1. Natural Intelligence

Human intelligence processes information incrementally while maintaining an internal model of what it's processing, built from past information and constantly updated as new information comes in.

1.1 Processes information incrementally

Human brain, process information sequentially over time. For example, in language comprehension, we understand words one after another in sequence. In perception we build up a scene as we scan it visually or move through it physically. This incremental processing allows for real-time decision-making, rather than waiting for all data to arrive before acting.

1.2 Maintaining an internal model of what it's processing

The brain maintains a kind of mental model or representation of the environment or situation it's dealing with. This model helps in:

- Predicting what's likely to happen next (e.g., in a conversation or while driving).
- Filling in gaps in sensory data (e.g., in noisy or low-light environments).
- Guiding attention and memory.

This model is not static; it's adaptive and context-sensitive.

1.3 Built from past information and constantly updated as new information comes in

This touches on learning and adaptation, these are hallmarks of biological intelligence. The brain uses past experiences (long-term memory) to interpret new inputs, updates its beliefs and predictions as new evidence arrives (e.g., Bayesian inference, predictive coding), and modifies its internal model to reduce prediction error and improve performance.

2. Artificial Neural Network

2.1 RNN processes sequences by iterating through the sequence elements

Unlike traditional feedforward neural networks that take a fixed-size input all at once, an RNN handles sequential data, such as time series (e.g., stock prices), text (e.g., words or characters in a sentence), audio (e.g., speech waveforms). RNNs process one element at a time (e.g., one word in a sentence), moving step by step through the sequence.

2.2 Maintaining a state containing information relative to what it has seen so far

This is the memory of the RNN. At each time step, the RNN updates its hidden state based on the current input and the previous hidden state. This allows it to carry forward context, much like how biological systems remember previous inputs while processing new ones.

2.3 RNN has an internal loop

The key architectural difference data flows in one direction, from input to output in Feedforward NN while in RNN, there is a loop in the architecture, the hidden state is fed back into the network at each time step. This internal loop lets the RNN memorize and build up context over time, like how we understand a sentence word by word.

2.4 The state of the RNN is reset between processing two different, independent sequences

Unless we are doing stateful RNNs (special cases), the hidden state is usually reset to zero between independent sequences. This ensures that no unintended memory leakage between unrelated examples. Each sequence is treated as a standalone unit (one data point). This is important during training and evaluation to maintain consistency.

2.5 The network internally loops over sequence elements

Even though we feed the whole sequence into the model, it processes it step-by-step internally, updating its state at each element. So, the data point (a sequence) is not processed all at once like in traditional NNs.

Figure 1 displays the core training loop of any neural network, including forward propagation, loss computation, and backpropagation using an optimizer. The process begins with an input sample or batch X, which enters the network. The input passes through a series of layers (e.g., fully connected layers, convolutional layers, RNN cells, etc.), each performing a data transformation. These layers have weights (parameters) that are initially random and learnable. At each layer, the data is transformed using matrix multiplication and a non-linear activation function. After the final layer, the network produces an output or prediction, denoted as $\hat{Y}$.

The prediction $\hat{Y}$ is compared with the true target value Y using a loss function (e.g., Mean Squared Error, Cross-Entropy). The loss function computes a loss score, which measures how far off the prediction is from the true target. The optimizer (e.g., SGD, Adam) uses the loss score to adjust the network's weights via backpropagation. It calculates the gradients of the loss w.r.t. each weight (i.e., how much each weight contributed to the error). Then it updates the weights to reduce the loss in future predictions.

The updated weights are fed back into the network layers. This loop repeats across many training examples (epochs) until the model achieves acceptable performance.

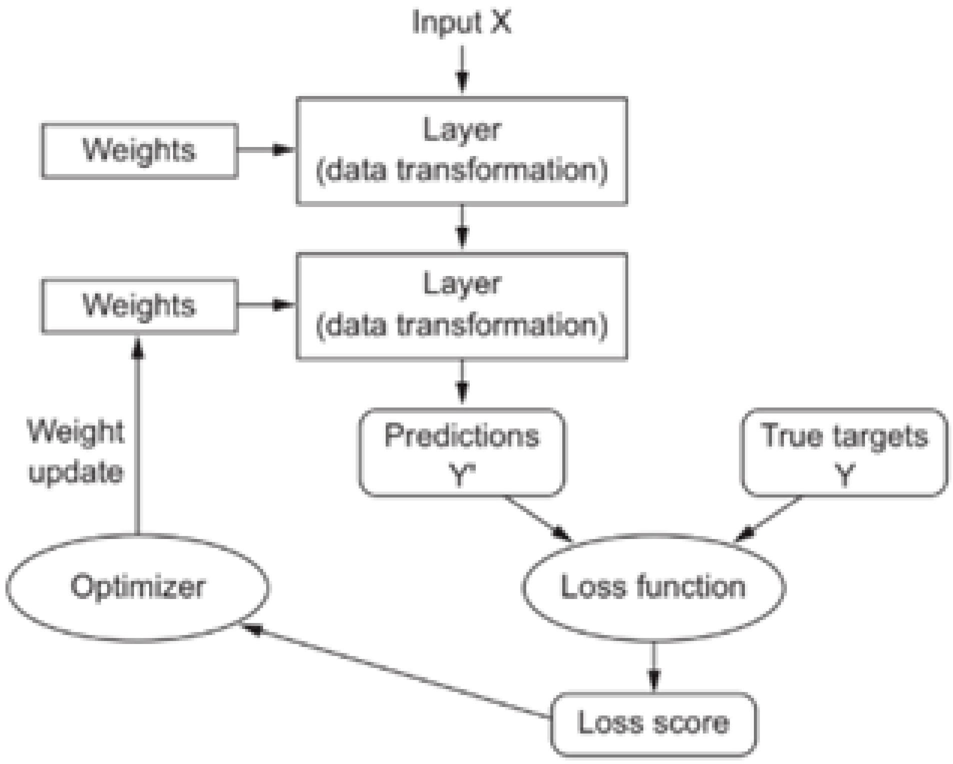

Figure 1. Training loop of a neural network, including forward propagation, loss computation, and backpropagation using an optimizer.