Task 5.1, a

The learning approach described in the article uses deep learning in convolutional neural net, called Q-network, for developing optimal $Q^*(s,a)$ action-value function for playing set of computer games in Atari 2600.
Two key ideas were introduced to update Q-learning in order to use with neural network: first one is using the pool of stored samples to take random and perform "replay" on them, second one is periodically updating target values, not always. Both are for reducing correlations: in observations and in targets. For updating was used loss function

$$L_i(\theta_i) = E_{(s,a,r,s')\approx U(D)}[(r + \gamma max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2]$$

Here θ_i are the parameters (weights of the neural net) on the $i-th$ step, γ is a discount factor (that was taken 0.99) for learning, θ_i^- the parameters that are used for obtaining target values on the $i-th$ step (they are updated every C steps), $U(D)$ is a batch of experience drawn uniformly from stored experiences (tuples (s_t, a_t, r_t, s_{t+1})).
The input for the convolutional neural net was $84 \times 84 \times 4$ image, preprocessed from the raw input from Atari 2600 and the output is a vector of $Q - values$ for each of the possible actions. Convolutional layers of Q-network here played role of feature extractor and were used for dimension reduction: instead of $4\times 84\times 84$ input nodes we got $64\times 3\times 3$ input nodes. For different games were set different sets of actions. Also, the rewards in the game were simplified to 1, 0 and -1, as the scaling was different in every game. For generating next action was used greedy policy, action is given to the framework Atari 2600 and next image and reward are returned.
In general algorithm worked in the following sequence:

1. Make an action following greedy policy
2. Get the reward and next state, save them to the pool
3. Sample transitions from the pool and set the max value to calculate loss function
4. Perform gradient descent step
5. Every C steps update weights for generating max Q values

For testing the results the greedy policy with $\epsilon = 0.05$ was also used in order to minimise the possibility of overfitting.

Task 5.1, b

States are the frames from Atari 2600, simplified slightly in order to reduce the memory and processing consumptions. So it is $84 \times 84 \times 4$ image of the current positions in the game.
Actions are possible actions in the specific game. They are set as initial values for the neural net, and represented by the output layer. So each output in the net is giving Q-value for the corresponding action.

Task 5.1, c

Parameters that are adjusted by learning are the weights θ_i of the convolutional neural net (Q-network) at $i-th$ iteration. Number of the parameters (weigths): $8 \times 8 \times 4 \times 32$ (1st hidden layer has 32 filters, and each contains four 8×8 weight matrixes - as we have four, not one frame as input) + $4 \times 4 \times 32 \times 64$ (2nd hidden layer: 64 filters, each has thirty two 4×4 matrixes, for each input (filter from previous layer)) + $3 \times 3 \times 64 \times 64$ (3rd hidden, consists of 64 filters 3×3 for 64 output from previous layer) + $64 \times 3 \times 3 \times 512$ (last hidden, that consists of 512 units, each has sixty four 3×3 matrix - 64 - number of inputs from layer before) + $512 \times ActionsNumber$ (output, $ActionsNumber$ is number of actions for the specific game).

Task 5.2

[**?**] Deep learning with neural network of form 80-40-20 was used (after experiments with other formations). The notation 80-40-20 means that the stacked autoencoders consist of three layers, where the first one receives the 175-dimensional input and computes an 80-dimensional hidden representation which is transformed into a 40-dimensional representation in the second layer until finally the third layer returns a 20-dimensional feature representation. So the number of the adjusted parameters is $175 \times 80 + 80 \times 40 + 40 \times 20$.

References

[1] *Deep Learning for Reinforcement Learning in Pacman*, http://tuprints.ulb.tu-darmstadt.de/id/eprint/3832, Bachelor-Thesis von Aaron Hochlï£¡nder aus Wiesbaden Juli 2014.