_t (Reset)] --\\
            [z_t (Update)] |⟶ [h_t]
            [tilde{h}_t] --/
```

4. Echo State Networks (ESNs)

- **Concept**: Fixed reservoir, trainable output (Page 214).
- **Diagram (Page 229)**:

```
Input (x_t) ⟶ [Reservoir (r)] ⟶ [W_out] ⟶ Output (y)
```

Tips for Notion

- **Copy-Paste**: Add these under relevant sections in your Notion notes.
- **Enhance**: Use Notion's drawing tools to refine ASCII art (e.g., arrows, boxes) during prep.
- **Page References**: Use during the exam to locate originals (e.g., "Page 19" for filter).

These diagrams cover all key structures from the documents, mapped and drawn in text form for easy integration. Let me know if you need further assistance!