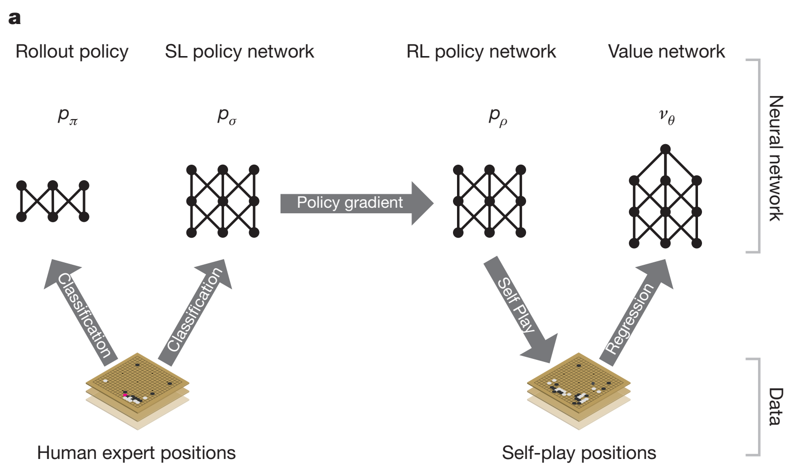


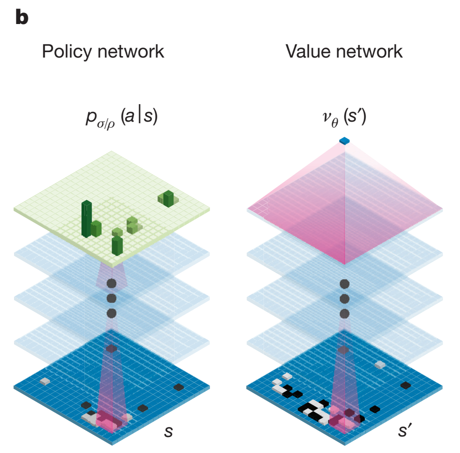


Figure 1 | Neural network training pipeline and architecture. a, A fast rollout policy p_π and supervised learning (SL) policy network p_σ are trained to predict human expert moves in a data set of positions. A reinforcement learning (RL) policy network p_ρ is initialized to the SL policy network, and is then improved by policy gradient learning to maximize the outcome (that is, winning more games) against previous versions of the policy network. A new data set is generated by playing games of self-play with the RL policy network. Finally, a value network v_θ is trained by regression to predict the expected outcome (that is, whether the current player wins) in positions from the self-play data set. **b**, Schematic representation of the neural network architecture used in AlphaGo. The policy network takes a representation of the board position s as its input, passes it through many convolutional layers with parameters σ (SL policy network) or ρ (RL policy network), and outputs a probability distribution $p_\sigma(a|s)$ or $p_\rho(a|s)$ over legal moves a, represented by a probability map over the board. The value network similarly uses many convolutional layers with parameters θ, but outputs a scalar value $v_\theta(s')$ that predicts the expected outcome in position s'.

sampled state-action pairs (s, a), using stochastic gradient ascent to maximize the likelihood of the human move a selected in state s

$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial\sigma}$$

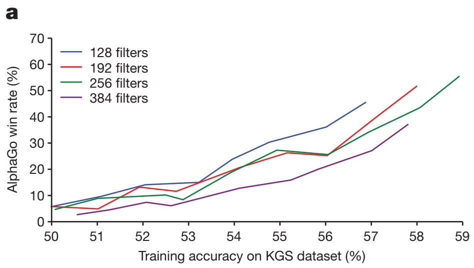
We trained a 13-layer policy network, which we call the SL policy network, from 30 million positions from the KGS Go Server. The network predicted expert moves on a held out test set with an accuracy of 57.0% using all input features, and 55.7% using only raw board position and move history as inputs, compared to the state-of-the-art from other research groups of 44.4% at date of submission[24] (full results in Extended Data Table 3). Small improvements in accuracy led to large improvements in playing strength (Fig. 2a); larger networks achieve better accuracy but are slower to evaluate during search. We also trained a faster but less accurate rollout policy $p_\pi(a|s)$, using a linear softmax of small pattern features (see Extended Data Table 4) with weights π; this achieved an accuracy of 24.2%, using just 2 μs to select an action, rather than 3 ms for the policy network.

Reinforcement learning of policy networks

The second stage of the training pipeline aims at improving the policy network by policy gradient reinforcement learning (RL)[25,26]. The RL policy network p_ρ is identical in structure to the SL policy network, and its weights ρ are initialized to the same values, $\rho = \sigma$. We play games between the current policy network p_ρ and a randomly selected previous iteration of the policy network. Randomizing from a pool of opponents in this way stabilizes training by preventing overfitting to the current policy. We use a reward function $r(s)$ that is zero for all non-terminal time steps $t < T$. The outcome $z_t = \pm r(s_T)$ is the terminal reward at the end of the game from the perspective of the current player at time step t: +1 for winning and −1 for losing. Weights are then updated at each time step t by stochastic gradient ascent in the direction that maximizes expected outcome[25]

$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t|s_t)}{\partial\rho} z_t$$

We evaluated the performance of the RL policy network in game play, sampling each move $a_t \sim p_\rho(\cdot|s_t)$ from its output probability distribution over actions. When played head-to-head, the RL policy network won more than 80% of games against the SL policy network. We also tested against the strongest open-source Go program, Pachi[14], a sophisticated Monte Carlo search program, ranked at 2 amateur *dan* on KGS, that executes 100,000 simulations per move. Using no search at all, the RL policy network won 85% of games against Pachi. In comparison, the previous state-of-the-art, based only on supervised



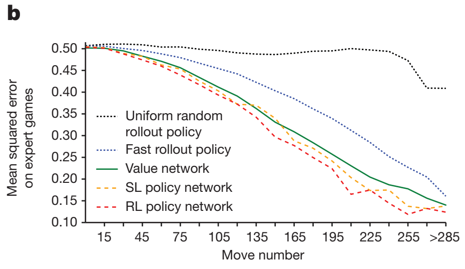


Figure 2 | Strength and accuracy of policy and value networks. a, Plot showing the playing strength of policy networks as a function of their training accuracy. Policy networks with 128, 192, 256 and 384 convolutional filters per layer were evaluated periodically during training; the plot shows the winning rate of AlphaGo using that policy network against the match version of AlphaGo. **b**, Comparison of evaluation accuracy between the value network and rollouts with different policies. Positions and outcomes were sampled from human expert games. Each position was evaluated by a single forward pass of the value network v_θ, or by the mean outcome of 100 rollouts, played out using either uniform random rollouts, the fast rollout policy p_π, the SL policy network p_σ or the RL policy network p_ρ. The mean squared error between the predicted value and the actual game outcome is plotted against the stage of the game (how many moves had been played in the given position).