Question 1 Answer :

For designing an ANN-based system to navigate a self-driving car in urban environments, a convolutional neural network (CNN) combined with recurrent neural network (RNN) layers would be appropriate. This design would leverage the spatial hierarchical pattern recognition capabilities of CNNs and the temporal dynamic behavior understanding of RNNs, suitable for processing data from multiple sensors under varying conditions.

Network Architecture:

1. Input Layer: The input to the network would be multidimensional data from cameras, LIDAR, radar, and other sensors. Each type of data would be preprocessed appropriately to normalize the range and improve the neural network's learning efficiency.

2. Convolutional Layers: The first part of the network would consist of multiple convolutional layers. For example, three convolutional layers with 32, 64, and 128 filters respectively. These layers help in detecting features like edges, shapes, and other road and object characteristics from visual and spatial sensor data. Each convolutional layer would be followed by a max-pooling layer to reduce dimensionality and improve computational efficiency.

3. Recurrent Layers: After convolutional layers, I would add LSTM (Long Short-Term Memory) layers to process the temporal sequence data from the sensors. Two LSTM layers with 100 units each would help the system understand the dynamics of the environment over time, crucial for predictive navigation and decision-making.

4. Fully Connected Layers: Towards the end, a few fully connected layers (e.g., 512 and 256 neurons) with ReLU activation function would integrate the learned features into a format suitable for making driving decisions.

5. Output Layer: The output layer would consist of several neurons representing different potential actions or driving commands, using a softmax activation function to output a probability distribution over possible maneuvers.

Handling Varying Conditions:

- Data Augmentation: To ensure robustness against varying lighting and weather conditions, the training data would include augmented variations simulating different times of day, weather conditions like rain or fog, and different seasons. This approach helps the model generalize well across different scenarios.

- Dynamic Adjustment: Incorporate feedback loops within the network that adjust sensitivity based on real-time assessments of visibility and sensor reliability. For instance, in poor visibility conditions, the system could rely more on radar and less on visual input.

- Regularization and Dropout: To prevent overfitting and ensure the model remains generalizable, I would implement dropout layers between the fully connected layers and possibly use batch normalization after each convolutional layer to stabilize the learning process.

Rubric for Evaluating the Answer:

- Architectural Design (30 points): Clarity in explaining the integration of CNN and RNN layers, the rationale behind the number of layers, and types of layers used.

- Handling Variability (30 points): Specific strategies to adapt to different environmental conditions, including data augmentation and dynamic adjustment techniques.

- Technical Details (30 points): Detailed description of the number of nodes in each layer, types of activation functions used, and any additional network components (e.g., dropout, batch normalization).

- Clarity and Structure (10 points): Overall clarity of the explanation, logical flow, and how well the response is organized and presented.

Question 2 Answer:

For an ANN model designed to diagnose diseases from complex medical imaging like MRIs or CT scans, a deep convolutional neural network (CNN) would be optimal due to its ability to process spatial hierarchies in image data. The architecture will focus on extracting intricate features at various levels, critical for identifying subtle differences in disease stages.

Network Architecture:

1. Input Layer: The input layer would handle images resized to a consistent dimension, such as 256x256 pixels, with preprocessing steps including normalization to scale the pixel values between 0 and 1.

2. Convolutional Layers: The core of the network would consist of multiple convolutional layers. Starting with two initial layers of 32 filters each for basic feature extraction, followed by additional layers with increasing numbers of filters (e.g., 64, 128, 256) to capture more complex features. Each convolutional layer would use the ReLU activation function to introduce non-linearity and batch normalization to speed up training and reduce overfitting.

3. Pooling Layers: Max pooling layers would be interspersed between convolutional layers to reduce dimensionality and to increase the field of view of higher layers, which helps in detecting features in larger regions of the image.

4. Fully Connected Layers: After flattening the output of the last convolutional layer, one or two fully connected layers with 512 neurons each would integrate the high-level features extracted by the convolutional layers. These layers would also use ReLU activation.

5. Output Layer: For a multi-class classification of disease stages, the output layer would consist of neurons equal to the number of stages, using a softmax activation function to output a probability distribution over the stages.

Training Strategy:
- Data Augmentation: To train the network to recognize subtle variations in medical images, extensive data augmentation (rotations, scaling, translations) would be employed to simulate different imaging conditions and patient positions.

- Advanced Techniques: Using techniques like transfer learning, where the model is initially trained on a large dataset of medical images before fine-tuning on a specific disease, can help the model learn nuanced differences in the disease stages effectively.

Loss Function:
- Cross-Entropy Loss: This loss function is appropriate for multi-class classification problems. It is effective because it penalizes incorrect classifications heavily when the model is confident about its wrong predictions, which is crucial for medical diagnostics where mistakes can have serious consequences.

- Weighted Loss Function: Considering the critical nature of accurate diagnosis and the potential imbalance in the dataset (some stages may be less common), applying a weighted cross-entropy loss can help prioritize learning from underrepresented classes, ensuring the model performs well across all disease stages.

Rubric for Evaluating the Answer:
- Architectural Details (30 points): Detailed description of the types and arrangements of layers, including activation functions and any special components like batch normalization.

- Training Strategy (30 points): Explanation of how data augmentation and advanced techniques like transfer learning are used to improve the model’s ability to distinguish between subtle variations.

- Choice of Loss Function (20 points): Justification for the choice of cross-entropy loss and the consideration of a weighted scheme to address class imbalance.

- Clarity and Structure (20 points): Overall clarity of the explanation, coherence in the flow of ideas, and how effectively the response is organized and articulated.