1. **Vanishing Gradient Problem in deep learning:**

In a deep neural network, when we do Back-propagation and calculating gradients of loss(Error) with respect to the weights , the gradients tends to get smaller and smaller as we keep on moving backward in the Network which means that the neurons in the Earlier layers learn very slowly as compared to the neurons in the later layers. This is Vanishing Gradient Problem.

Earlier layers are responsible to learn the simple patterns and if they give inaccurate results, then the next layers will also produce inaccurate results which results in too long training process and decrease of prediction accuracy.

**Solution to the problem:**

We use RELU based activation functions in training a Deep Neural Network Model to avoid such problem and improve the accuracy. For recurrent neural networks, we use LSTM(Long Short term memory) and Gated Recurrent networks to prevent such a problem.

1. **Result of convolution:**

[4 6

10 7]

1. **Pooling operation in neural network:**

Pooling is a layer in CNN that serves to progressively reduce the spatial size of the representation by reducing the number of parameters.

**Why is it important:**

It reduces the amount of computation in the network. It also helps to control overfitting. The exact location of a feature is less important than its rough location relative to other features.

1. **Gated Recurrent Network:**

GRU aims to solve vanishing gradient problem. GRU uses update gate and reset gate.  These are two vectors which decide what information should be passed to the output.

The update gate helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future. Forget gate is used from the model to decide how much of the past information to forget which is irrelevant to the prediction.

**Example: LSTM(Long short term memory) and GRU(Gated Recurrent unit) are two examples**

1. **Policy in reinforcement learning:**

A policy π is a function that takes as input a state s and returns an action a. That is: π(s) → a

The policy is typically used by the agent to decide what action a should be performed when it is in a given state s. The goal of any RL algorithm is to learn an optimal policy that achieve a specific goal. The policy can be stochastic instead of deterministic. In such a case, instead of returning a unique action a, the policy returns a probability distribution over a set of actions.

1. **Difference between SARSA and Q learning algorithm**

On-policy SARSA learns action values relative to the policy it follows, while off-policy Q-Learning does it relative to the greedy policy. Q-Learning tends to converge a little slower, but has the capabilitiy to continue learning while changing policies. In practical terms, under the ε-greedy policy, Q-Learning computes the difference between Q(s,a) and the maximum action value, while SARSA computes the difference between Q(s,a) and the weighted sum of the average action value and the maximum:

Q-Learning: Q(st+1,at+1) = maxaQ(st+1,a)

SARSA: Q(st+1,at+1) = ε·meanaQ(st+1,a) + (1-ε)·maxaQ(st+1,a)

1. **Dropout and why is it used in deep networks**

Dropout refers to ignoring units (i.e. neurons) during the training phase of certain set of neurons which is chosen at random.

It is done in deep networks to prevent over fitting. A fully connected layer occupies most of the parameters, and hence, neurons develop co-dependency amongst each other during training which curbs the individual power of each neuron leading to over-fitting of training data.

1. **Optimal Values for T-maze**

**Below two pictures are the same.**



