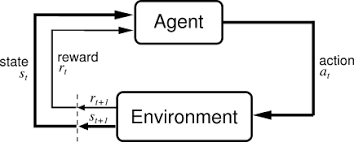
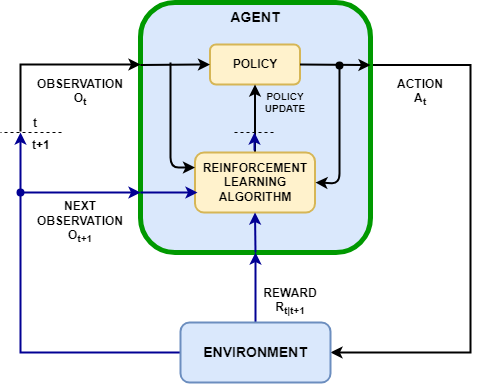
**eeMODULE = 5**

**NEURAL NETWORK**

**⭐ 1. Reinforcement Learning (RL)**





4

**What is Reinforcement Learning?**

Reinforcement Learning is a type of machine learning where:

* An **agent** interacts with an **environment**
* It takes **actions**
* Receives **rewards** (good/bad)
* Learns the **best strategy (policy)** to maximize long-term rewards.

👉 Think of it like training a dog—if it does something good, you reward it; if it does something wrong, you say “no”.  
Similarly, the RL agent learns from trial and error.

**Important Terms (Tuples of RL)**

**1. Agent**

The decision maker  
Example: Robot, self-driving car, game-player AI.

**2. Environment**

Where the agent operates  
Example: Road, chessboard, game world.

**3. State (S)**

Current situation  
Example: Car’s current position, chessboard configuration.

**4. Action (A)**

What the agent can do  
Example: Move forward, turn left, pick object.

**5. Reward (R)**

Feedback  
Positive → good action  
Negative → bad action

**6. Policy (π)**

Rule or strategy for choosing actions  
Example: “If obstacle is near → slow down”

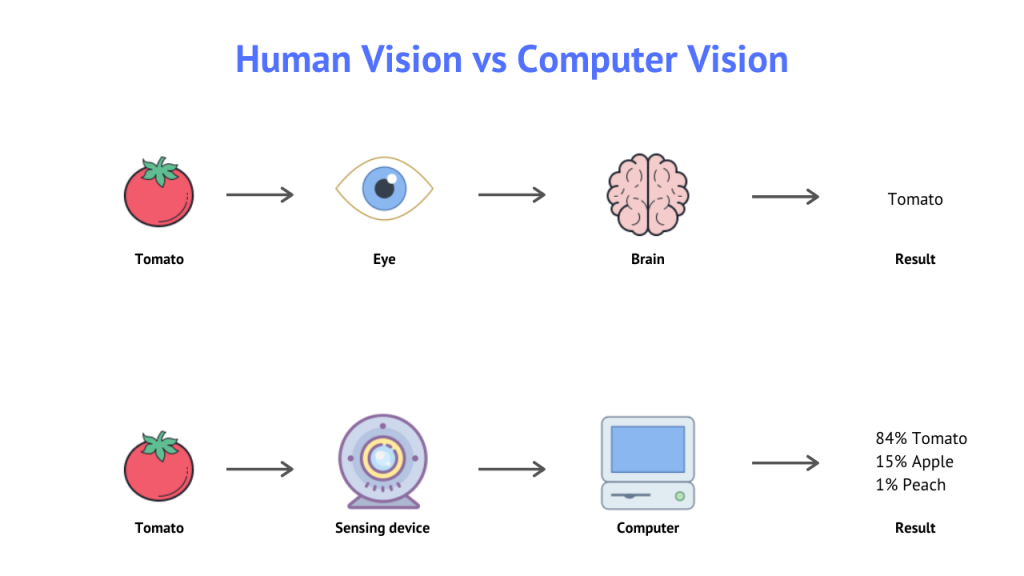
**7. Value Function (V)**

How good a state is (expected future reward)

**8. Q-Value Q(s, a)**

How good an action is in a specific state

**⭐ 2. Computer Vision (CV)**



4

**What is Computer Vision?**

Computer Vision allows computers to **see, understand, and interpret images/videos** just like humans.

Example tasks:

* Detect objects
* Classify images
* Recognize faces
* Read text (OCR)

**How Computer Vision Works? (Human vs Computer)**

**Human Vision**

Eye → Brain → Understanding of objects

**Computer Vision**

Camera → Algorithms → Understanding of image

Steps:

1. Input image/video is captured
2. CV system processes image
3. It extracts patterns/features
4. Recognizes object (using trained models)

**Computer Vision Tasks**

**🔹 Image Classification**

Identify what object is in image (cat/dog/car)

**🔹 Object Detection**

Detect multiple objects + draw bounding boxes

**🔹 Image Segmentation**

Divide image into meaningful regions (road, car, tree)

**🔹 Edge Detection**

Find boundaries of objects

**🔹 Face & Person Recognition**

Used in biometrics (Face ID)

**🔹 Motion Analysis**

Track object movement in video

**Applications of Computer Vision**



4

Examples:

* **Intruder detection** (security cameras)
* **Metrology** (measurement in manufacturing)
* **Defect detection** (quality control)
* **Robotics + CV** (bin picking in factories)
* **Assembly verification** (checking correct parts)
* **Screen readers** (for blind people)
* **OCR** (reading text from images)

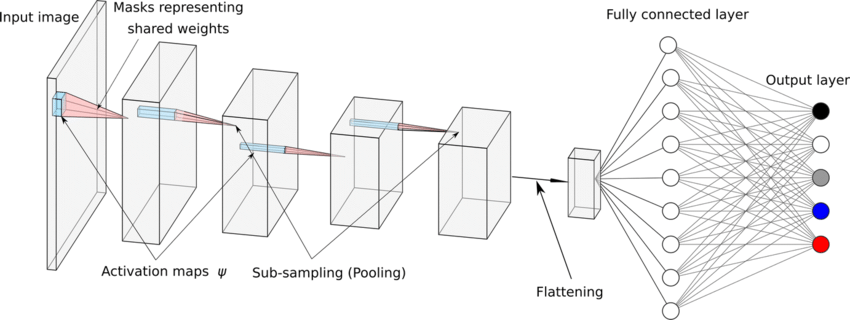
**⭐ 3. Deep Learning for Computer Vision**

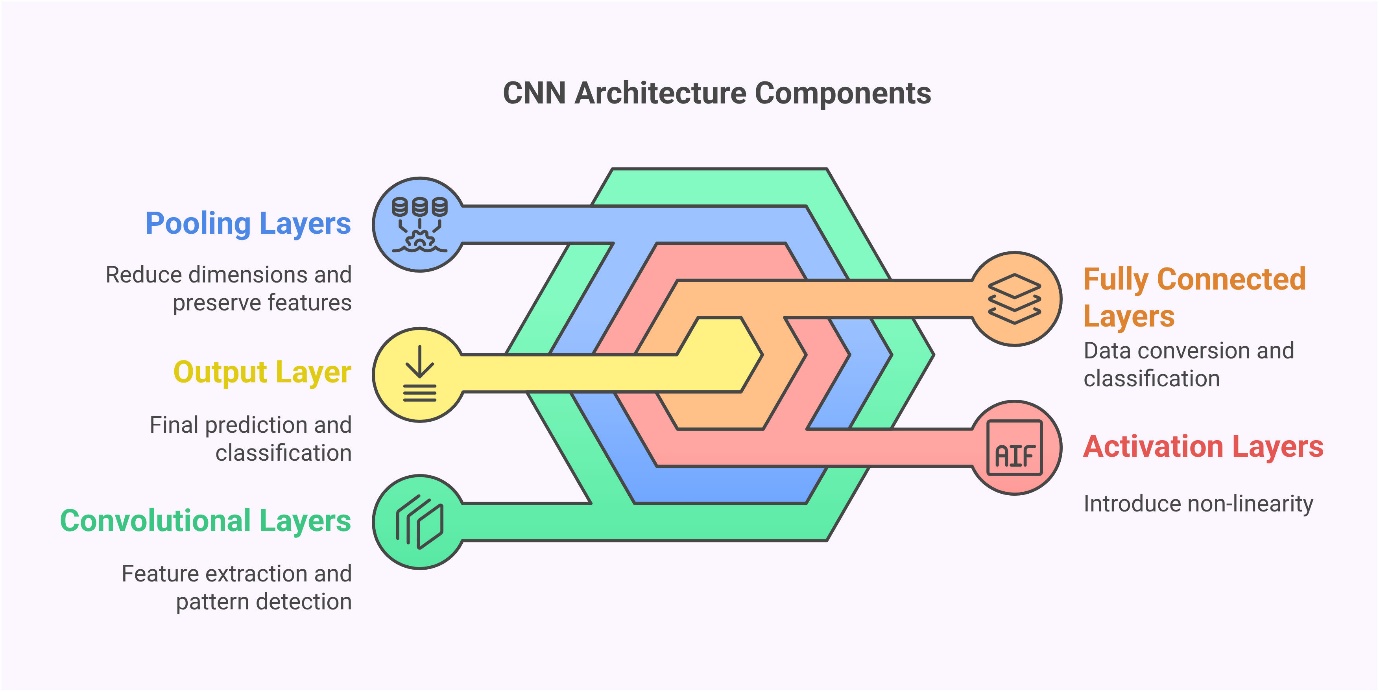
Includes:

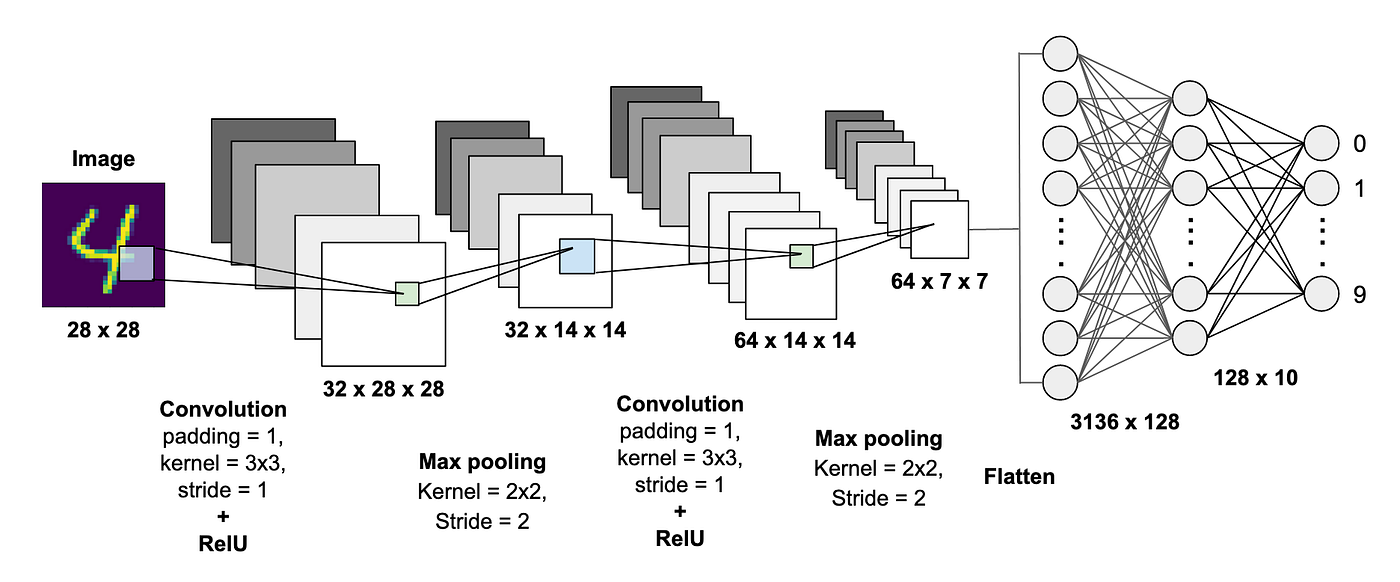
1. CNNs (Convolutional Neural Networks)
2. GANs
3. VAEs
4. Vision Transformers (ViT)
5. Vision-Language Models (like CLIP)

But for your exam, **CNN is most important**.

**⭐ 4. Convolutional Neural Networks (CNN)**







4

CNN is a special type of Deep Learning model designed for **images**.

Why CNNs?

* They automatically extract features
* More efficient than traditional ANN

**⭐ Working of CNN (Step-by-Step)**

**1. Input Image**

Example: 28×28 grayscale picture

**2. Convolution Layer**

This is the most important part.

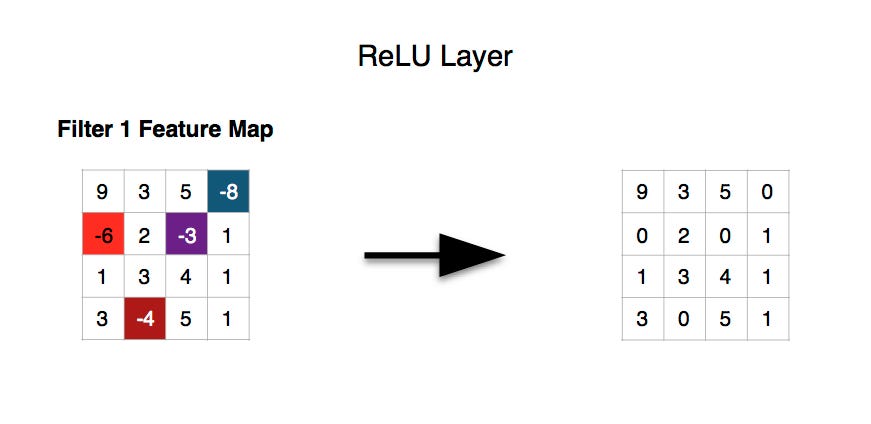
**What happens?**

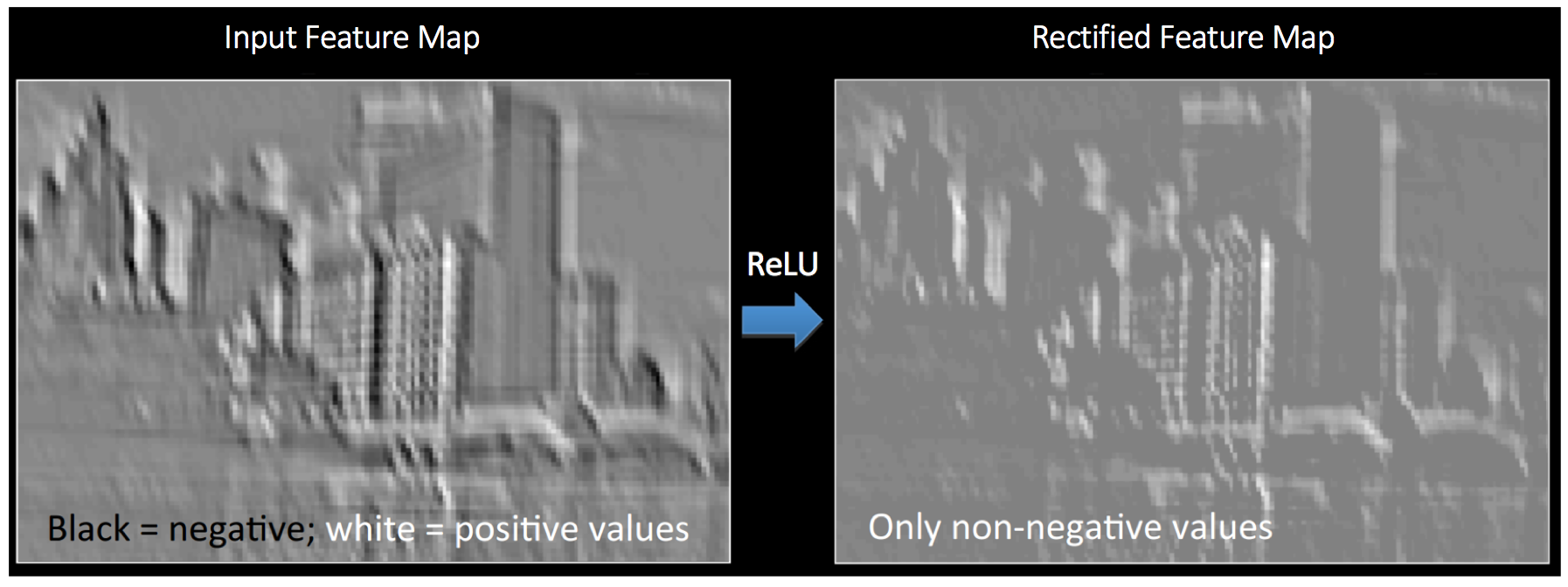
* A **filter/kernel** (3×3 or 5×5 matrix) slides over image
* Performs dot product
* Creates **feature map**

Result:

* Detects edges, textures, shapes

**3. Activation Function (ReLU)**





4

ReLU = Rectified Linear Unit  
Formula: **ReLU(x) = max(0, x)**  
It removes negative values → keeps only important features.

**4. Padding**

Adds zeros around image to:

* Prevent shrinking of image
* Preserve border information

Types:

* Valid Padding (no padding)
* Same Padding (output = input size)

**5. Stride**

How many pixels the filter moves at a time  
Stride 1 → moves 1 pixel  
Stride 2 → moves 2 pixels (reduces output size)

**6. Pooling Layer**

Reduces size of feature map  
Most common: **Max Pooling (2×2)**

Benefits:

* Reduces computation
* Keeps strong features

**7. Flatten Layer**

Converts 2D feature maps → 1D vector  
Required before passing to dense layers.

**8. Fully Connected Layer**

Learns patterns and performs classification.

**9. Output Layer**

Uses **Softmax** for multi-class classification.

Example:  
Image → [Horse: 0.2, Zebra: 0.7, Dog: 0.1]

**⭐ 5. Popular CV Libraries**

1. **OpenCV** – image processing
2. **TensorFlow** – deep learning
3. **PyTorch** – research deep learning
4. **scikit-image** – classical image processing

**⭐ 6. Need for Computer Vision**

* High job demand
* Important for automation
* Powers self-driving cars
* Helps medical diagnosis
* Improves accessibility
* Enhances customer experience

**⭐ Ready-to-Use Exam Explanation**

If your teacher asks:

**Q. Explain Convolution?**

Ans: Convolution is sliding a filter over an image and computing dot product to produce a feature map. It extracts features like edges, textures, corners.

**Q. Why ReLU?**

Ans: It introduces non-linearity and removes negative values, making training faster.

**Q. What is pooling?**

Ans: Pooling reduces image dimensions and keeps important features. Max pooling selects the largest value in each region.

**Q. What is stride?**

Ans: Stride determines how the filter moves across the image. Larger stride reduces output size.

**Q. Why CNN over ANN?**

Ans: CNN captures spatial patterns (edges, textures) and is less computationally expensive.

**⭐ 1. PADDING (Very Important Concept)**



4

Padding = **adding zeros around the borders of the input image**.

**🔥 Why do we use Padding?**

Because when we apply convolution, the output size becomes smaller.  
Padding helps to:

✔ Preserve information at the edges  
✔ Keep output size same as input (for SAME padding)  
✔ Prevent shrinking after every convolution

**💡 Example from your image:**

You see:

* Input → surrounded by zeros
* Kernel slides over padded image
* Output size becomes bigger because borders are included

Padding Types:

**1️⃣ VALID Padding**

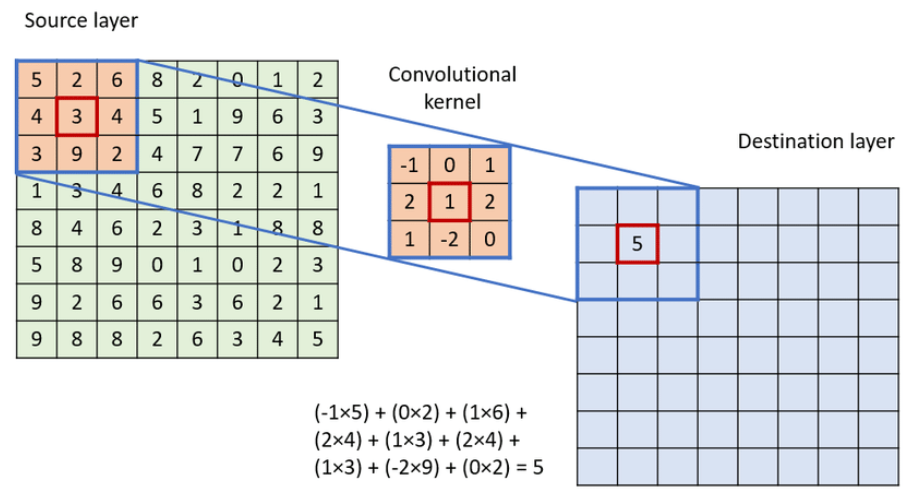
* No padding
* Output size shrinks
* Faster but loses border info

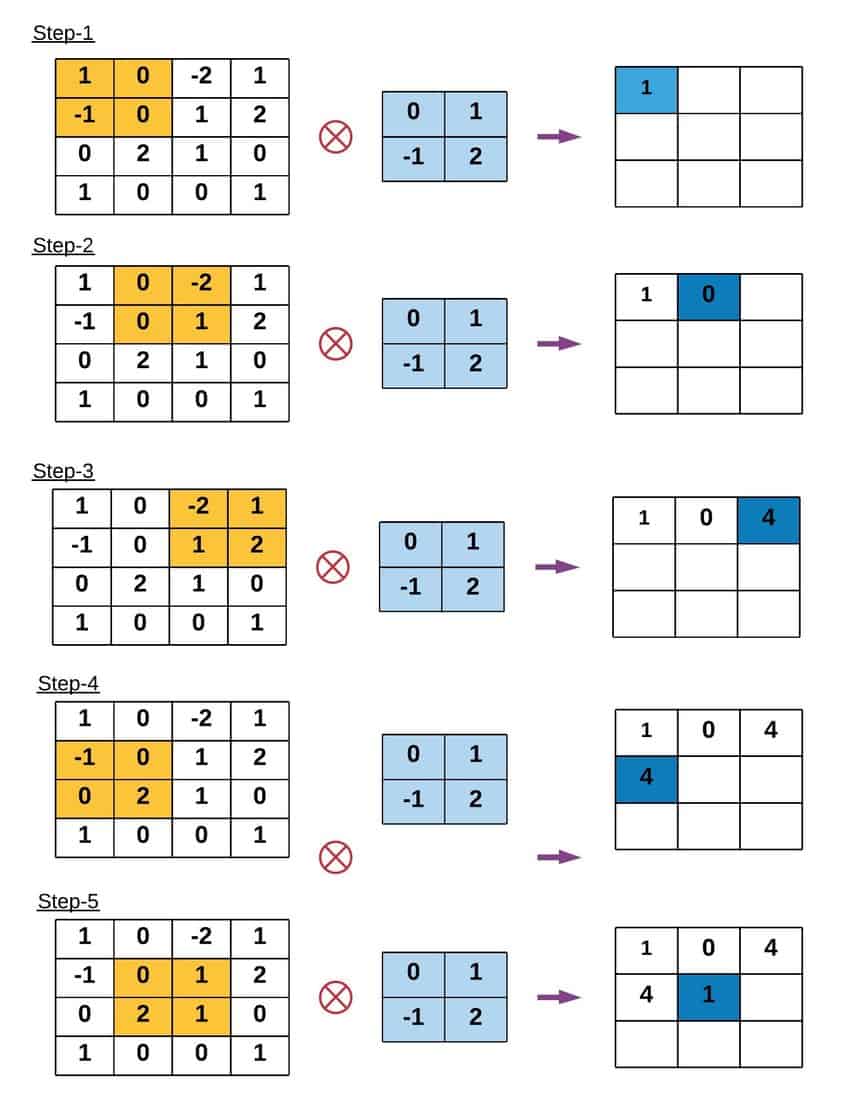
**2️⃣ SAME Padding**

* Zeros added
* Output size stays **same as input**
* Most commonly used in CNNs

**⭐ 2. Convolution Example (Your second image)**







4

Input matrix is padded → Kernel of 3×3 is applied → Dot product is calculated.

Kernel given:

0 -1 0

-1 5 -1

0 -1 0

This is a **sharpening filter** (enhances edges).

When the filter multiplies a patch of the image and sums values, we get output values like:

114, 328, -26, 470, 158 …

This forms the **Feature Map**.

👉 **Purpose:** Extract edges, textures, patterns.

**⭐ 3. POOLING (Max Pooling & Average Pooling)**

4

Pooling reduces the size of feature maps → **Down-sampling**.

**⭐ Max Pooling (Most common)**

Filter = 2×2  
Stride = 2

Look at your image:

**Block 1 (Top-left)**

2 2

9 4

Max = **9**

**Block 2 (Top-right)**

7 3

6 1

Max = **7**

**Block 3 (Bottom-left)**

8 5

3 1

Max = **8**

**Block 4 (Bottom-right)**

2 4

2 6

Max = **6**

So the output =

9 7

8 6

**⭐ Average Pooling**

Same 2×2 blocks, but you take **average** instead of max.

Example block:

2 2

9 4

Average = (2+2+9+4) / 4 = **4.25**

Hence all the 2×2 blocks give output like:

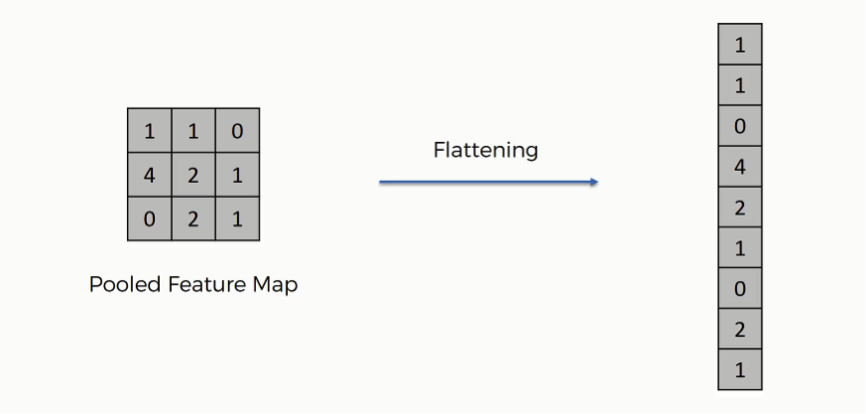
4.25 4.25

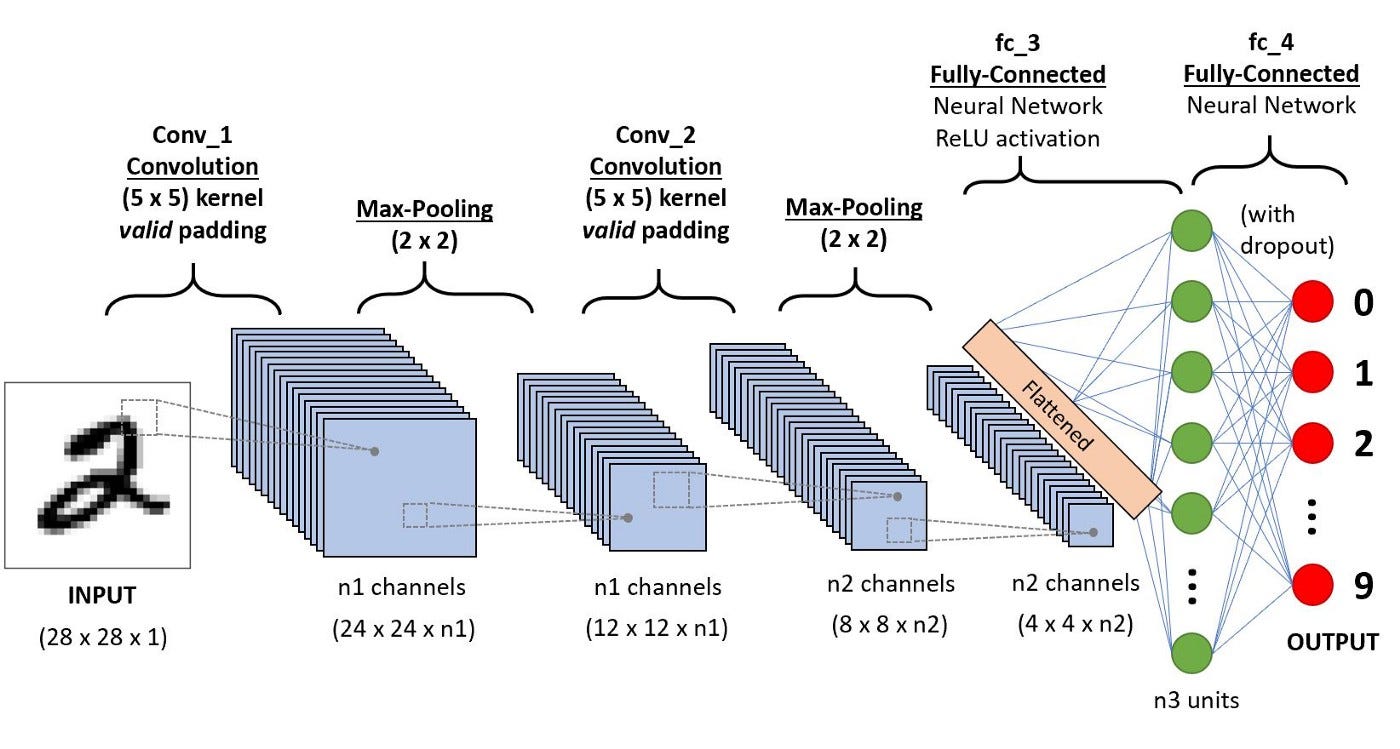
4.25 3.5

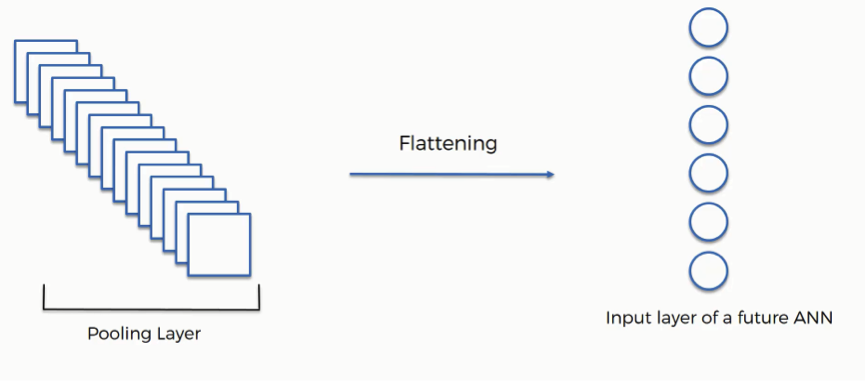
**⭐ Why Pooling is Important?**

✔ Reduces computation  
✔ Prevents overfitting  
✔ Extracts strong features  
✔ Makes model fast and stable

**⭐ 4. FLATTENING**







4

Flattening = Converting **2D feature map → 1D vector**.

Example from your slide:

Feature Map:

1 1 0

4 2 1

0 2 1

After flattening:

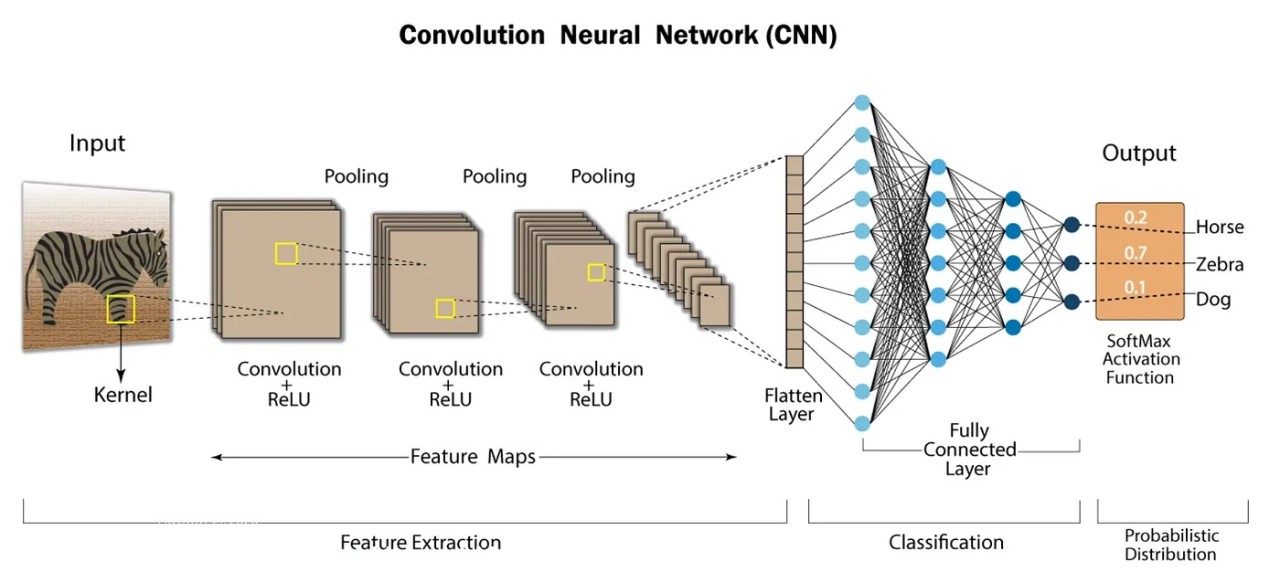
[1,1,0,4,2,1,0,2,1]

**Why flatten?**

Because **Fully Connected layers require 1D input**.

Without flattening, dense layers cannot accept data.

**⭐ 5. Fully Connected Layer (Dense Layer)**



4

Fully Connected Layers (FCL):

* Take flattened vector
* Learn high-level patterns
* Make final classification

Each neuron is connected to **every neuron** in the next layer.

Example:  
After all feature extraction, model decides:

Digit 0 → 0.02

Digit 1 → 0.01

Digit 2 → 0.15

Digit 9 → 0.95 ← highest → predicted result

Biggest drawback:

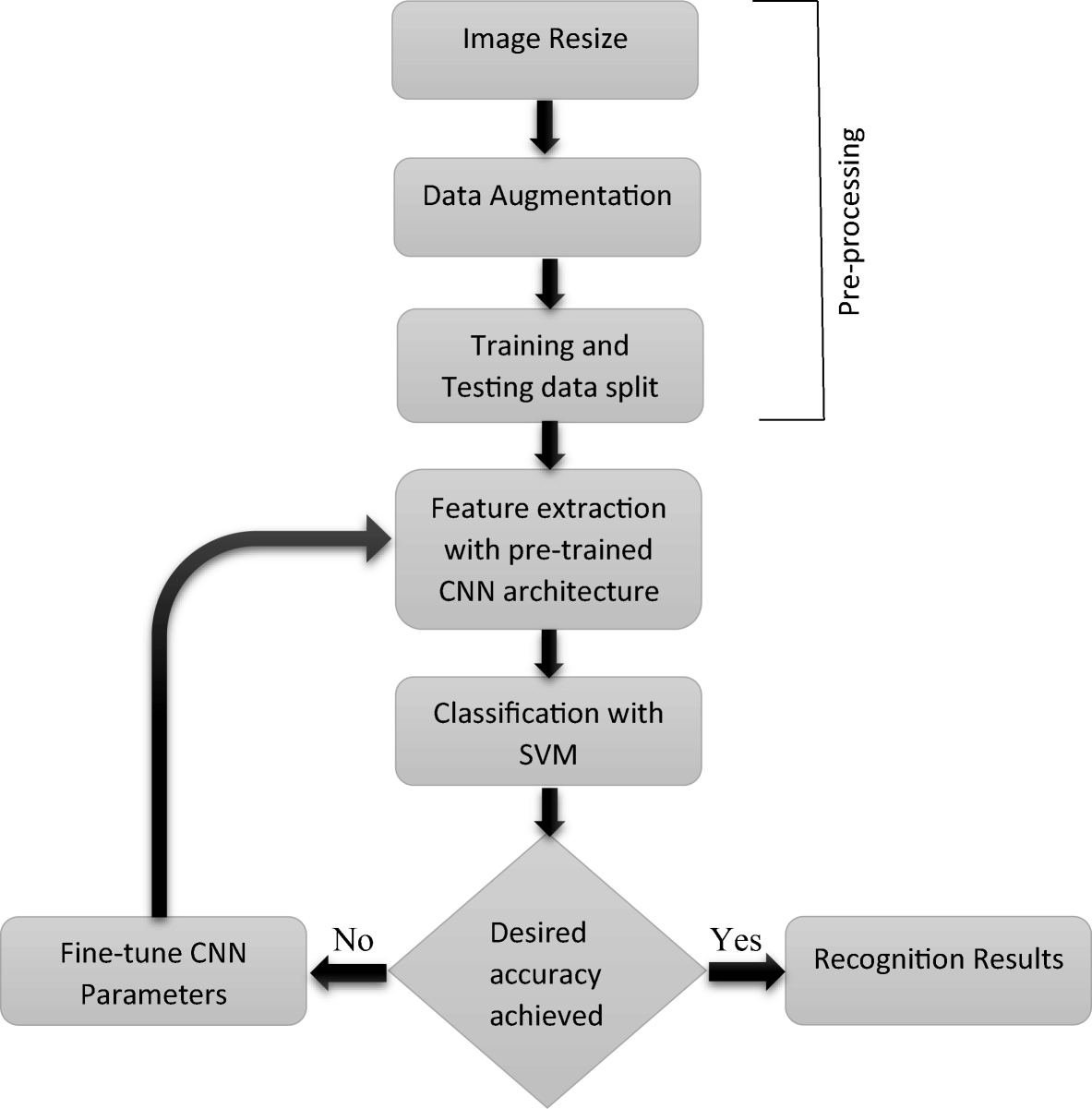
* Many parameters → **overfitting**
* Solution: **Dropout**

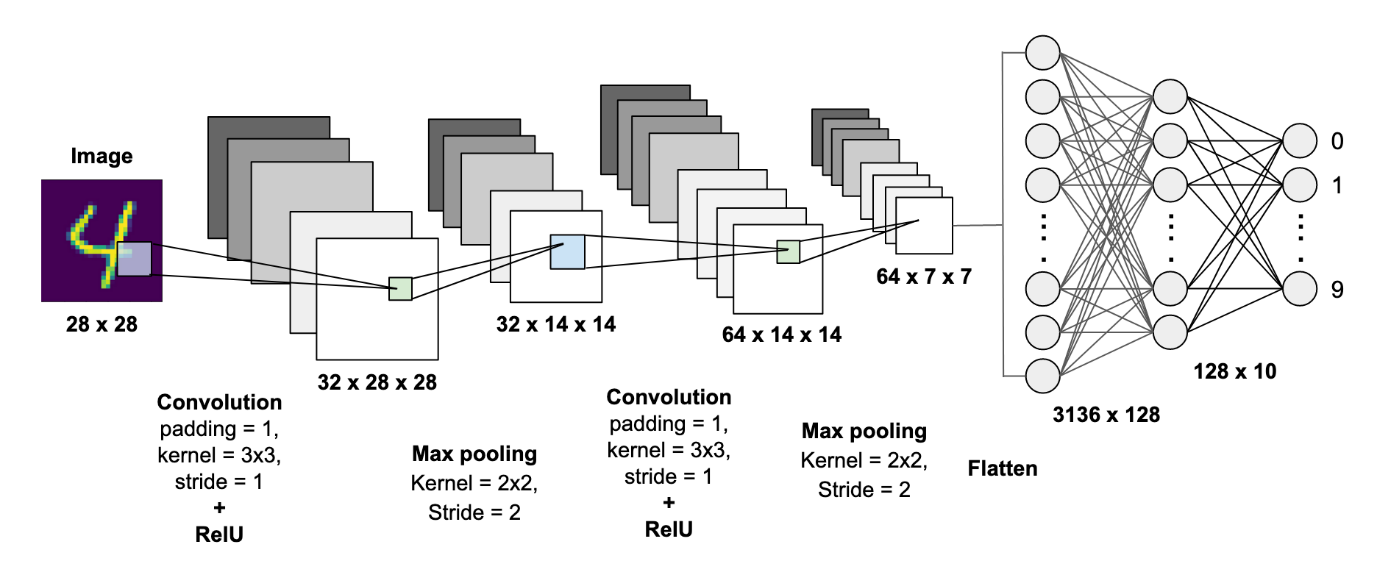
**⭐ 6. Dropout Layer**

Dropout randomly removes some neurons during training.

Benefits:  
✔ Reduces overfitting  
✔ Prevents model memorizing data  
✔ Improves generalization

**⭐ 7. How CNN Works Altogether (Full Pipeline)**





4

Your slides show:

1. **Input Image (e.g., number 7)**
2. **1st Convolution → Feature Maps**
3. **Pooling**
4. **2nd Convolution → More Features**
5. **Pooling**
6. **Flatten**
7. **Fully Connected Layers**
8. **Output → Class prediction**

This is exactly how CNN identifies digits, animals, objects, etc.

Fully connected layers cause overfitting because they contain a very large number of parameters.  
After flattening, every input value is connected to every neuron, creating millions of weights.  
This gives the model too much flexibility, causing it to memorize the training data instead of learning general features.  
Therefore FC layers are the main source of overfitting in CNNs.

**⭐ 8. Evaluation Metrics of CNN**

Used to measure model performance.

**✔ Accuracy**

% of correctly classified images

**✔ Precision**

Of all predicted positives, how many were actually positive?

📌 Example:  
Model predicted “Cat” 20 times, 18 were correct → Precision = 18/20

**✔ Recall**

Of all actual positives, how many were correctly predicted?

📌 Example:  
There are 25 cats in dataset, model found 18 → Recall = 18/25

**✔ F1 Score**

Harmonic mean of precision and recall  
Used for **imbalanced datasets**

**⭐ 9. Applications of CNN**

📌 **Image Classification**  
(cat/dog, digit recognition, cancer detection)

📌 **Object Detection**  
Detect humans, cars, animals (YOLO, SSD)

📌 **Image Segmentation**  
Medical imaging, self-driving cars  
Identify regions (road, building, sky, etc.)

📌 **Video Analysis**  
Action recognition  
Traffic monitoring  
Robotic vision

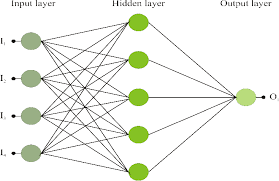
**⭐ 10. Advantages of CNN**

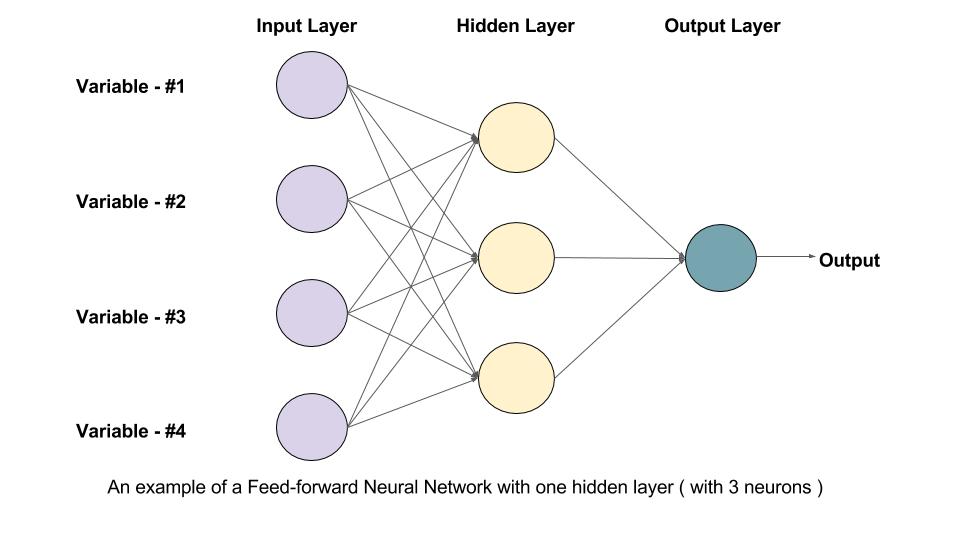
✔ **High Accuracy**  
✔ **Automatic feature extraction**  
✔ **Less parameters than ANN for images**  
✔ **Works well for complex visual tasks**  
✔ **Powerful with GPUs**

**⭐ 11. Disadvantages of CNN**

✖ Needs **large labeled datasets**  
✖ Requires high computational power  
✖ **Slow to train**  
✖ Hard to interpret (black box)  
✖ Fully connected layers cause overfitting

**⭐ PART 1 — ARTIFICIAL NEURAL NETWORK (ANN)**





4

An **Artificial Neural Network (ANN)** is inspired by the human brain.  
It consists of:

**✔ Input Layer**

Takes raw data like numbers, images, etc.

**✔ Hidden Layer(s)**

Most learning happens here.  
Each neuron uses **weights + bias + activation function**.

**✔ Output Layer**

Produces final prediction (classification/regression).

**⭐ How ANN works?**

For each neuron:

Output = activation(Wx + b)

Information flows in **one single direction** → **left → right**  
There is **NO memory**, **NO loops**, **NO time understanding**.

**⭐ Why ANN fails for sequential data?**

Because ANN treats each input **independently**.

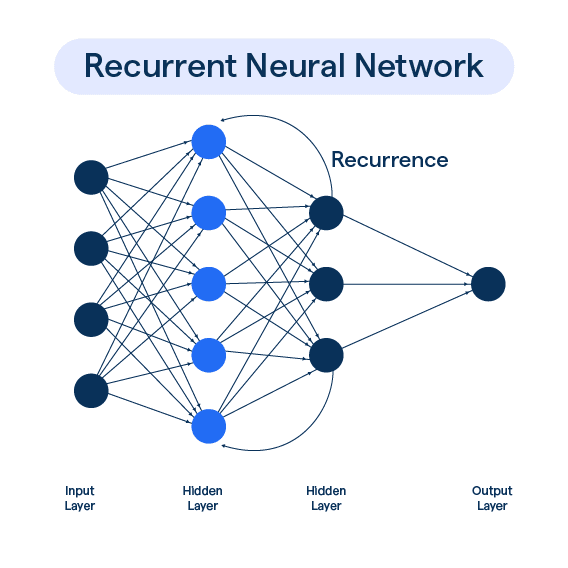
Example:  
Sentence prediction: “I love …”  
To predict the next word, the network must **remember previous words**.

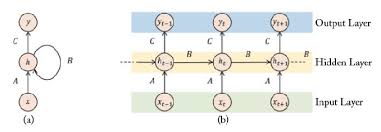
ANN **cannot remember**, hence it fails for:

* Text
* Speech
* Weather forecasting
* Stock prediction
* Sensor time-series

To solve this — **RNN was created**.

**⭐ PART 2 — RECURRENT NEURAL NETWORK (RNN)**





4

RNN = ANN + Memory  
RNN has **loops** inside the hidden layer to remember past information.

**⭐ KEY IDEA OF RNN**

At every time-step **t**, RNN takes:

* Current input → xₜ
* Previous memory → hₜ₋₁

And produces:

* Output → yₜ
* New memory → hₜ

Formula:

hₜ = f(Wx \* xₜ + Wh \* hₜ₋₁ + b)

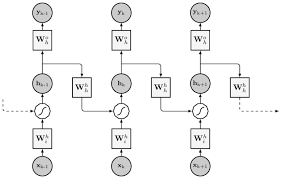
yₜ = g(Why \* hₜ)

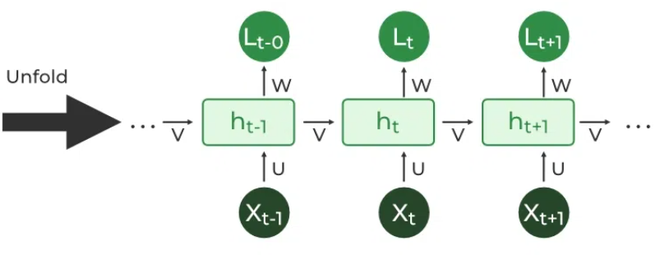
**⭐ The Hidden State (Memory)**

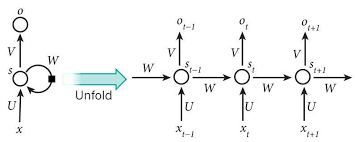
The hidden state **hₜ** is the MOST important part.  
It stores sequence information.

Example:  
If the sentence is:  
“I love playing …”  
RNN remembers “I love playing” before predicting the next word.

**⭐ PART 3 — RNN UNFOLDING (UNROLLING)**







4

The structure with a loop is difficult to understand.  
So we **unfold** it across time:

**Before Unfolding (compact representation)**

One RNN cell that loops back to itself.

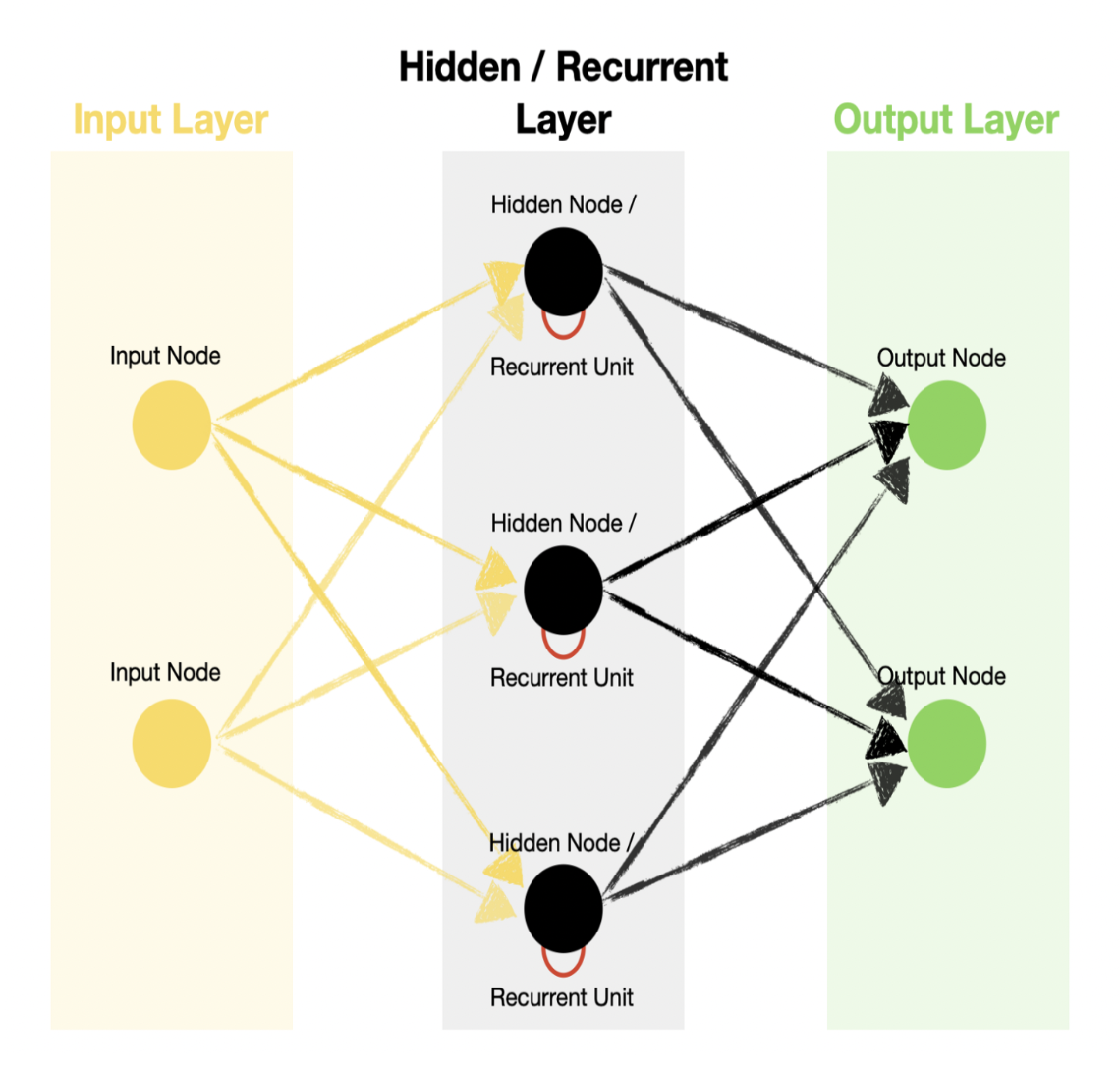
**After Unfolding (expanded)**

h₀ → h₁ → h₂ → h₃ → …  
Each step corresponds to one time input x₀, x₁, x₂, …

**⭐ Why Unfolding is Important?**

* Helps us understand how sequence flows
* Makes training possible using **Backpropagation Through Time (BPTT)**
* Shows how past hidden states affect the future ones

**⭐ PART 4 — Recurrent Neurons (Detailed Explanation)**



4

A recurrent neuron takes:

* **xₜ** (current input)
* **hₜ₋₁** (past memory)

Produces:

* **hₜ** (new memory)
* **yₜ** (output)

It has 3 weight matrices:

* **U**: weight for input → h
* **V**: weight for h → output
* **W**: weight for hₜ₋₁ → hₜ

This sharing of weights across time makes RNN powerful.

**⭐ PART 5 — How RNN Works (Step-by-Step)**

At time step **t**:

**1️⃣ Input comes: xₜ**

**2️⃣ Previous memory is used: hₜ₋₁**

**3️⃣ New memory is computed:**

hₜ = tanh(Wx \* xₜ + Wh \* hₜ₋₁)

**4️⃣ Output:**

yₜ = Why \* hₜ

**5️⃣ Next time step uses this memory.**

**⭐ PART 6 — Loss Function of RNN**

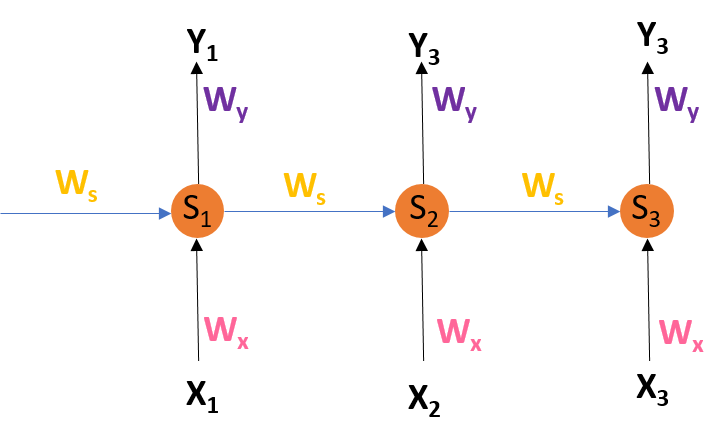
Loss is calculated **at every time step**.

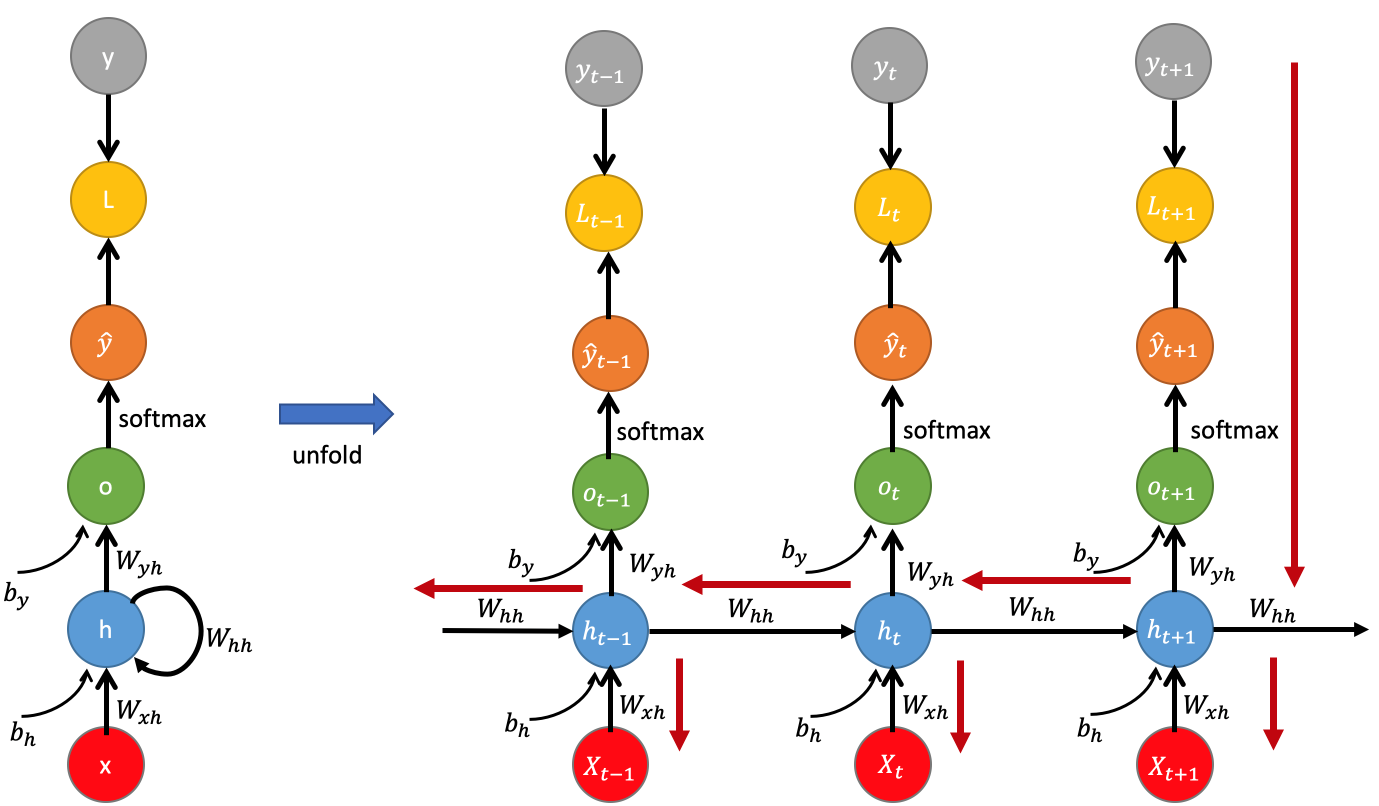
If sequence has T time steps:

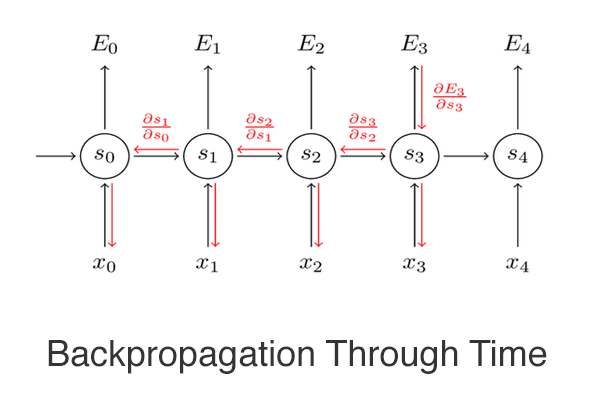
Total Loss = sum of all losses from t = 1 to T

Why?  
Because each output depends on previous hidden states.

**⭐ PART 7 — Backpropagation Through Time (BPTT)**







4

BPTT = Backpropagation for sequences.

**⭐ How it works?**

1. Unfold the RNN across time
2. Compute loss at each time-step
3. Backpropagate errors **backward through time**
4. Update weights (U, V, W)

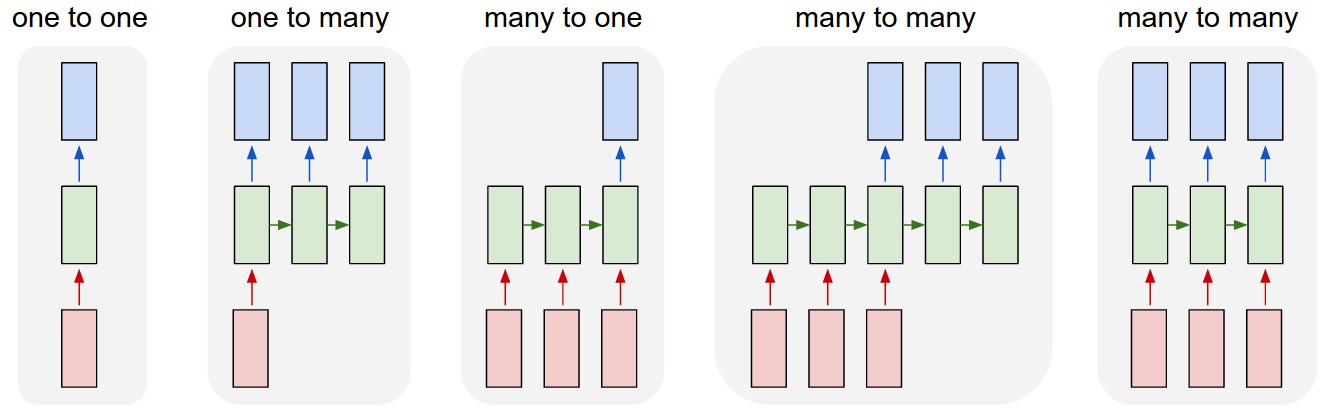
**⭐ Important:**

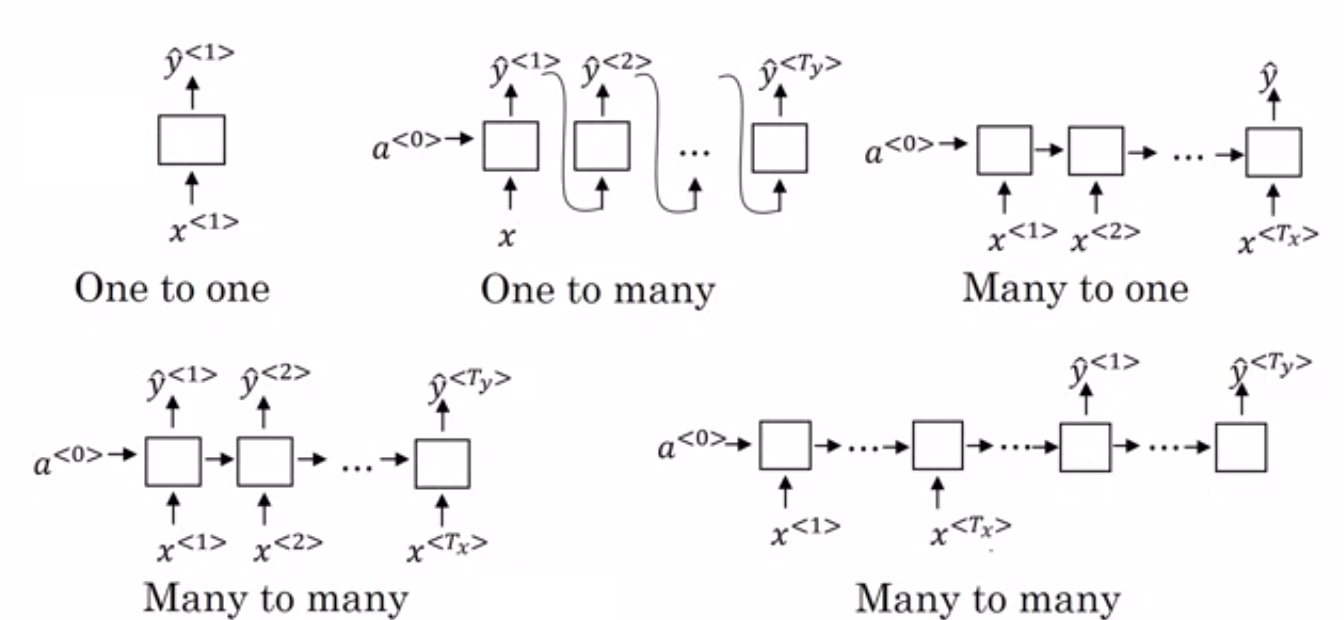
Gradients flow through many time steps → can cause:

* **Vanishing gradient**
* **Exploding gradient**

This is why RNN struggles with long sequences.

**⭐ PART 8 — Types of RNN (VERY IMPORTANT FOR EXAMS)**





4

**1️⃣ One-to-One**

Static input → static output  
Example: Normal ANN (image classification)

**2️⃣ One-to-Many**

One input → sequence output  
Example:  
Image → Caption generation (multiple words)

**3️⃣ Many-to-One**

Sequence input → one output  
Example:  
Sentiment analysis (text → positive/negative)

**4️⃣ Many-to-Many**

Sequence input → sequence output  
Example:  
Machine translation (English → French)

**⭐ PART 9 — Difference Between FNN and RNN**

| **Feature** | **FNN** | **RNN** |
| --- | --- | --- |
| Data Flow | Straight, no loops | Loops through time |
| Memory | No memory | Has hidden state |
| Inputs | Independent | Sequential |
| Architecture | Fixed layers | Unfolds over time |
| Training | Backpropagation | BPTT |
| Use Case | Images, tabular data | Text, speech, time-series |

**⭐ PART 10 — Advantages of RNN**

✔ Handles sequential data  
✔ Remembers past information  
✔ Useful for speech, text, time series  
✔ Weight sharing → fewer parameters  
✔ Can handle variable length inputs  
✔ Foundation for advanced models (LSTM, GRU)

**⭐ PART 11 — Disadvantages of RNN**

❌ Vanishing gradient  
❌ Exploding gradient  
❌ Slow training (sequential processing)  
❌ Hard to learn long-term memory  
❌ High computation cost  
❌ Difficult to tune

(This is why LSTM & GRU were invented — they solve these problems.)

**⭐ SUPER SHORT EXAM ANSWERS (Write these EXACTLY)**

**⭐ Define RNN**

RNN is a neural network designed for sequential data. It uses hidden states and recurrent connections to remember past information.

**⭐ What is RNN unfolding?**

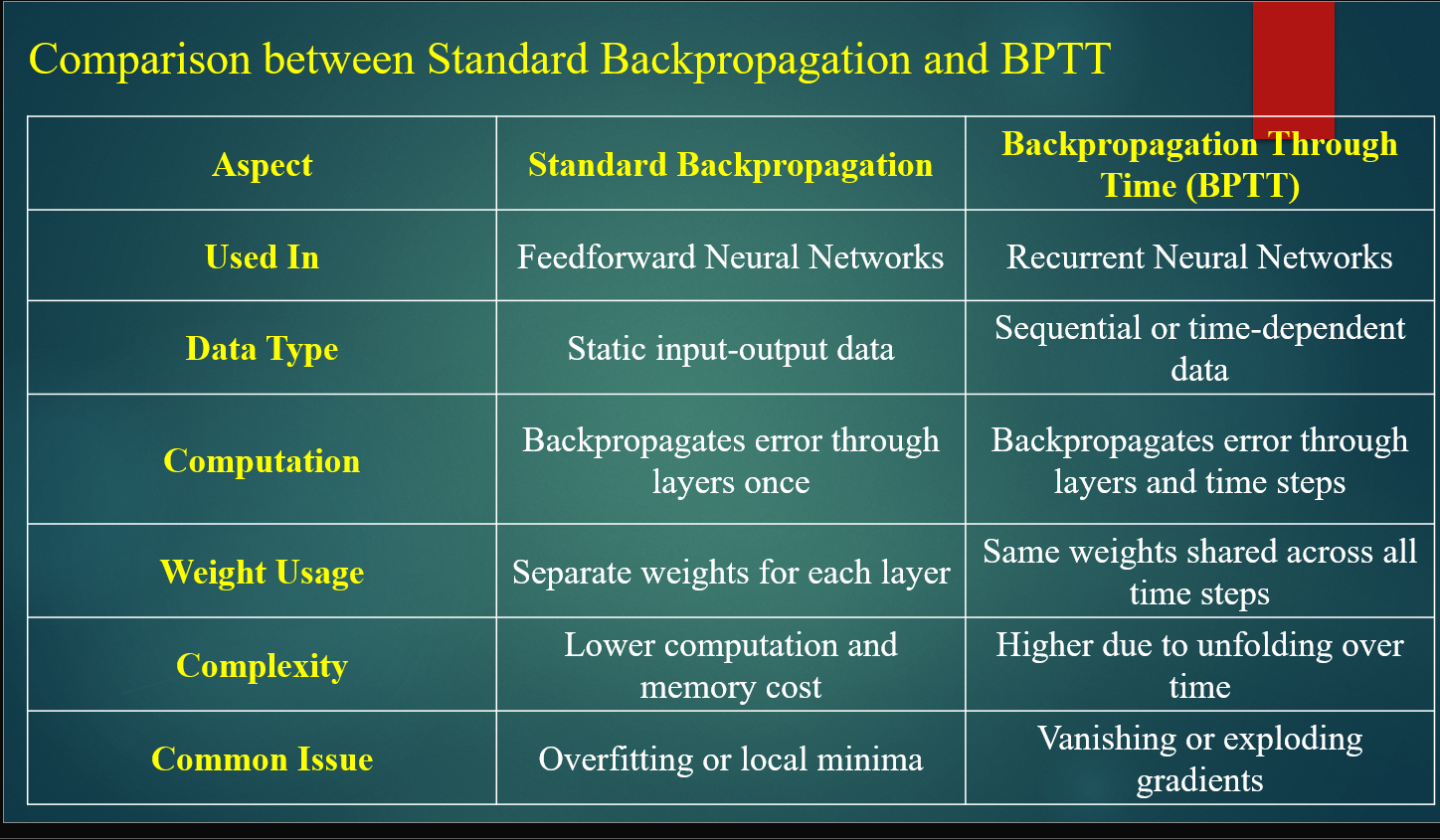
Unfolding is expanding the RNN across time steps to show how the hidden state flows and how BPTT works.

**⭐ Why do we use BPTT?**

To train RNN by backpropagating errors through time and updating weights based on all time steps.

**⭐ Applications of RNN**

Text generation, speech recognition, machine translation, sentiment analysis, stock prediction.



**✅ 1. SIGMOID FUNCTION & ITS DERIVATIVE (EXAM NOTES)**

**✅ Sigmoid Function:**

The sigmoid function is given by:

It converts any input value into the range **0 to 1**.

**✅ Uses of Sigmoid:**

* Used when output is **probability**
* Used in **binary classification**

**✅ Sigmoid Derivative:**

This derivative is used during **backpropagation to update weights**.

**✅ 2. WHY SIGMOID CAUSES VANISHING GRADIENT**

For very large positive or negative input values, the sigmoid function becomes **saturated** and its derivative becomes **very close to zero**.

During backpropagation, gradients are multiplied across layers:

If each derivative ≈ 0.2, then:

This very small value makes the weight updates extremely small. As a result, early layers almost stop learning. This problem is called the **Vanishing Gradient Problem**.

**✅ 3. VANISHING GRADIENT PROBLEM**

**✅ Definition:**

The vanishing gradient problem occurs when gradient values become extremely small during training, causing very slow or no learning in the early layers of deep neural networks.

**✅ Causes:**

* Sigmoid or tanh activation functions
* Deep neural networks
* Repeated multiplication of small gradients
* Long time steps in RNNs

**✅ Effects:**

* Early layers stop learning
* Long-term dependencies are not captured
* RNN fails to remember old information

**✅ 4. EXPLODING GRADIENT PROBLEM**

**✅ Definition:**

The exploding gradient problem occurs when gradient values become extremely large during training, causing unstable weight updates and model divergence.

**✅ Causes:**

* Large weights
* Poor weight initialization
* Long RNN sequences
* Large derivative values

**✅ Example:**

If each derivative = 5, then:

This results in:

* Huge weight updates
* Model oscillation
* Loss becomes NaN
* Training fails

**✅ 5. DIFFERENCE BETWEEN VANISHING & EXPLODING GRADIENT**

| **Vanishing Gradient** | **Exploding Gradient** |
| --- | --- |
| Gradient → 0 | Gradient → ∞ |
| Learning slows | Learning becomes unstable |
| Weights hardly update | Weights change drastically |
| No long-term memory | Training diverges |

**✅ 6. WHY BASIC RNN FAILS**

Basic RNN fails because:

* It uses **sigmoid/tanh activation functions**
* It uses **Backpropagation Through Time (BPTT)**
* Gradients pass through many time steps
* This causes **vanishing or exploding gradient problem**
* It cannot retain **long-term dependencies**

Therefore, **LSTM was developed to solve these issues**.

**✅ 7. WHAT IS LSTM (EXAM DEFINITION)**

**LSTM (Long Short-Term Memory) is an advanced type of Recurrent Neural Network designed to overcome the vanishing and exploding gradient problems of traditional RNNs. It introduces a memory cell and gating mechanism to learn long-term dependencies.**

**✅ 8. KEY FEATURES OF LSTM**

* Has **Cell State (long-term memory)**
* Has **Hidden State (short-term memory)**
* Uses **three gates**
* Controls information flow
* Can remember information for a long time

**✅ 9. LSTM CELL STRUCTURE**

| **Component** | **Purpose** |
| --- | --- |
| Cell State (Cₜ) | Long-term memory |
| Hidden State (hₜ) | Short-term memory |
| Forget Gate | Decides what to delete |
| Input Gate | Decides what to store |
| Output Gate | Decides what to output |

**✅ 10. FORGET GATE**

**✅ Formula:**

**✅ Function:**

* If → forget information
* If → keep information

It decides which past memories are useless.

**✅ 11. INPUT GATE**

**✅ Gate Decision:**

**✅ Candidate Memory:**

**✅ Cell State Update:**

This adds **new useful information** to the memory.

**✅ 12. OUTPUT GATE**

**✅ Formula:**

It decides:

* What information goes to output
* What goes to the next time-step

**✅ 13. WHY LSTM SOLVES VANISHING GRADIENT**

* Cell state follows **additive updates**
* Gradient flows smoothly
* Gates control information flow
* No repeated shrinking of gradients

Therefore, **long-term memory is preserved**.

**✅ 14. APPLICATIONS OF LSTM**

* Language Translation
* Text Generation
* Speech Recognition
* Time-Series Prediction
* Stock Price Forecasting
* Fraud Detection
* Video Analysis
* Recommender Systems

**✅ 15. FINAL 10-MARK ANSWER (DIRECT WRITE IN EXAM)**

**LSTM (Long Short-Term Memory) is an advanced type of Recurrent Neural Network designed to overcome the vanishing and exploding gradient problems of traditional RNNs. It introduces a memory cell and three gates: Forget gate, Input gate, and Output gate. The forget gate decides what past information to remove, the input gate decides what new information to store, and the output gate controls what information is passed forward. This gated mechanism allows LSTM to retain long-term dependencies effectively, making it suitable for applications such as language modeling, speech recognition, time-series forecasting, and anomaly detection.**

**✅ 16. QUICK REVISION (1-LINERS)**

* Sigmoid causes vanishing gradient due to small derivatives.
* Vanishing gradient → no learning.
* Exploding gradient → unstable learning.
* RNN fails to remember long-term information.
* LSTM solves this using memory cell and gates.
* Forget gate deletes memory.
* Input gate adds memory.
* Output gate controls output.

**✅ 1. GRU (GATED RECURRENT UNIT) — EXAM NOTES**

**✅ Definition of GRU**

A Gated Recurrent Unit (GRU) is a type of Recurrent Neural Network (RNN) that uses two gates: **Update Gate** and **Reset Gate**. It has only one hidden state and no separate cell state. GRU is simpler and faster than LSTM.

**✅ Key Features of GRU**

* Uses **two gates**
* Has **only one hidden state**
* **No separate cell state**
* **Fewer parameters**
* **Fast training**
* **Less memory consumption**

**✅ Gates in GRU**

**✅ 1. Update Gate (zₜ)**

It controls how much of the previous memory should be carried forward.

* If → More new information
* If → More old information

**✅ 2. Reset Gate (rₜ)**

It controls how much of the previous hidden state should be forgotten.

* If → Forget past
* If → Keep past

**✅ Candidate Hidden State**

**✅ Final Hidden State**

It is a controlled mixture of **old memory and new memory**.

**✅ GRU vs LSTM (EXAM TABLE)**

| **Feature** | **GRU** | **LSTM** |
| --- | --- | --- |
| Number of Gates | 2 | 3 |
| Cell State | No | Yes |
| Complexity | Low | High |
| Speed | Faster | Slower |
| Long-Term Memory | Medium | Excellent |
| Parameters | Less | More |

**✅ 2. MLOps — EXAM NOTES**

**✅ Definition of MLOps**

MLOps (Machine Learning Operations) is a set of practices that combines machine learning and DevOps to automate the complete lifecycle of machine learning models including development, training, deployment and monitoring.

**✅ Need of MLOps**

* Manual deployment is slow
* Model version tracking is difficult
* Model performance may degrade
* Data drift can occur
* Scaling becomes difficult

MLOps solves these problems using **automation, versioning, monitoring and retraining**.

**✅ Stages of MLOps**

1. Data Collection
2. Data Processing
3. Model Training
4. Model Testing
5. Model Deployment
6. Monitoring and Maintenance

**✅ MLOps vs DevOps**

| **MLOps** | **DevOps** |
| --- | --- |
| Handles ML models | Handles software |
| Data + Model versioning | Only code versioning |
| Drift monitoring | No drift monitoring |
| ML performance metrics | IT system metrics |

**✅ 3. MLFLOW — EXAM NOTES**

**✅ Definition of MLflow**

MLflow is an open-source platform used to manage the complete machine learning lifecycle including experiment tracking, model packaging, version control and deployment.

**✅ Key Features of MLflow**

1. **Experiment Tracking** – Logs parameters, metrics and models
2. **Model Packaging** – Saves trained models
3. **Model Registry** – Manages model versions
4. **Model Deployment** – Deploys models in production

**✅ 4. NLP (NATURAL LANGUAGE PROCESSING) — EXAM NOTES**

**✅ Definition of NLP**

Natural Language Processing (NLP) is a branch of Artificial Intelligence that enables computers to understand, interpret and generate human language.

**✅ Applications of NLP**

* Chatbots
* Google Translate
* Voice Assistants
* Sentiment Analysis
* Spam Filtering
* Speech Recognition

**✅ Components of NLP**

| **NLU** | **NLG** |
| --- | --- |
| Understands text | Generates text |
| Extracts meaning | Forms sentences |
| Detects emotions | Produces responses |

**✅ 5. STEPS OF NLP — EXAM NOTES**

1. **Lexical Analysis** – Splits text into words
2. **Syntactic Analysis** – Checks grammar
3. **Semantic Analysis** – Extracts meaning
4. **Discourse Analysis** – Understands references
5. **Pragmatic Analysis** – Understands real-world context

**✅ Lexical Analysis Tools**

* **Stemming:** playing → play
* **Lemmatization:** better → good

**✅ Syntactic Analysis**

It checks the grammatical structure of a sentence.

**✅ Semantic Analysis**

It extracts meaning from the sentence.

**✅ Pragmatic Analysis**

It interprets meaning based on situation and context.

**✅ 6. WORD EMBEDDING — EXAM NOTES**

**✅ Definition of Word Embedding**

Word Embedding is a technique used to convert words into numerical vectors in such a way that they capture semantic and syntactic meaning.

Example:  
king → [0.5, 0.2, −0.1]  
queen → [0.6, 0.3, −0.2]

**✅ Need of Word Embedding**

* Computers understand only numbers
* Text must be converted into numerical form
* Helps capture word similarity and meaning

**✅ Similarity Measure (Cosine Similarity)**

Used in:

* Search engines
* Recommendation systems
* Similar word detection

**✅ Advantages of Word Embedding**

* Captures semantic meaning
* Low dimensional representation
* Improves NLP model accuracy
* Fast computation

**✅ Limitations of Word Embedding**

* Static embeddings ignore context
* Bias in data
* Cannot handle new unseen words easily

**✅ FINAL ONE-LINE DEFINITIONS (FAST REVISION)**

* **GRU:** A simplified RNN with two gates and one hidden state.
* **MLOps:** A system to manage ML model lifecycle in production.
* **MLflow:** An open-source ML lifecycle management tool.
* **NLP:** A technique to enable computers to understand human language.
* **Word Embedding:** A numerical vector representation of words.

**✅ PART 1: TEXT REPRESENTATION MODELS (BEGINNER CONCEPT)**

Computer **text directly samajh nahi sakta**, usse numbers chahiye.  
Isliye hum **text ko numerical form me convert karte hain** — ise **Text Representation** kehte hain.

**✅ Types of Text Representation**

**✅ 1. Frequency-Based Methods**

1. **Bag of Words (BoW)**
2. **TF-IDF**
3. **Co-occurrence Matrix**

**✅ 2. Prediction-Based Embeddings**

1. **Word2Vec (CBOW & Skip-gram)**
2. **GloVe**
3. **FastText**

**✅ 3. Contextualized Word Embeddings**

1. **ELMo**
2. **BERT**
3. **GPT**

**✅ PART 2: FREQUENCY-BASED METHODS (EXAM NOTES)**

**✅ 1. BAG OF WORDS (BoW)**

BoW ek method hai jisme:

* Sentence ko **word count vectors** me convert kiya jata hai
* Grammar aur order ignore ho jata hai

Example:  
"I love AI"  
→ [I=1, love=1, AI=1]

✅ Advantage: Simple  
❌ Disadvantage: Meaning & context lost

**✅ 2. TF-IDF (Term Frequency – Inverse Document Frequency)**

TF-IDF word ki **importance calculate karta hai**.

✅ High TF-IDF = Important word  
✅ Used in: Search Engines, Ranking

**✅ 3. CO-OCCURRENCE MATRIX**

Word ke saath kaunsa word **kitni baar aaya**, ye matrix show karta hai.  
✅ Ye **GloVe ka base bana**.

**✅ PART 3: WORD EMBEDDING (CORE CONCEPT)**

**✅ Word Embedding Definition (EXAM)**

**Word Embedding is a technique that converts words into dense numerical vectors in such a way that semantic and syntactic meaning is preserved.**

Example:  
king → [0.5, 0.4, -0.2]  
queen → [0.6, 0.5, -0.3]

✅ Similar words → Similar vectors  
✅ Dimensionality kam hoti hai  
✅ Meaning preserved hota hai

**✅ NEED OF WORD EMBEDDING**

* Computer sirf numbers samajhta hai
* One-hot encoding me dimensions bahut zyada hoti hain
* Word Embedding:  
  ✅ Dimension kam karta hai  
  ✅ Meaning retain karta hai  
  ✅ Model ko smart banata hai

**✅ PART 4: POPULAR WORD EMBEDDING MODELS**

**✅ 1. WORD2VEC (Google)**

Word2Vec word ke **context se seekhta hai**.

**✅ Two Models:**

1. **CBOW** – Context se target word predict karta hai
2. **Skip-gram** – Target word se context predict karta hai

✅ Learns from sentences  
✅ Fast & accurate

**✅ 2. GloVe (Stanford)**

* **Global word co-occurrence** se embeddings banta hai
* Statistics + Prediction dono ka use

✅ Better global meaning capture karta hai

**✅ 3. FASTTEXT (Facebook)**

* Word ko **sub-word (character)** me tod deta hai
* Rare & new words ke liye best

Example:  
playing → play + ing

✅ OOV (Out of Vocabulary) problem solve karta hai

**✅ 4. BERT / TRANSFORMER EMBEDDINGS**

Ye **contextual embeddings** hote hain.  
Word ka meaning **sentence ke hisab se change hota hai**.

Example:  
"river bank" → bank = river side  
"money bank" → bank = financial institution

✅ Word ka meaning dynamically change hota hai

**✅ PART 5: SIMILARITY MEASURES**

**✅ COSINE SIMILARITY**

✅ Similar words find karne ke liye  
✅ Search engines  
✅ Recommendation systems

**✅ PART 6: ADVANTAGES & LIMITATIONS**

**✅ Advantages**

* Meaning capture hota hai
* Dimension kam hoti hai
* NLP models fast ho jate hain
* Semantic similarity milti hai

**✅ Limitations**

* Static embeddings context ignore karti hain
* Bias hota hai
* New word problem (OOV)

**✅ PART 7: TRANSFORMERS (BEGINNER → ADVANCED)**

**✅ Transformer Definition (EXAM)**

**A Transformer is a deep learning architecture introduced in 2017 in the paper “Attention Is All You Need” that uses self-attention instead of RNN or LSTM to process sequences in parallel.**

**✅ CORE IDEA OF TRANSFORMERS**

Transformer **Self-Attention** use karta hai:

* Har word → har word se relation dekhta hai
* Long-distance relation easily samajh jata hai

Example:  
"The cat sat on the mat because it was tired."  
"it" → "cat" ko refer karta hai

RNN/LSTM yaha fail ho jate hain, Transformer nahi.

**✅ PART 8: NEED OF TRANSFORMERS**

Problems with RNN/LSTM:

* Sequential processing (slow)
* Vanishing gradient
* Long-term dependency problem
* Parallel processing nahi hota

Transformers solve these by:

* Self-attention
* Parallel processing
* Long-range dependency capture
* Fast training

**✅ PART 9: TRANSFORMER ARCHITECTURE**

Transformer ke 2 main parts hote hain:

| **Component** | **Function** |
| --- | --- |
| Encoder | Input samajhta hai |
| Decoder | Output generate karta hai |

Each Encoder/Decoder block has:

* Self-Attention Layer
* Feed Forward Layer
* Positional Encoding

**✅ PART 10: TYPES OF TRANSFORMER MODELS**

| **Type** | **Models** | **Use** |
| --- | --- | --- |
| Encoder-only | BERT, RoBERTa | Understanding |
| Decoder-only | GPT | Text Generation |
| Encoder-Decoder | T5, BART | Translation, Summarization |

**✅ PART 11: DIFFERENCE BETWEEN BERT & GPT**

| **Feature** | **BERT** | **GPT** |
| --- | --- | --- |
| Architecture | Encoder-only | Decoder-only |
| Training | Masked Word Prediction | Next Word Prediction |
| Context | Bidirectional | Left to Right |
| Task | Language Understanding | Language Generation |
| Example | Sentiment Analysis | Chatbot, Story writing |

**✅ PART 12: APPLICATIONS OF TRANSFORMERS**

* Machine Translation
* Text Summarization
* Chatbots
* Speech Recognition
* Image Classification
* Recommendation Systems
* Text & Music Generation

**✅ FINAL QUICK REVISION (1-LINERS)**

* Word Embedding = Words → Vectors
* Word2Vec → Context-based learning
* GloVe → Global statistics
* FastText → Subword embeddings
* BERT → Context-based embeddings
* GPT → Text generation model
* Transformer → Self-attention based architecture
* Encoder → Understands input
* Decoder → Generates output

**✅ FINAL 10-MARK ANSWER (TRANSFORMER – DIRECT WRITE)**

**A Transformer is a deep learning architecture introduced in 2017 that uses self-attention instead of RNNs and LSTMs to process sequences in parallel. It consists of encoder and decoder blocks. Each block contains self-attention, feed-forward layers and positional encoding. Transformers overcome problems like vanishing gradient, slow sequential processing and long-term dependency issues. They are widely used in NLP tasks like translation, text summarization, chatbots and speech recognition. Popular transformer models include BERT, GPT, T5 and BART.**