Neural networks may underperform due to local optima, saddle points, overfitting, etc. More fatally, learning may not start at all.

**Goals:**
1. To ensure that networks at least get above random chance.
2. To develop simple principles for architecture and initialization.

We identify two simple reasons learning fails and prove how to avoid them.

- Poor initialization and poor architecture both stop networks from learning.

**Summary for Engineers**
- **Initialization:** Use i.i.d. weights with variance $\frac{2}{\text{fan-in}}$ (e.g. He normal).
- **Architecture:** Width (or #features in ConvNets) should grow with depth. Even a single narrow layer makes training hard.

**Summary for Mathematicians**
- We study ReLU networks at initialization, with randomly initialized weights.

**Proof approach**
1. Layer activations in feedforward neural nets form a Markov chain.
2. Squared size of layer activation vectors is an integrable submartingale. Therefore may apply Doob’s Pointwise Martingale Convergence Theorem.
3. Variance of squared size of layer activations is exponential in sum of reciprocals of hidden layer widths / residual module weights. Thus, need uniform bounds on sum of reciprocals or layer widths / residual module weights to apply Doob’s $L_p$ Martingale Convergence Theorem.