1. Gradient-based methods learn a parameter's value by understanding how a small change in the parameter's value will affect the network's output. If a change in the parameter's value causes a very small change in the network's output - the network just can't learn the parameter effectively, which is a problem. Vanishing gradient problem depends on the choice of the activation function. Many common activation functions (E.g. sigmoid or tanh) 'squash' their input into a very small output range in a very non-linear fashion. We can avoid this problem by using activation functions which don't have this property of 'squashing' the input space into a small region. A popular choice is Rectified Linear Unit which maps x to max(0,x).

2. When we convolve an array A= array([[0, 0, 0, 2],[2, 1, 0, 0],[2, 1, 0, 2],[2, 2, 2, 1]]) with filter B=array([[2, 0, 1],[0, 0, 0], [2, 0, 1]]) without zero padding we get C=array([[4,6],[10,7]]).

3. Spatial pooling is down sampling strategy which reduces the dimensionality of each feature map but retains important information. Pooling can be of different types, If we take the largest element from the rectified feature map within defined filter range it becomes Max pooling, etc. Pooling is very important operation because it reduces the number of parameters and operations in the network, therefore controlling overfitting. It also generalizes results from a convolutional filter -making detection of features invariant to scale or orientation.

4. Gated Recurrent Networks are used to combat the vanishing gradient problem. They have gated units i.e. memory cells (which means that they have internal state) with recurrent connection and additional neurons inside called gates. When the portion of signal arrives, the gate regulates which parts of the signal should be allowed into the unit. The gates use sigmoid activation function which takes values from 0 to 1 and acts as the filter.

LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Units) are two models for a gated recurrent network.

5. A policy defines the learning agent's way of behaving at a given time. It is a mapping from perceived states of the environment to actions to be taken when in those states. The goal of RL is to learn the best policy.

6. The biggest difference between Q Learning and SARSA is Q learning is Off policy and SARSA is on policy.
Q Learning: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\, [r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$

SARSA: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\, [r_t + \gamma \quad Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$

where $s_t, a_t, r_t, s_{t+1}, a_{t+1}$are state, action in reward at time steps t and γ is a discount factor.

SARSA is on policy because both the policy which is used for exploration and learning are the same. This is done by computing and saving a_{t+1}before updating Q.
Q learning is off policy because policy used for exploration and learning are different. a_{t+1} used for updating Q[s_t,] is different from Q[s_{t+1},a_{t+1}].

7. In order to reduce overfitting for our deep network, dropouts randomly sets hidden activations to zero. This way, hidden units in later layers cannot co-adapt to specific hidden units of previous layers; thus, overfitting is reduced. This method is called dropout.

8.
L-left, R-right, U-up, D-down
States 0-7.

states	**L**	**R**	**U**	**D**
0	**0**	**0**	**0**	**0.5**
1	**0**	**0**	**1**	**0.25**
2	**0**	**0.5**	**0.5**	**0.125**
3	**0.25**	**1**	**0**	**0**
4	**0.5**	**2**	**0**	**0**
5	**1**	**0**	**0**	**0**
6	**0**	**0**	**0.25**	**1**
7	**0**	**0**	**0.125**	**0**

References:

1. https://studywolf.wordpress.com/2013/07/01/reinforcement-learning-sarsa-vs-q-learning/
2. https://stackoverflow.com/questions/46260775/what-does-exactly-the-policy-mean-reinforcement-learning
3. https://www.quora.com/What-are-gated-units-in-recurrent-NNs
4. https://www.quora.com/What-is-the-difference-between-Q-learning-and-SARSA-learning