📚 Deep Learning for AI — Final Exam Notes

✏️ ANN (Artificial Neural Networks)

🧠 Theory

1. Forward Propagation

- Data moves from input layer ➔ hidden layer ➔ output layer.
- Each layer applies weights, biases, and activation functions to pass data forward.

2. Backward Propagation

- Error is calculated at output.
- The error is propagated backward through the network to adjust weights.

3. Weight Adaptation

- **Gradient Descent** is used to minimize the loss by updating weights.
- **Weight Update Formula**:

sql

CopyEdit

New Weight = Old Weight – (Learning Rate × ∂Loss/∂Weight)

4. Activation Functions

Function	Usage
Sigmoid / Softmax	Used in **output layers** (classification).
ReLU / Leaky ReLU	Used in **hidden layers** (to avoid vanishing gradients).

- **Sigmoid**: squashes output between 0 and 1 (good for binary outputs).
- **Softmax**: converts outputs into probability distribution (multi-class).
- **ReLU**: fast and avoids vanishing gradient.
- **Leaky ReLU**: allows small gradient even for negative values (fixes dying ReLU problem).

5. Loss Functions

- Loss measures the error between prediction and actual output.
- Common losses:
  - **Binary Crossentropy** (binary classification)
  - **Categorical Crossentropy** (multi-class classification)
  - **MSE** (regression)

6. Optimizers

- Algorithms that adjust learning rate/steps.
- Common types:
    - **SGD** (Stochastic Gradient Descent)
    - **Adam** (Adaptive learning rate optimizer)

7. Overfitting

- When model memorizes training data but fails on unseen data.
- **Solutions**:
    - Dropout
    - Early stopping
    - Regularization (L1, L2 penalties)

8. Regularization Techniques

- **Dropout**: Randomly drops neurons during training to prevent dependency.
- **Early Stopping**: Stops training when validation loss stops improving.

9. Parameter Calculation

- **Formula**:

scss

CopyEdit

(Input nodes × Output nodes) + (Bias nodes × Output nodes)

- Helps calculate total trainable parameters between layers.

✏️ CNN (Convolutional Neural Networks)

🧠 Theory

1. Convolution Layer

- Applies filters (kernels) to extract features (edges, textures) from input images.

2. Pooling Layer

- Reduces spatial size of features (downsampling).
- Types:
    - Max pooling (most common)
    - Average pooling

3. Stride

- Number of pixels by which filter moves over input.
- Larger stride ➔ smaller output size.

4. Padding

- Adding zeros around the input to preserve spatial dimensions.
- Needed for:
  - Edge/corner pixels
  - Same input-output size

5. Output Dimension Formula

mathematica

CopyEdit

Output Size = (Input Size – Kernel Size + 2 × Padding) / Stride + 1

✏️ Transfer Learning (Optional Alternative for CNN Part)

- Pre-trained Models: VGG16, VGG19, ResNet50, InceptionV3
- Load pre-trained model ➔ add custom output layers ➔ train on your data.

✏️ RNN (Recurrent Neural Networks)

🧠 Theory

1. Why ANN is not suitable for sequence/text

- ANN cannot remember order or context (no memory).

2. RNN Architecture

- Has loops that allow information to persist (memory of past).

3. Drawbacks of RNN

- **Short-term memory**: cannot remember long sequences.
- **Vanishing gradient**: training becomes difficult for long sequences.

4. Advanced Architectures

- **LSTM**: Long Short-Term Memory — solves vanishing gradient problem.
- **GRU**: Gated Recurrent Units — simpler, faster version of LSTM.

🛡️ Final Tip

- Focus mainly on **running codes** correctly in Kaggle — simple structure wins.
- Memorize **key theory bullet points**.
- Practice **formula for output size** and **parameter counts**.
- Understand **activation choices** clearly (hidden layer = ReLU, output = Sigmoid/Softmax).