DEEP LEARNING ANSWERS

1.

a) Implementing Deep Learning in a Real-World Application

Implementing Deep Learning (DL) in a real-world application involves several steps and considerations:

Problem Understanding:

First, clearly define the problem you want to solve with DL. This could be image recognition, natural language processing, time-series prediction, etc.

Data Collection and Preprocessing:

Gather relevant data for your problem. This could be from databases, APIs, or other sources. Data preprocessing is crucial to clean, normalize, and prepare the data for training.

Model Selection:

Choose the appropriate deep learning model architecture for your problem. This could be Convolutional Neural Networks (CNNs) for image tasks, Recurrent Neural Networks (RNNs) for sequential data, or Transformer models for NLP tasks.

Model Building

Construct the neural network using libraries like TensorFlow or PyTorch. This involves defining the layers, activation functions, loss functions, and optimization algorithms.

Training:

Train the model on the prepared data. This step requires a large amount of computational resources, especially for complex models and large datasets. Training involves forward and backward passes, where the model adjusts its weights to minimize the loss function.

Hyperparameter Tuning:

Fine-tune the model by adjusting hyperparameters such as learning rate, batch size, and regularization techniques to improve performance.

Validation and Testing:

Evaluate the trained model on a separate validation set to assess its performance and make adjustments if needed. Finally, test the model on unseen data to assess its real-world performance.

Deployment:

Once the model is trained and validated, it needs to be deployed to a production environment. This involves integrating the model into an application or system where it can make predictions on new, unseen data.

Monitoring and Maintenance:

Continuous monitoring of the deployed model is essential to ensure it performs as expected. Models may need retraining with new data over time to maintain their accuracy.

(b) Activation Functions in Artificial Neural Networks (ANNs)

Activation functions are crucial components of artificial neural networks. They introduce non-linearity into the network, allowing it to learn complex patterns and make the network capable of approximating any continuous function. Here's why activation functions are used and the problems if we don't use them:

Introducing Non-Linearity:

Activation functions introduce non-linear properties to the network, enabling it to learn and represent non-linear and complex relationships in the data. Without non-linear activation functions, the entire neural network would behave as a linear model, regardless of its depth, as it would just be a series of linear operations stacked together.

Squashing the Output:

Activation functions also "squash" the output of each neuron to a specific range. For example, sigmoid function squashes values between 0 and 1, tanh between -1 and 1, and ReLU (Rectified Linear Activation) function keeps only positive values. This normalization of output helps in stabilizing and speeding up the training process.

Problem without Activation Functions:

If we were to remove activation functions from neural networks, each layer would simply be a linear transformation of the previous layer. The network would then be limited to learning only linear transformations of the input data, severely limiting its capacity to learn complex patterns. Essentially, it would behave like a single-layer linear regression model, no matter how many layers we add.

Vanishing Gradient:

Another problem is the vanishing gradient problem. In deep networks with many layers and without non-linear activation functions, gradients can become very small during backpropagation. This makes training d…