1. end to end training of the network using RL algorithms — training is fully offline so that test time only needs a single forward pass and is thus fairly fast
2. The network has two components → observation network that encodes the frame into a feature vector and a recurrent network that outputs the tracker location.
3. The appearance feature vector x_t provided by the ON is concatenated with the object location in the first frame and zeros in subsequent ones which allows the RN to jointly encode image features and location predictions and apparently makes it easier to perform regression on location
4. RN uses $o_t = f(x_t, s_t)$ to update its hidden state h_{t-1} into h_t from where the location $l_t = (x, y, w, h)$ is obtained
5. Note that all location values are relative, i.e they $\in [0, 1]$ and infact when s_t is non zero, the image patch right under it is treated as the entire image so that something can be learned about the object's appearance.
6. l_t is extracted directly from the last four elements of h_t so that the RN that represents the RL agent makes a decision that is a function of the past observations and their predicted locations.
7. During training, l_t is sampled from a Gaussian distribution with mean μ_t and a fixed variance while at test time the variance becomes zero so that $l_t = \mu_t$ or equivalently μ_t are the last four elements of h_t.
8. Two different measures of similarity between the predicted and gt locations are used for computing the reward The first one is:

$$r_1 = -\text{avg}(l - g) - \max(l - g)$$

while the other one is just the IoU → $r_2 = |l \cap g| / |l \cup g|$; r_1 is used during the earlier iterations while r_2 is used later on;
9. Both ON and RN, that are parameterized as $W = \{W_o, W_r\}$ are trained simultaneously The networks effectively learn a policy function that provides a mapping from the history of past interactions with the environment to a distribution over the actions in the current timestep.
10. Gradient is approximated using the REINFORCE algorithm where it is computed wrt log probability of the policy function after subtracting a reinforcement baseline which is taken to be the mean reward over the episode
11. An episodic version of the algorithm is used where the gradient is computed for many episodes and then a mean is taken. For each episode, the rewards in all frames are added up and only the cumulative reward is multiplied with the log of the policy function (after subtracting the baseline in each frame obviously)
12. derivative of the policy π is computed by considering that the network outputs μ and l is drawn from a distribution $g = \frac{1}{(2\pi)^{1/2}\sigma} e^{-\frac{(l-\mu)^2}{2\sigma^2}}$ so that

$$\nabla_\mu \ln \pi = \frac{\partial \ln g}{\partial \mu} = \frac{l - \mu}{\sigma^2}$$