What are we about to learn?

How can a trained network reliably execute an addition algorithm?

In the neural tangent kernel (NTK) limit:

$$\underbrace{f_\infty(x)}_{\text{trained prediction}} = \underbrace{\mu(x)}_{\text{fixed mean}} + \underbrace{g(x)}_{\text{Gaussian remainder}} .$$

Choose instruction data so the mean gives the correct bit decisions.

Average independent networks so fluctuations rarely change those decisions.

Worked example: learn the updates for $10_2 + 11_2 = 101_2$.

Scope: squared loss, the converged NTK limit, and the stated mean-sign and decoding assumptions.

Outline: the three parts of the argument

We will connect training, instruction data, and reliable computation.

1. **Derive the trained prediction.**

   Gradient flow ⟶ fixed NTK ⟶ explicit solution.

   Explain why random initialization leaves a Gaussian remainder.

2. **Make the mean execute an instruction.**

   Encode input conditions in orthogonal directions.

   Use the full output labels to recover the two-bit addition update.

3. **Bound the probability of an incorrect execution.**

   Reduce variance by averaging independent trained networks.

   Choose the ensemble size to protect every predicted bit.

The endpoint is an error guarantee for fixed inputs and a fixed execution horizon, under the stated conditions.

Gradient flow is continuous gradient descent

For one output: $f_\theta(x) = m^{-1/2} W_2 \,\mathrm{ReLU}(W_1 x)$, with trainable $\theta = (W_1, W_2)$.

$x \in \mathbb{R}^d$, $W_1 \in \mathbb{R}^{m \times d}$, $W_2 \in \mathbb{R}^{1 \times m}$. m is the number of hidden neurons.

On fixed training data (X, y), follow the squared-loss gradient:

$$L(\theta) = \frac{1}{2} \| f_\theta(X) - y \|^2, \qquad \frac{d\theta_t}{dt} = -\nabla_\theta L(\theta_t).$$

Squared loss. Full-batch training. Lee et al., NeurIPS 2019, Section 2.1.

Gradient flow is continuous gradient descent

For one output: $f_\theta(x) = m^{-1/2} W_2 \,\mathrm{ReLU}(W_1 x)$, with trainable $\theta = (W_1, W_2)$.

$x \in \mathbb{R}^d$, $W_1 \in \mathbb{R}^{m \times d}$, $W_2 \in \mathbb{R}^{1 \times m}$. m is the number of hidden neurons.

On fixed training data (X, y), follow the squared-loss gradient:

$$L(\theta) = \frac{1}{2}\| f_\theta(X) - y \|^2, \qquad \frac{d\theta_t}{dt} = -\nabla_\theta L(\theta_t).$$

Gradient descent takes one downhill step at a time:

$$\theta^{k+1} = \theta^k - \eta \nabla_\theta L(\theta^k), \qquad t_k = k\eta.$$

Squared loss. Full-batch training. Lee et al., NeurIPS 2019, Section 2.1.

Gradient flow is continuous gradient descent

For one output: $f_\theta(x) = m^{-1/2} W_2 \,\mathrm{ReLU}(W_1 x)$, with trainable $\theta = (W_1, W_2)$.

$x \in \mathbb{R}^d$, $W_1 \in \mathbb{R}^{m \times d}$, $W_2 \in \mathbb{R}^{1 \times m}$. m is the number of hidden neurons.

On fixed training data (X, y), follow the squared-loss gradient:

$$L(\theta) = \frac{1}{2}\| f_\theta(X) - y \|^2, \qquad \frac{d\theta_t}{dt} = -\nabla_\theta L(\theta_t).$$

Gradient descent takes one downhill step at a time:

$$\theta^{k+1} = \theta^k - \eta \nabla_\theta L(\theta^k), \qquad t_k = k\eta.$$

$$\frac{\theta^{k+1} - \theta^k}{\eta} \xrightarrow[\eta \to 0]{\text{smaller time steps}} \frac{d\theta_t}{dt}.$$

Smaller steps, taken more often, trace a continuous downhill path.

Squared loss. Full-batch training. Lee et al., NeurIPS 2019, Section 2.1.

How do the predictions change?

Write $f_t = f_{\theta_t}$. Apply the chain rule, then substitute gradient flow:

$$\frac{d f_t(x)}{dt} = \nabla_\theta f_t(x)^\top \frac{d\theta_t}{dt}$$

Exact at finite width. Chain rule and squared loss. Lee et al., NeurIPS 2019, Eqs. (2) to (4).

How do the predictions change?

Write $f_t = f_{\theta_t}$. Apply the chain rule, then substitute gradient flow:

$$\frac{d f_t(x)}{dt} = \nabla_\theta f_t(x)^\top \frac{d\theta_t}{dt} \overset{\text{gradient flow}}{=} -\nabla_\theta f_t(x)^\top \nabla_\theta L(\theta_t).$$

Exact at finite width. Chain rule and squared loss. Lee et al., NeurIPS 2019, Eqs. (2) to (4).

How do the predictions change?

Write $f_t = f_{\theta_t}$. Apply the chain rule, then substitute gradient flow:

$$\frac{d f_t(x)}{dt} = \nabla_\theta f_t(x)^\top \frac{d\theta_t}{dt} \overset{\text{gradient flow}}{=} -\nabla_\theta f_t(x)^\top \nabla_\theta L(\theta_t).$$

For squared loss, the remaining gradient is

$$\nabla_\theta L(\theta_t) = \sum_i \big(f_t(x_i) - y_i\big) \nabla_\theta f_t(x_i).$$

Exact at finite width. Chain rule and squared loss. Lee et al., NeurIPS 2019, Eqs. (2) to (4).

How do the predictions change?

Write $f_t = f_{\theta_t}$. Apply the chain rule, then substitute gradient flow:

$$\frac{d f_t(x)}{dt} = \nabla_\theta f_t(x)^\top \frac{d\theta_t}{dt} \overset{\text{gradient flow}}{=} -\nabla_\theta f_t(x)^\top \nabla_\theta L(\theta_t).$$

For squared loss, the remaining gradient is

$$\nabla_\theta L(\theta_t) = \sum\nolimits_i \big(f_t(x_i) - y_i\big) \nabla_\theta f_t(x_i).$$

$$\frac{d f_t(x)}{dt} = -\Theta_t(x, X)\Big(f_t(X) - y\Big).$$

$$\Theta_t(u, v) = \Big\langle \nabla_\theta f_t(u), \nabla_\theta f_t(v) \Big\rangle \quad \text{(neural tangent kernel).}$$

Exact at finite width. Chain rule and squared loss. Lee et al., NeurIPS 2019, Eqs. (2) to (4).

Where infinite width enters the calculation

On the training data, set $r_t = f_t(X) - y$. The equation becomes

$$\frac{dr_t}{dt} = -\Theta_t(X, X)\, r_t.$$

This coefficient depends on the unknown, changing weights.

NTK regime. Averaging at initialization and stability during training. K positive definite. Jacot et al., 2018. Lee et al., 2019.

Where infinite width enters the calculation

On the training data, set $r_t = f_t(X) - y$. The equation becomes

$$\frac{dr_t}{dt} = -\Theta_t(X, X)\, r_t.$$

This coefficient depends on the unknown, changing weights.

By the law of large numbers and NTK stability:

$$\Theta_t(u, v) \xrightarrow{m \to \infty} \Theta(u, v).$$

Θ is non-random and constant in training time t.

NTK regime. Averaging at initialization and stability during training. K positive definite. Jacot et al., 2018. Lee et al., 2019.

Where infinite width enters the calculation

On the training data, set $r_t = f_t(X) - y$. The equation becomes

$$\frac{dr_t}{dt} = -\Theta_t(X, X)\, r_t.$$

This coefficient depends on the unknown, changing weights.

By the law of large numbers and NTK stability:

$$\Theta_t(u, v) \xrightarrow{m \to \infty} \Theta(u, v).$$

Θ is non-random and constant in training time t.

$$\frac{dr_t}{dt} = -K r_t \qquad\qquad K = \Theta(X, X).$$

NTK regime. Averaging at initialization and stability during training. K positive definite. Jacot et al., 2018. Lee et al., 2019.

Where infinite width enters the calculation

On the training data, set $r_t = f_t(X) - y$. The equation becomes

$$\frac{dr_t}{dt} = -\Theta_t(X, X)\, r_t.$$

This coefficient depends on the unknown, changing weights.

By the law of large numbers and NTK stability:

$$\Theta_t(u, v) \xrightarrow{m \to \infty} \Theta(u, v).$$

Θ is non-random and constant in training time t.

$$\frac{dr_t}{dt} = -Kr_t \quad \Longrightarrow \quad r_t = e^{-Kt} r_0, \qquad K = \Theta(X, X).$$

NTK regime. Averaging at initialization and stability during training. K positive definite. Jacot et al., 2018. Lee et al., 2019.

Integrate to obtain the prediction

Insert $r_t = e^{-Kt} r_0$ into the prediction equation:

$$\frac{d f_t(x)}{dt} = -\Theta(x, X) e^{-Kt} r_0.$$

Integrate to obtain the prediction

Insert $r_t = e^{-Kt} r_0$ into the prediction equation:

$$\frac{d f_t(x)}{dt} = -\Theta(x, X) e^{-Kt} r_0.$$

Integrate both sides from time 0 to time t:

$$f_t(x) - f_0(x) = -\underbrace{\Theta(x, X) \int_0^t e^{-Ks}\, ds}_{h_t(x)} r_0.$$

Integrate to obtain the prediction

Insert $r_t = e^{-Kt} r_0$ into the prediction equation:

$$\frac{d f_t(x)}{dt} = -\Theta(x, X) e^{-Kt} r_0.$$

Integrate both sides from time 0 to time t:

$$f_t(x) - f_0(x) = -\underbrace{\Theta(x, X) \int_0^t e^{-Ks}\, ds}_{h_t(x)}\, r_0.$$

$$f_t(x) = f_0(x) + h_t(x)\Big(y - f_0(X)\Big).$$

$h_t(x)$ depends on the inputs and time, not on the random initialization.

Use $r_0 = f_0(X) - y$. Lee et al., NeurIPS 2019, Eqs. (9) to (11).

Evaluate the integral: an explicit prediction

The matrix $K = \Theta(X, X)$ is fixed and positive definite, so K^{-1} exists.

$$\frac{d}{ds}\left(-K^{-1}e^{-Ks}\right) = e^{-Ks}.$$

I is the identity matrix. Infinite-width NTK, squared loss. Evaluation of the preceding integral. Lee et al., 2019, Eqs. (9) to (11).

Evaluate the integral: an explicit prediction

The matrix $K = \Theta(X, X)$ is fixed and positive definite, so K^{-1} exists.

$$\frac{d}{ds}\left(-K^{-1}e^{-Ks}\right) = e^{-Ks}.$$

$$\int_0^t e^{-Ks}\,ds = \left[-K^{-1}e^{-Ks}\right]_{s=0}^{s=t} = K^{-1}\left(I - e^{-Kt}\right).$$

I is the identity matrix. Infinite-width NTK, squared loss. Evaluation of the preceding integral. Lee et al., 2019, Eqs. (9) to (11).

Evaluate the integral: an explicit prediction

The matrix $K = \Theta(X, X)$ is fixed and positive definite, so K^{-1} exists.

$$\frac{d}{ds}\left(-K^{-1}e^{-Ks}\right) = e^{-Ks}.$$

$$\int_0^t e^{-Ks}\,ds = \left[-K^{-1}e^{-Ks}\right]_{s=0}^{s=t} = K^{-1}\left(I - e^{-Kt}\right).$$

Substitute into the prediction from the previous slide:

$$f_t(x) = f_0(x) + \underbrace{\Theta(x, X)K^{-1}\left(I - e^{-Kt}\right)}_{h_t(x)}\left(y - f_0(X)\right).$$

I is the identity matrix. Infinite-width NTK, squared loss. Evaluation of the preceding integral. Lee et al., 2019, Eqs. (9) to (11).

Evaluate the integral: an explicit prediction

The matrix $K = \Theta(X, X)$ is fixed and positive definite, so K^{-1} exists.

$$\frac{d}{ds}\left(-K^{-1}e^{-Ks}\right) = e^{-Ks}.$$

$$\int_0^t e^{-Ks}\,ds = \left[-K^{-1}e^{-Ks}\right]_{s=0}^{s=t} = K^{-1}\left(I - e^{-Kt}\right).$$

Substitute into the prediction from the previous slide:

$$f_t(x) = f_0(x) + \underbrace{\Theta(x, X)K^{-1}\left(I - e^{-Kt}\right)}_{h_t(x)}\left(y - f_0(X)\right).$$

$$t \to \infty : \quad e^{-Kt} \to 0, \qquad f_\infty(x) = f_0(x) + \Theta(x, X)K^{-1}\left(y - f_0(X)\right).$$

I is the identity matrix. Infinite-width NTK, squared loss. Evaluation of the preceding integral. Lee et al., 2019, Eqs. (9) to (11).

Regroup the trained prediction

Write $h(x) = \lim_{t\to\infty} h_t(x) = \Theta(x, X)K^{-1}$. The previous result is

$$f_\infty(x) = f_0(x) + h(x)\Big(y - f_0(X)\Big).$$

Algebraic regrouping of the preceding solution. Fixed data, deterministic limiting NTK, and K positive definite.

Regroup the trained prediction

Write $h(x) = \lim_{t\to\infty} h_t(x) = \Theta(x, X)K^{-1}$. The previous result is

$$f_\infty(x) = f_0(x) + h(x)\Big(y - f_0(X)\Big).$$

$$f_\infty(x) = \underbrace{h(x)y}_{\text{fixed part}} + \underbrace{\Big[f_0(x) - h(x)f_0(X)\Big]}_{g(x)\text{: depends on initialization}}$$

Algebraic regrouping of the preceding solution. Fixed data, deterministic limiting NTK, and K positive definite.

Regroup the trained prediction

Write $h(x) = \lim_{t\to\infty} h_t(x) = \Theta(x, X)K^{-1}$. The previous result is

$$f_\infty(x) = f_0(x) + h(x)\Big(y - f_0(X)\Big).$$

$$f_\infty(x) = \underbrace{h(x)y}_{\text{fixed part}} + \underbrace{\Big[f_0(x) - h(x)f_0(X)\Big]}_{g(x)\text{: depends on initialization}}$$

Evaluate at all training inputs X:

$$h(X) = \Theta(X, X)K^{-1} = KK^{-1} = I.$$

Algebraic regrouping of the preceding solution. Fixed data, deterministic limiting NTK, and K positive definite.

Regroup the trained prediction

Write $h(x) = \lim_{t\to\infty} h_t(x) = \Theta(x, X)K^{-1}$. The previous result is

$$f_\infty(x) = f_0(x) + h(x)\Big(y - f_0(X)\Big).$$

$$f_\infty(x) = \underbrace{h(x)y}_{\text{fixed part}} + \underbrace{\Big[f_0(x) - h(x)f_0(X)\Big]}_{g(x)\text{: depends on initialization}}$$

Evaluate at all training inputs X:

$$h(X) = \Theta(X, X)K^{-1} = KK^{-1} = I.$$

$$f_\infty(X) = y + \Big[f_0(X) - f_0(X)\Big] = y.$$

The initial predictions cancel, so $g(X) = 0$.

Algebraic regrouping of the preceding solution. Fixed data, deterministic limiting NTK, and K positive definite.

Why is the remaining term Gaussian?

The coefficients $1, -h_1(x), \ldots, -h_n(x)$ are non-random.

$$g(x) = f_0(x) - h_1(x) f_0(x_1) - \cdots - h_n(x) f_0(x_n).$$

C is the initial covariance. The outputs can be correlated. The $1/\sqrt{m}$ scaling is in the model. Lee et al., 2019, Sections 2.3 and 2.4.

Why is the remaining term Gaussian?

The coefficients $1, -h_1(x), \ldots, -h_n(x)$ are non-random.

$$g(x) = f_0(x) - h_1(x) f_0(x_1) - \cdots - h_n(x) f_0(x_n).$$

Initialize $\theta_0 \sim \mathcal{N}(0, I)$. Each f_0 sums independent neuron contributions.

C is the initial covariance. The outputs can be correlated. The $1/\sqrt{m}$ scaling is in the model. Lee et al., 2019, Sections 2.3 and 2.4.

Why is the remaining term Gaussian?

The coefficients $1, -h_1(x), \ldots, -h_n(x)$ are non-random.

$$g(x) = f_0(x) - h_1(x) f_0(x_1) - \cdots - h_n(x) f_0(x_n).$$

Initialize $\theta_0 \sim \mathcal{N}(0, I)$. Each f_0 sums independent neuron contributions.

Central limit theorem, as $m \to \infty$: the initial outputs form a Gaussian vector.

$$\big(f_0(x), f_0(x_1), \ldots, f_0(x_n)\big)^\top \sim \mathcal{N}(0, C).$$

C is the initial covariance. The outputs can be correlated. The $1/\sqrt{m}$ scaling is in the model. Lee et al., 2019, Sections 2.3 and 2.4.

Why is the remaining term Gaussian?

The coefficients $1, -h_1(x), \ldots, -h_n(x)$ are non-random.

$$g(x) = f_0(x) - h_1(x) f_0(x_1) - \cdots - h_n(x) f_0(x_n).$$

Initialize $\theta_0 \sim \mathcal{N}(0, I)$. Each f_0 sums independent neuron contributions.

Central limit theorem, as $m \to \infty$: the initial outputs form a Gaussian vector.

$$\big(f_0(x), f_0(x_1), \ldots, f_0(x_n)\big)^\top \sim \mathcal{N}(0, C).$$

The fixed weighted sum in the blue box is therefore Gaussian:

$$g(x) \sim \mathcal{N}\big(0, \Sigma(x)\big), \qquad \Sigma(x) = \mathrm{Var}\big(g(x)\big).$$

C is the initial covariance. The outputs can be correlated. The $1/\sqrt{m}$ scaling is in the model. Lee et al., 2019, Sections 2.3 and 2.4.

Corollary: the trained prediction is Gaussian

We have proved $g(x) \sim \mathcal{N}(0, \Sigma(x))$. Now add the fixed part:

$$f_\infty(x) = \underbrace{h(x)y}_{\text{fixed}} + \underbrace{g(x)}_{\text{Gaussian, mean zero}}$$

Randomness is initialization. Exact in the NTK limit. Approximate at finite width. Lee et al., NeurIPS 2019.

Corollary: the trained prediction is Gaussian

We have proved $g(x) \sim \mathcal{N}(0, \Sigma(x))$. Now add the fixed part:

$$f_\infty(x) = \underbrace{h(x)y}_{\text{fixed}} + \underbrace{g(x)}_{\text{Gaussian, mean zero}}$$

Adding $h(x)y$ shifts the mean and leaves the variance unchanged.

$$f_\infty(x) \sim \mathcal{N}\Big(\mu(x), \Sigma(x)\Big).$$
$$\mu(x) = h(x)y, \qquad \Sigma(x) = \mathrm{Var}\Big(g(x)\Big).$$

Randomness is initialization. Exact in the NTK limit. Approximate at finite width. Lee et al., NeurIPS 2019.

Corollary: the trained prediction is Gaussian

We have proved $g(x) \sim \mathcal{N}(0, \Sigma(x))$. Now add the fixed part:

$$f_\infty(x) = \underbrace{h(x)y}_{\text{fixed}} + \underbrace{g(x)}_{\text{Gaussian, mean zero}}$$

Adding $h(x)y$ shifts the mean and leaves the variance unchanged.

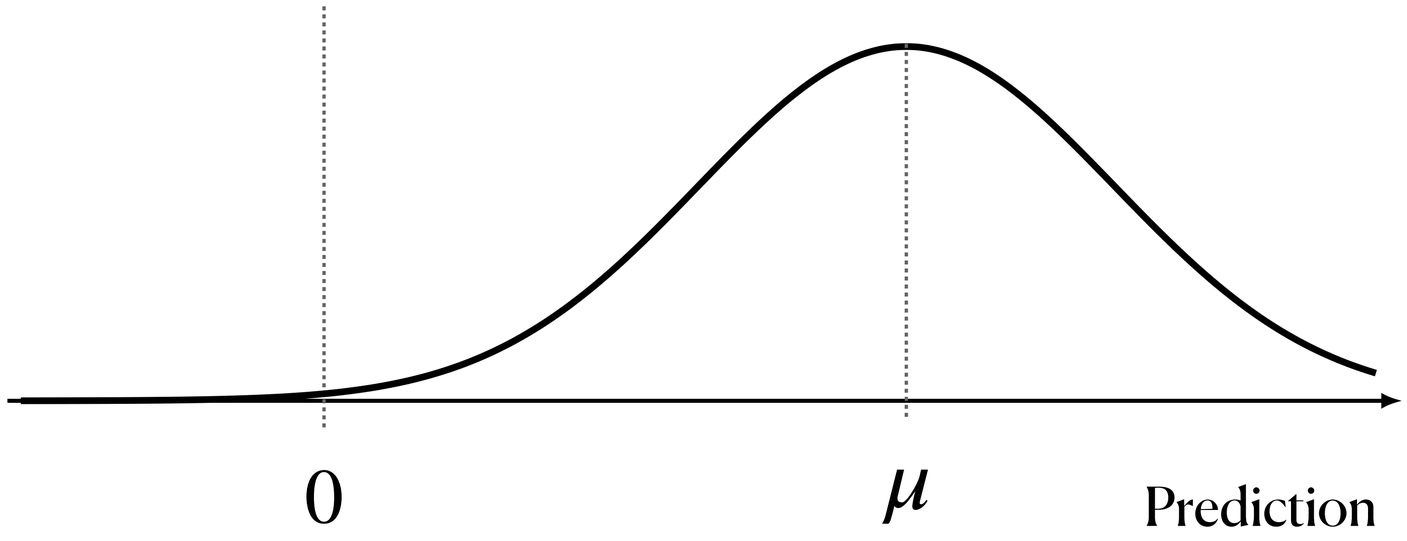
$$f_\infty(x) \sim \mathcal{N}\Big(\mu(x), \Sigma(x)\Big).$$
$$\mu(x) = h(x)y, \qquad \Sigma(x) = \mathrm{Var}\Big(g(x)\Big).$$

At training inputs, $\Sigma(x_i) = 0$. At new inputs, the variance can remain.

Randomness is initialization. Exact in the NTK limit. Approximate at finite width. Lee et al., NeurIPS 2019.

A Gaussian prediction is not enough

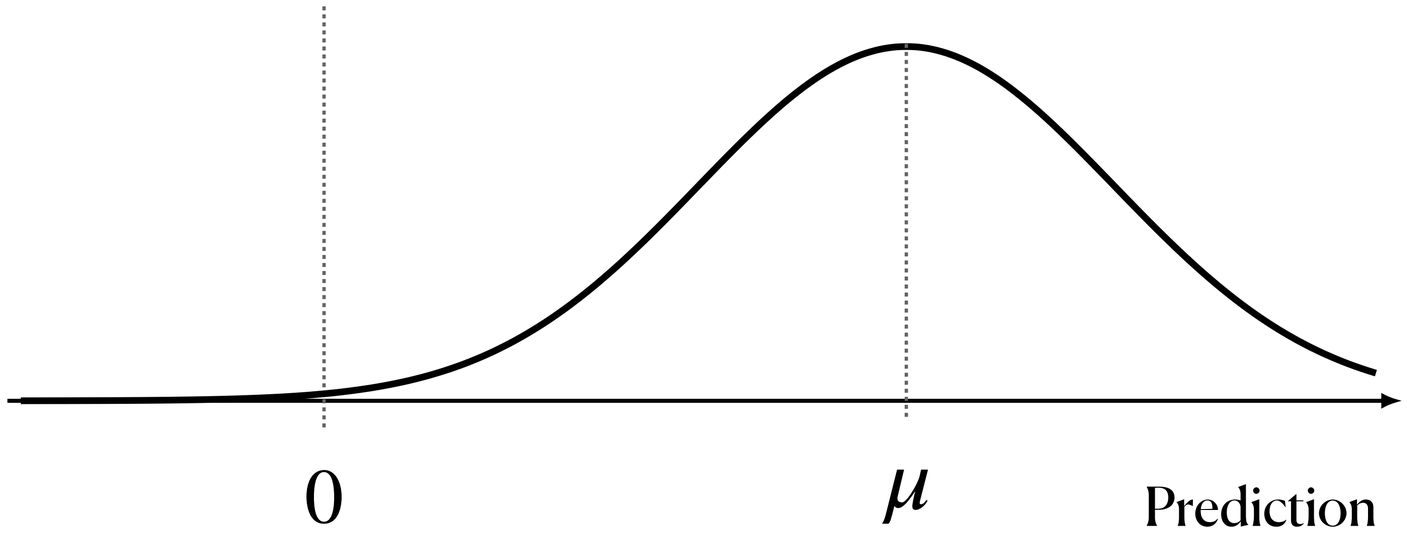
For a bit that should be 1, the prediction must stay above zero.



Paper, Sections 5 and 6. One output coordinate. Schematic distribution. Randomness is over initialization.

A Gaussian prediction is not enough

For a bit that should be 1, the prediction must stay above zero.



The mean:

$$\mu > 0$$

The spread:

$$\sigma \ll |\mu|$$

The data control the mean. Averaging controls the remaining spread.

Paper, Sections 5 and 6. One output coordinate. Schematic distribution. Randomness is over initialization.

The two-bit state has four blocks

Follow the paper's example: $p = 10_2$, $q = 11_2$, with both carries initially zero.

	carry 2	p_2	q_2	carry 1	p_1	q_1
x	0	1	1	0	0	1

$$x = (0 \mid 1\ 1 \mid 0 \mid 0\ 1)^{\top} \in \mathbb{R}^6.$$

Two blocks hold summand bits. Two blocks hold carries.

Paper, Figure 2 and Section 4.1: the complete four-block state.

The eight data points as full input–output vectors

Each training example isolates one instruction in the six-bit state.

Data point	Raw input $u_r^\top$ carry 2 | p_2 q_2 | carry 1 | p_1 q_1	Complete output $y^{(r)\top}$ carry 2′ | p_2' q_2' | carry 1′ | p_1' q_1'
1	$(1 \mid 0\ 0 \mid 0 \mid 0\ 0)$	$(1 \mid 0\ 0 \mid 0 \mid 0\ 0)$
2	$(0 \mid 1\ 1 \mid 0 \mid 0\ 0)$	$(1 \mid 0\ 0 \mid 0 \mid 0\ 0)$
3	$(0 \mid 0\ 1 \mid 0 \mid 0\ 0)$	$(0 \mid 1\ 0 \mid 0 \mid 0\ 0)$
4	$(0 \mid 1\ 0 \mid 0 \mid 0\ 0)$	$(0 \mid 1\ 0 \mid 0 \mid 0\ 0)$
5	$(0 \mid 0\ 0 \mid 1 \mid 0\ 0)$	$(0 \mid 0\ 1 \mid 0 \mid 0\ 0)$
6	$(0 \mid 0\ 0 \mid 0 \mid 1\ 1)$	$(0 \mid 0\ 0 \mid 1 \mid 0\ 0)$
7	$(0 \mid 0\ 0 \mid 0 \mid 0\ 1)$	$(0 \mid 0\ 0 \mid 0 \mid 1\ 0)$
8	$(0 \mid 0\ 0 \mid 0 \mid 1\ 0)$	$(0 \mid 0\ 0 \mid 0 \mid 1\ 0)$

Paper, Figures 2–3. Vectors are displayed transposed; bars mark the four blocks.

The eight data points as full input–output vectors

Each training example isolates one instruction in the six-bit state.

Data point	Raw input $u_r^\top$ carry 2 | p_2 q_2 | carry 1 | p_1 q_1	Complete output $y^{(r)\top}$ carry 2′ | p_2' q_2' | carry 1′ | p_1' q_1'
1	$(1 \mid 0\ 0 \mid 0 \mid 0\ 0)$	$(1 \mid 0\ 0 \mid 0 \mid 0\ 0)$
2	$(0 \mid 1\ 1 \mid 0 \mid 0\ 0)$	$(1 \mid 0\ 0 \mid 0 \mid 0\ 0)$
3	$(0 \mid 0\ 1 \mid 0 \mid 0\ 0)$	$(0 \mid 1\ 0 \mid 0 \mid 0\ 0)$
4	$(0 \mid 1\ 0 \mid 0 \mid 0\ 0)$	$(0 \mid 1\ 0 \mid 0 \mid 0\ 0)$
5	$(0 \mid 0\ 0 \mid 1 \mid 0\ 0)$	$(0 \mid 0\ 1 \mid 0 \mid 0\ 0)$
6	$(0 \mid 0\ 0 \mid 0 \mid 1\ 1)$	$(0 \mid 0\ 0 \mid 1 \mid 0\ 0)$
7	$(0 \mid 0\ 0 \mid 0 \mid 0\ 1)$	$(0 \mid 0\ 0 \mid 0 \mid 1\ 0)$
8	$(0 \mid 0\ 0 \mid 0 \mid 1\ 0)$	$(0 \mid 0\ 0 \mid 0 \mid 1\ 0)$

State $x^\top$	
$(0 \mid 1\ 1 \mid 0 \mid 0\ 1)$	Matches data points 2 and 7. Outside-block zeros are padding.

Paper, Figures 2–3. Vectors are displayed transposed; bars mark the four blocks.

Recover the sum by matching data rows

Match(x) returns the rows whose condition holds in that row's block.

$$\mathrm{Match}(x) = \{2, 7\}.$$

$$f_{\mathrm{match}}(x) = \bigvee_{r \in \mathrm{Match}(x)} y^{(r)}.$$

Paper, Eqs. (5)–(6) and Figure 2. Match(x) is row-index notation for the paper's blockwise matcher.

Recover the sum by matching data rows

Match(x) returns the rows whose condition holds in that row's block.

$$\mathrm{Match}(x) = \{2, 7\}.$$

$$f_{\mathrm{match}}(x) = \bigvee_{r \in \mathrm{Match}(x)} y^{(r)}.$$

$$f_{\mathrm{match}}(x) = y^{(2)} + y^{(7)} = (1 \mid 0\ 0 \mid 0 \mid 1\ 0)^{\top}.$$

Here the two writes are disjoint, so bitwise OR equals their sum.

Read carry 2, then p_2, p_1: $\quad 10_2 \ + \ 11_2 \ = \ 101_2.$

Paper, Eqs. (5)–(6) and Figure 2. Match(x) is row-index notation for the paper's blockwise matcher.

How does the network use these data points?

Data points 2 and 7 match our state. Can the mean recover their outputs?

$$\mu_j(x) = \underbrace{\Theta(x, X)}_{\text{similarities}} \underbrace{K^{-1}}_{\text{reweighting}} \underbrace{y_j}_{\text{output labels}} .$$

$K = \Theta(X, X)$ compares the training inputs with each other.

y_j lists one output bit's labels across the eight training data points.

Paper, Eq. (3). One output bit j; X denotes the chosen training-input representation.

Encode the whole condition in its own direction

Make similarity recognize the whole condition, not just shared 1s.

$$
\begin{aligned}
(1,1) &\longmapsto (1,0,0)\\
(0,1) &\longmapsto (0,1,0)\\
(1,0) &\longmapsto (0,0,1)
\end{aligned}
$$

Use $(0,0,0)$ for $(0,0)$. Carry blocks keep their one coordinate.

Paper, Section 5.1. A chosen one-hot orthonormal encoding; the output labels do not change.

Encode the whole condition in its own direction

Make similarity recognize the whole condition, not just shared 1s.

$$
\begin{aligned}
(1,1) &\longmapsto (1,0,0)\\
(0,1) &\longmapsto (0,1,0)\\
(1,0) &\longmapsto (0,0,1)
\end{aligned}
$$

Use $(0,0,0)$ for $(0,0)$. Carry blocks keep their one coordinate.

Different conditions now have zero overlap, even when their raw bits overlap.

Paper, Section 5.1. A chosen one-hot orthonormal encoding; the output labels do not change.

Apply the encoding to the full vectors

Data points 1–4: the high carry and high summand block. Every zero is shown.

Data point	Raw input $u_r^\top$ carry 2 | p_2 q_2 | carry 1 | p_1 q_1	Encoded input $v_r^\top$ carry 2 | (11, 01, 10) | carry 1 | (11, 01, 10)
1	$(1 \mid 0\ 0 \mid 0 \mid 0\ 0)$	$(1 \mid 0\ 0\ 0 \mid 0 \mid 0\ 0\ 0)$
2	$(0 \mid 1\ 1 \mid 0 \mid 0\ 0)$	$(0 \mid 1\ 0\ 0 \mid 0 \mid 0\ 0\ 0)$
3	$(0 \mid 0\ 1 \mid 0 \mid 0\ 0)$	$(0 \mid 0\ 1\ 0 \mid 0 \mid 0\ 0\ 0)$
4	$(0 \mid 1\ 0 \mid 0 \mid 0\ 0)$	$(0 \mid 0\ 0\ 1 \mid 0 \mid 0\ 0\ 0)$

Paper, Section 5.1 and Figure 3. Full vectors displayed transposed; output labels are unchanged from slide 11.

The other four data points

Data points 5–8: the low carry and low summand block. The row order stays fixed.

Data point	Raw input $u_r^\top$ carry 2 | p_2 q_2 | carry 1 | p_1 q_1	Encoded input $v_r^\top$ carry 2 | (11, 01, 10) | carry 1 | (11, 01, 10)
5	$(0 \mid 0\ 0 \mid 1 \mid 0\ 0)$	$(0 \mid 0\ 0\ 0 \mid 1 \mid 0\ 0\ 0)$
6	$(0 \mid 0\ 0 \mid 0 \mid 1\ 1)$	$(0 \mid 0\ 0\ 0 \mid 0 \mid 1\ 0\ 0)$
7	$(0 \mid 0\ 0 \mid 0 \mid 0\ 1)$	$(0 \mid 0\ 0\ 0 \mid 0 \mid 0\ 1\ 0)$
8	$(0 \mid 0\ 0 \mid 0 \mid 1\ 0)$	$(0 \mid 0\ 0\ 0 \mid 0 \mid 0\ 0\ 1)$

Paper, Section 5.1 and Figure 3. Full vectors displayed transposed; output labels are unchanged from slide 11.

Orthogonality simplifies the inverse in the mean

The eight encoded training inputs are orthonormal.

$$\mu_j(\hat{x}) = \Theta(\hat{x}, X)\, K^{-1} y_j.$$

$$K^{-1} = \alpha I_8 - \beta \mathbf{1}\mathbf{1}^\top, \qquad \alpha, \beta > 0.$$

Paper, Appendix C, inverse after Eq. (13). K is the training NTK; exact constants are in the notes.

Orthogonality simplifies the inverse in the mean

The eight encoded training inputs are orthonormal.

$$\mu_j(\hat{x}) = \Theta(\hat{x}, X)\, K^{-1} y_j.$$

$$K^{-1} = \alpha I_8 - \beta \mathbf{1}\mathbf{1}^\top, \qquad \alpha, \beta > 0.$$

The constants α and β are fixed by the training kernel.

The identity rescales $\Theta(\hat{x}, X)$.
The all-ones term subtracts the same correction from every entry.

Paper, Appendix C, inverse after Eq. (13). K is the training NTK; exact constants are in the notes.

What multiplies each data point's output?

For our two matches, $\hat{x} = (v_2 + v_7)/\sqrt{2}$. Write the mean as $\mu_j = h\, y_j$.

$$h := \Theta(\hat{x}, X) K^{-1}.$$

Worked multiplication of Appendix C's inverse. h is a row; y_j is a column. No new predictor is introduced.

What multiplies each data point's output?

For our two matches, $\hat{x} = (v_2 + v_7)/\sqrt{2}$. Write the mean as $\mu_j = h\, y_j$.

$$h := \Theta(\hat{x}, X) K^{-1}.$$

$$h_r = \underbrace{\alpha\, \Theta(\hat{x}, v_r)}_{\text{rescale data point } r\text{'s similarity}} - \underbrace{\beta \sum_{s=1}^{8} \Theta(\hat{x}, v_s)}_{\text{same subtraction for every data point}}.$$

This is the previous inverse multiplied into the similarity row.

Worked multiplication of Appendix C's inverse. h is a row; y_j is a column. No new predictor is introduced.

Evaluate that same calculation for our input

The paper's kernel gives the following rounded values.

Matching data point: 2 or 7	
Rescaled similarity	0.76
Shared sub-traction	−0.20

Any nonmatching data point	
Rescaled similarity	0.19
Shared sub-traction	−0.20

Numerical illustration of Appendix C: eight codes, two matches. Exact calculations and rounding are in the notes.

Evaluate that same calculation for our input

The paper's kernel gives the following rounded values.

Matching data point: 2 or 7

Rescaled similarity	0.76
Shared subtraction	−0.20

$$h_2 = h_7 \simeq +0.56$$

Any nonmatching data point

Rescaled similarity	0.19
Shared subtraction	−0.20

$$h_r \simeq -0.01$$

These are the coefficients in $\mu_j = h\, y_j$, not hand-chosen data point scores.

Numerical illustration of Appendix C: eight codes, two matches. Exact calculations and rounding are in the notes.

Low sum bit: multiply by the full label vector

y_{p_1} contains the eight labels for p'_1 in the dataset on slide 11.

$$\mu_{p_1}(\hat{x}) = h\, y_{p_1}.$$

Data point r	1	2	3	4	5	6	7	8
h (rounded)	−0.01	+0.56	−0.01	−0.01	−0.01	−0.01	+0.56	−0.01
$y_{p_1}^{\top}$	0	0	0	0	0	0	1	1

Multiply each coefficient by the label directly below it, then add.

Paper, Eq. (7). Full data-point order. Green columns have label 1; displayed coefficients are rounded.

Low sum bit: multiply by the full label vector

y_{p_1} contains the eight labels for p'_1 in the dataset on slide 11.

$$\mu_{p_1}(\hat{x}) = h\, y_{p_1}.$$

Data point r	1	2	3	4	5	6	7	8
h (rounded)	−0.01	+0.56	−0.01	−0.01	−0.01	−0.01	+0.56	−0.01
$y_{p_1}^{\top}$	0	0	0	0	0	0	1	1

Multiply each coefficient by the label directly below it, then add.

$$\mu_{p_1} = h_1 \cdot 0 + h_2 \cdot 0 + h_3 \cdot 0 + h_4 \cdot 0 + h_5 \cdot 0 + h_6 \cdot 0 + h_7 \cdot 1 + h_8 \cdot 1.$$

Paper, Eq. (7). Full data-point order. Green columns have label 1; displayed coefficients are rounded.

Low sum bit: multiply by the full label vector

y_{p_1} contains the eight labels for p'_1 in the dataset on slide 11.

$$\mu_{p_1}(\hat{x}) = h\, y_{p_1}.$$

Data point r	1	2	3	4	5	6	7	8
h (rounded)	−0.01	+0.56	−0.01	−0.01	−0.01	−0.01	+0.56	−0.01
$y_{p_1}^{\top}$	0	0	0	0	0	0	1	1

Multiply each coefficient by the label directly below it, then add.

$$\mu_{p_1} = h_1 \cdot 0 + h_2 \cdot 0 + h_3 \cdot 0 + h_4 \cdot 0 + h_5 \cdot 0 + h_6 \cdot 0 + h_7 \cdot 1 + h_8 \cdot 1.$$

$$\mu_{p_1} = h_7 + h_8 \simeq 0.55 > 0 \quad \Longrightarrow \quad p'_1 = 1.$$

Even matching data point 2 contributes zero here: its label for this bit is 0.

Paper, Eq. (7). Full data-point order. Green columns have label 1; displayed coefficients are rounded.

High sum bit: change only the label vector

Keep the same input and coefficients h. Now use the eight labels for p_2'.

$$\mu_{p_2}(\hat{x}) = h\, y_{p_2}.$$

Data point r	1	2	3	4	5	6	7	8
h (rounded)	−0.01	+0.56	−0.01	−0.01	−0.01	−0.01	+0.56	−0.01
$y_{p_2}^{\top}$	0	0	1	1	0	0	0	0

Multiply each coefficient by the label directly below it, then add.

Paper, Eq. (7). Full data-point order. Green columns have label 1; displayed coefficients are rounded.

High sum bit: change only the label vector

Keep the same input and coefficients h. Now use the eight labels for p_2'.

$$\mu_{p_2}(\hat{x}) = h\, y_{p_2}.$$

Data point r	1	2	3	4	5	6	7	8
h (rounded)	−0.01	+0.56	−0.01	−0.01	−0.01	−0.01	+0.56	−0.01
$y_{p_2}^{\top}$	0	0	1	1	0	0	0	0

Multiply each coefficient by the label directly below it, then add.

$$\mu_{p_2} = h_1 \cdot 0 + h_2 \cdot 0 + h_3 \cdot 1 + h_4 \cdot 1 + h_5 \cdot 0 + h_6 \cdot 0 + h_7 \cdot 0 + h_8 \cdot 0.$$

Paper, Eq. (7). Full data-point order. Green columns have label 1; displayed coefficients are rounded.

High sum bit: change only the label vector

Keep the same input and coefficients h. Now use the eight labels for p_2'.

$$\mu_{p_2}(\hat{x}) = h\, y_{p_2}.$$

Data point r	1	2	3	4	5	6	7	8
h (rounded)	−0.01	+0.56	−0.01	−0.01	−0.01	−0.01	+0.56	−0.01
$y_{p_2}^{\top}$	0	0	1	1	0	0	0	0

Multiply each coefficient by the label directly below it, then add.

$$\mu_{p_2} = h_1 \cdot 0 + h_2 \cdot 0 + h_3 \cdot 1 + h_4 \cdot 1 + h_5 \cdot 0 + h_6 \cdot 0 + h_7 \cdot 0 + h_8 \cdot 0.$$

$$\mu_{p_2} = h_3 + h_4 \simeq -0.02 < 0 \quad \Longrightarrow \quad p_2' = 0.$$

Data points 2 and 7 match the input, but both have label 0 for this bit.

Paper, Eq. (7). Full data-point order. Green columns have label 1; displayed coefficients are rounded.

The mean gives the desired update

Decode each mean coordinate: positive $\to$ 1, otherwise $\to$ 0.

$$\text{carry } 2' \mid p'_2\ q'_2 \mid \text{carry } 1' \mid p'_1\ q'_1$$

$$x' = (1 \mid 0\ 0 \mid 0 \mid 1\ 0)^\top.$$

Paper, Figure 2, Eq. (7) and Section 6. The known-zero q1 output is handled separately; see notes.

The mean gives the desired update

Decode each mean coordinate: positive $\to$ 1, otherwise $\to$ 0.

$$\text{carry } 2' \mid p_2' \; q_2' \mid \text{carry } 1' \mid p_1' \; q_1'$$

$$x' = (1 \mid 0\ 0 \mid 0 \mid 1\ 0)^\top.$$

Read carry 2, then p_2, p_1: 101_2.

The decoded mean is correct. A single network can still fluctuate.

Paper, Figure 2, Eq. (7) and Section 6. The known-zero q1 output is handled separately; see notes.

Now bring back the random initialization

The mean is fixed by the instruction data; individual trained networks can differ.

$$F_{\infty,j}(\hat{x}) = \mu_j(\hat{x}) + g_j(\hat{x}).$$

Paper, Theorem 3.1, Eqs. (3)–(4). Converged NTK limit; randomness is over initialization.

Now bring back the random initialization

The mean is fixed by the instruction data; individual trained networks can differ.

$$F_{\infty,j}(\hat{x}) = \mu_j(\hat{x}) + g_j(\hat{x}).$$

$$g_j(\hat{x}) \sim \mathcal{N}\left(0, \sigma^2(\hat{x})\right).$$

Correct mean and small random spread are two separate requirements.

Paper, Theorem 3.1, Eqs. (3)–(4). Converged NTK limit; randomness is over initialization.

What controls the remaining variance?

In the paper's model normalization, Lemma 6.1 gives

$$\sigma^2(\hat{x}) = O(1/D).$$

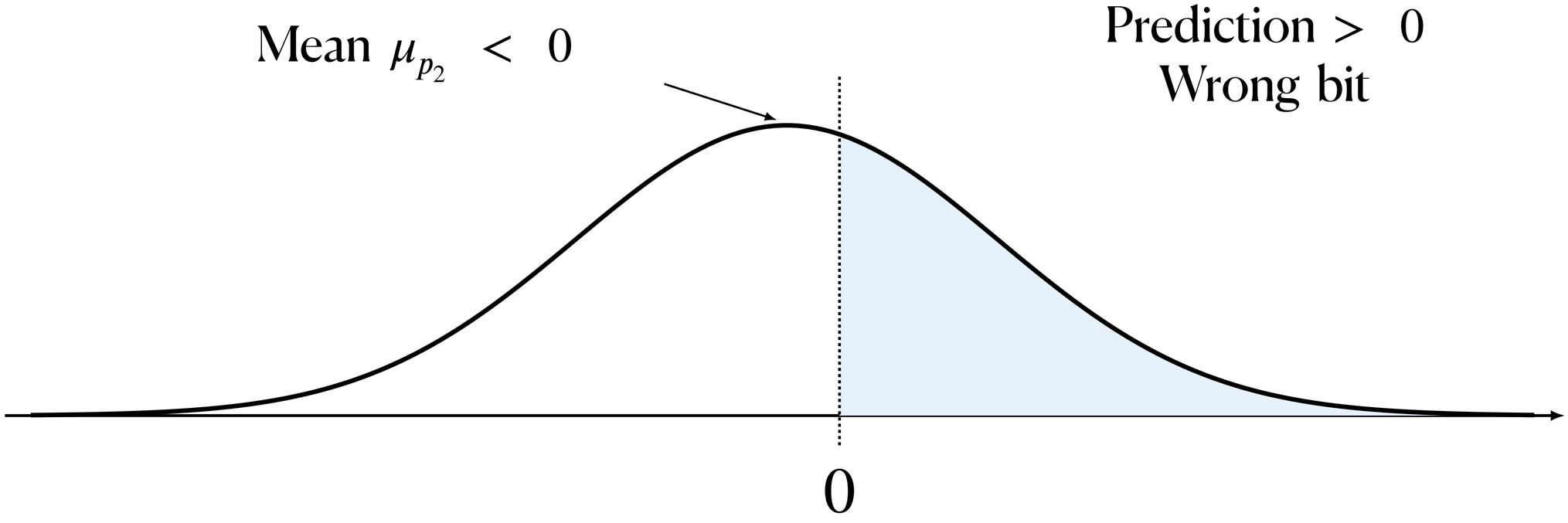
D is the encoded dimension. In our example, $D = 8$.

Small variance alone is not enough: compare the spread with the distance to zero.

Paper, Lemma 6.1. Extra model factor 1/sqrt(D) before training; normalization and derivation in notes.

A correct mean can still give a wrong prediction

The high sum bit should be 0. Its mean is negative, but a prediction can cross zero.



The spread must be small compared with the distance to zero.

Gaussian illustration for the high sum bit. No probability estimate is needed to read the picture.

Average predictions from independent models

Train N networks on the same instructions, with independent initializations.

$$\overline{F}_j = \frac{1}{N} \sum_{n=1}^{N} F_j^{(n)}.$$

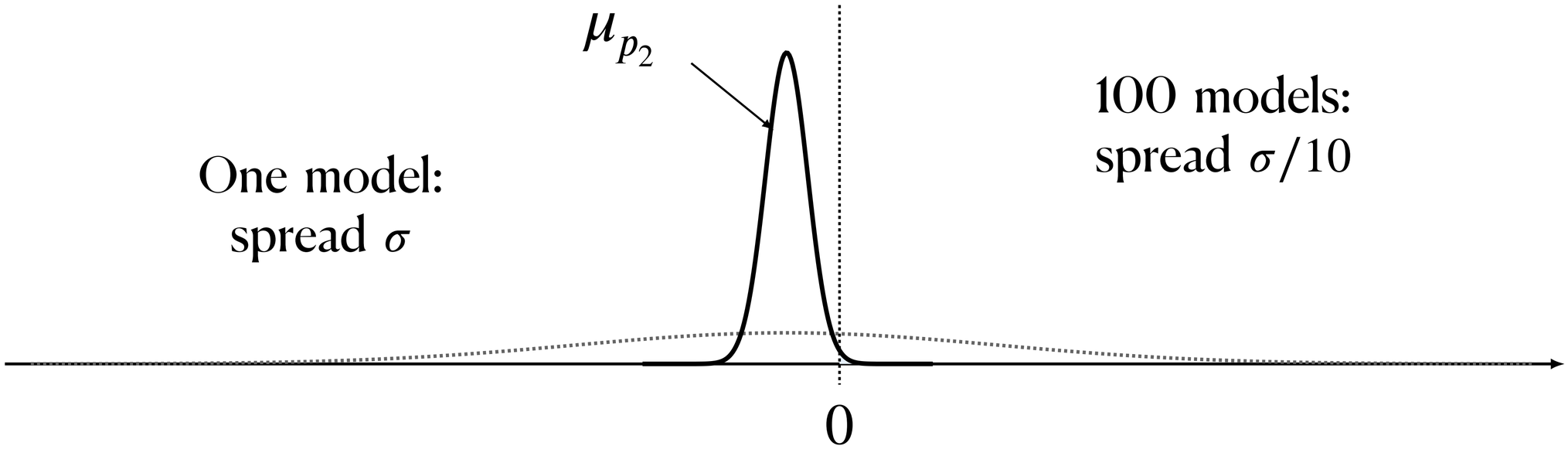
$$\mathrm{E}[\overline{F}_j] = \mu_j, \qquad \mathrm{Var}(\overline{F}_j) = \frac{\sigma^2}{N}.$$

The mean stays fixed. The standard deviation shrinks by a factor of $\sqrt{N}$.

Paper, Section 6. Average outputs, not network parameters.

Same bit, same mean, much less spread

Return to the same high sum bit p_2', now averaging $N = 100$ networks.



Averaging narrows the distribution without moving its center.

Gaussian illustration: same center and threshold. Dashed and solid curves have equal area.

Substitute the mean and variance bounds

For addition, use the weakest nonzero-mean rate in Lemma 6.1.

Variance upper bound

$$\sigma^2(\hat{x}) = O(D^{-1})$$

Mean magnitude lower bound

$$|\mu_j(\hat{x})| = \Omega(D^{-1})$$

$$\sigma^2(\hat{x})/\mu_j(\hat{x})^2 \leq CD.$$

$C > 0$ is independent of D and N; use the same C for every bit being protected.

$$\Pr(\text{wrong bit } j) \leq 2\exp\left(-\frac{N\mu_j^2}{8\sigma^2}\right) \leq 2\exp\left(-\frac{N}{8CD}\right).$$

What does M count?

One decision means rounding one predicted next-state bit to 0 or 1.

Each update of our two-bit state predicts five nonconstant bits:

$$\text{carry } 2', \quad p_2', \quad q_2', \quad \text{carry } 1', \quad p_1'.$$

The known-zero q_1' is set to 0 directly (the added decoder convention in the notes).

Which computation is protected?	Number of bit predictions M
Our $10_2 + 11_2$ example: one update	$M = 5$
One two-bit addition: up to 4 updates	$M \le 5 \times 4 = 20$
All 16 two-bit additions: up to 4 updates each	$M \le 16 \times 5 \times 4 = 320$

M counts output-bit predictions, not training data points or networks.

Worked count from Figure 2 and Section 4.1 (at most 2ℓ updates). Count predictions along the correct algorithm states.

Choose N to protect every predicted bit

Use the same ensemble at every update. Allow total failure probability at most δ.

$$\Pr(\text{at least one wrong bit}) \le 2M \exp\left(-\frac{N}{8CD}\right).$$

Add the one-bit error bounds over the M predictions. They need not be independent.

To make that total at most δ, it is sufficient to choose

$$N \ge 8CD \ln(2M/\delta).$$

$$\Pr(\text{every predicted bit is correct}) \ge 1 - \delta.$$

Correct state bits at every update give a correct execution.

Section 6: union bound, with explicit M and the rates from slide 28. Fixed inputs and horizon; constant outputs as stated.