```
FITTED-Q-LEARNING($\mathcal{A}, s_0, \gamma, \alpha, \epsilon, m$)
    $s = s_0$ // Or draw an $s$ randomly from $\mathcal{S}$
    $\mathcal{D} = \{\ \}$
    initialize neural-network representation of $Q$
    while True:
        $\mathcal{D}_{\text{new}}$ = experience from executing $\epsilon$-greedy policy based on $Q$ for $m$ steps
        $\mathcal{D} = \mathcal{D} \cup \mathcal{D}_{\text{new}}$ represented as $(s, a, r, s')$ tuples
        $D_{\text{sup}} = \{(x^{(i)}, y^{(i)})\}$ where $x^{(i)} = (s, a)$ and $y^{(i)} = r + \gamma \max_{a' \in \mathcal{A}} Q(s', a')$
            for each tuple $(s, a, r, s')^{(i)} \in \mathcal{D}$
        re-initialize neural-network representation of $Q$
        $Q = \text{supervised\_NN\_regression}(D_{\text{sup}})$
```