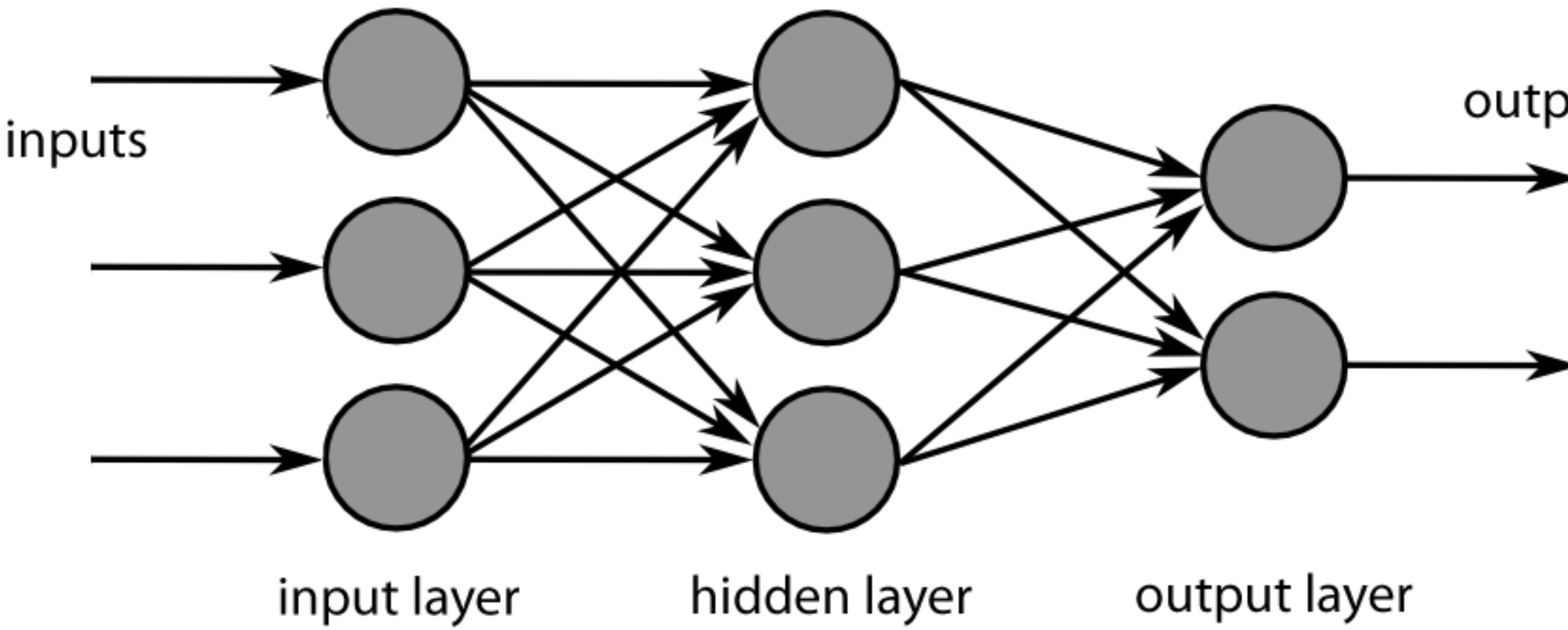
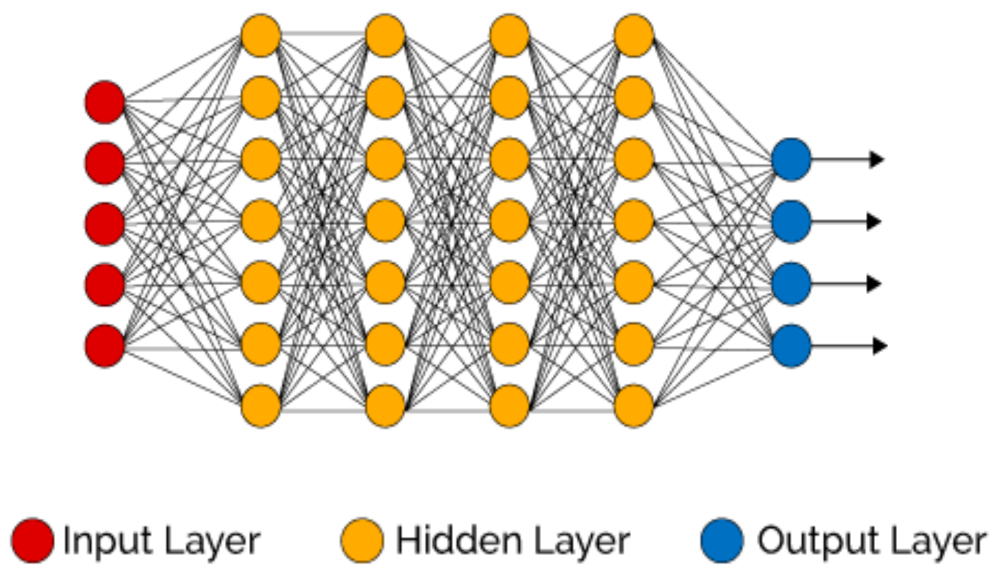
Recent Theoretical Results in Supervised Deep Learning

Speaker: Harish Guruprasad Ramaswamy
IIT Madras



Source: Google Images

Terminology



θ: Parameters of deep net

$(x_1,y_1), (x_2,y_2),\ldots$ iid (point, label) from distribution $\mathcal{D}$ *(training data)*

$\ell(\theta, x, y)$ Loss function (how well net output matched true label y on point x) Can be l_2, cross-entropy….

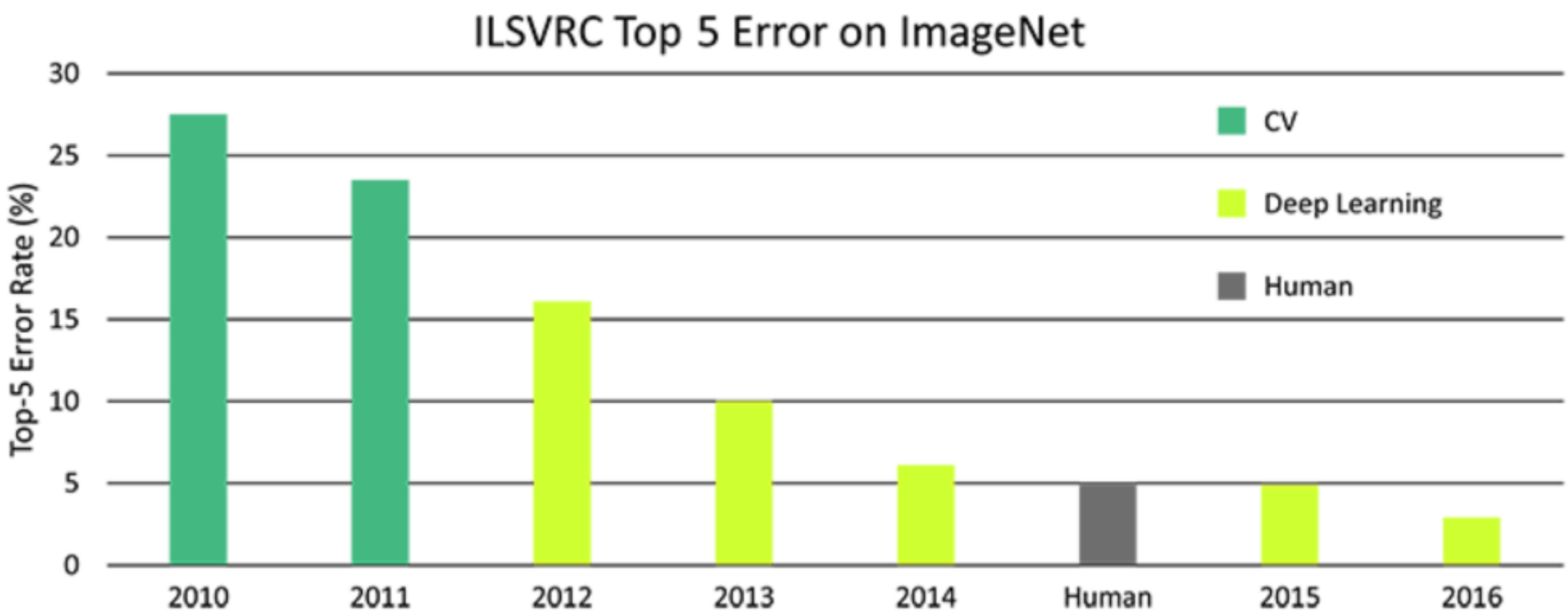
Objective $\text{argmin}_\theta E_i[\ell(\theta, x_i, y_i)]$

Gradient Descent $\theta^{(t+1)} \leftarrow \theta^{(t)} - \eta \nabla_\theta (E_i[\ell(\theta^{(t)}, x_i, y_i)])$

Stochastic GD: Estimate ∇ via small sample of training data.

Source: Sanjeev Arora ICML 2018 Tutorial

The Successes of Deep Learning

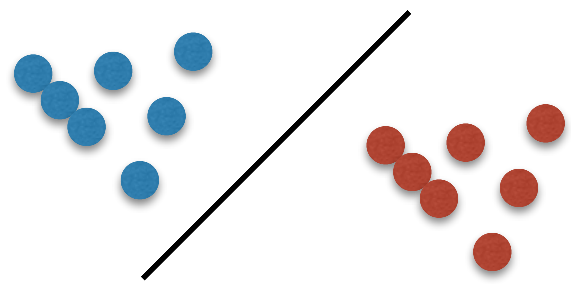


source: https://www.dsiac.org/resources/journals/dsiac/winter-2017-volume-4-number-1/real-time-situ-intelligent-video-analytics

Baby Steps in Deep Learning Theory

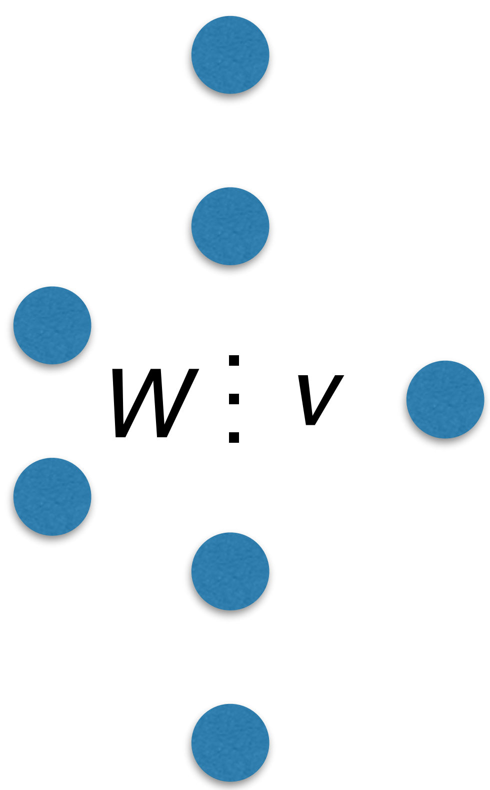
Can Deep Nets Learn Linear Separators?

Data:

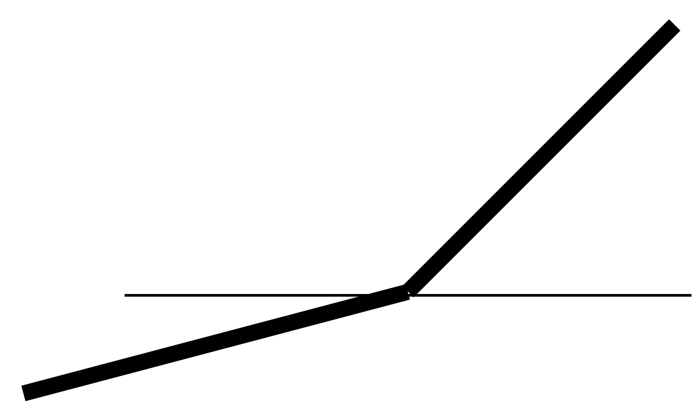


$$y = \text{sign}(\mathbf{x}^\top \mathbf{w}^*)$$

Model:



$$N_{\mathcal{W}}(\boldsymbol{x}) = \boldsymbol{v}^\top \sigma(W\boldsymbol{x})$$



Leaky ReLU

A. Brutzkus et al., “SGD learns over-parameterized net- works that provably generalize on linearly separable data,” in ICLR 2018.

Can Deep Nets Learn Linear Separators?

$$L_S(W) = \frac{1}{n}\sum_{i=1}^{n} \max\{1 - y_i N_{\mathcal{W}}(\boldsymbol{x}_i), 0\}$$

Solve: $$\arg\min_{W \in \mathbb{R}^{2k \times d}} L_S(W)$$

Gradient Descent: $$W_t = W_{t-1} - \eta \frac{\partial}{\partial W} L_{\{(\boldsymbol{x}_t, y_t)\}}(W_{t-1})$$

Why can it fail?

1. The loss is non-convex in W.
2. W can potentially “overfit” and not generalise.

Can Deep Nets Learn Linear Separators?

Theorem 1: Every critical point of $L_S(W)$ is a global minima.

Theorem 2: SGD converges to a global minimum after performing at most

$$M = \frac{||\mathbf{w}^*||^2}{\alpha^2} + O\left(\frac{||\mathbf{w}^*||^2}{\eta}\right)$$

non-zero updates.

Theorem 3: For $n > 2M$, the test error goes down as follows:

$$L_D^{0-1}(SGD) = \widetilde{O}\left(\frac{1}{n}\right)$$

Can Deep Nets Learn Linear Separators?

Take home points:

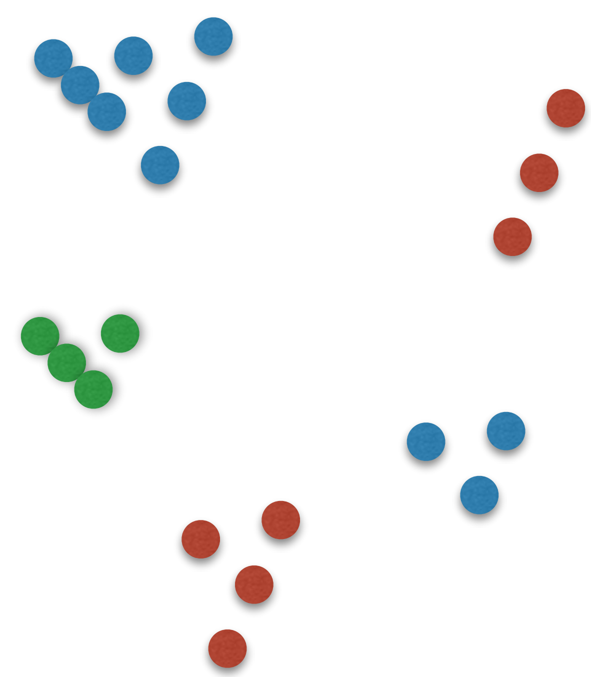
1. SGD acts as a “regulariser” biasing towards classifiers that can be expressed compactly.
2. Perceptron proof technique!

Future directions open:

1. What if the vector “v” was also learnt using SGD?
2. What if the activation function was plain ReLU?

Can Deep Nets Learn Structured Data?

Structured data:



(Separability) There exists $\delta > 0$ such that for every $i_1 \neq i_2 \in [k]$ and every $j_1, j_2 \in [l]$,

$$\mathrm{dist}\left(\mathrm{supp}(\mathcal{D}_{i_1,j_1}), \mathrm{supp}(\mathcal{D}_{i_2,j_2})\right) \geq \delta.$$

Moreover, for every $i \in [k], j \in [l]$,[1]

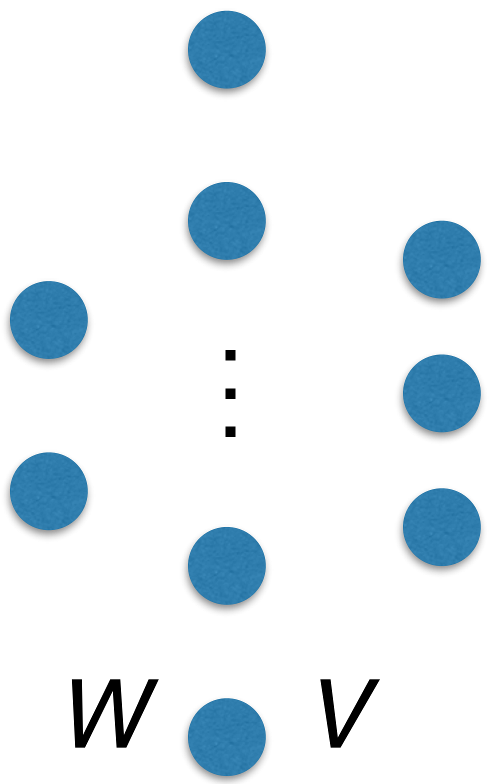
$$\mathrm{diam}(\mathrm{supp}(\mathcal{D}_{i,j})) \leq \lambda\delta, \text{ for } \lambda \leq 1/(8l).$$

Yuanzhi Li, Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. NIPS 2018.

Can Deep Nets Learn Structured Data?

m hidden nodes, and k classes.

Model:



One layer ReLU: $f_i(\mathbf{x}) = \sum_{r=1}^{m} V_{i,r}\phi(\mathbf{w}_r^\top \mathbf{x})$

Output Probs: $o(\mathbf{x}) = \text{softmax}(\mathbf{f}(\mathbf{x}))$

Loss function: $$L(W) = -\sum_{s=1}^{N} \log o_{y_s}(\mathbf{x}_s, W)$$

Can Deep Nets Learn Structured Data?

Key Theorem: For large enough number of hidden nodes, SGD will output “right” answer with high probability.

Can Deep Nets Learn Structured Data?

Proof Ideas:

Gradient:

$$\frac{\partial L(W)}{\partial \mathbf{w}_r} = \sum_{a \in [k], b \in [l]} p_{a,b} \left(\sum_{i \neq a} (V_{i,r} - V_{a,r}) o_i(\mathbf{x}_{alb}, W) \right) \cdot \mathbf{x}_{a,b} \cdot 1(\mathbf{w}_r^\top \mathbf{x}_{a,b} \geq 0)$$

"Pseudo-Gradient":

$$\frac{\widetilde{\partial} L(W)}{\partial \mathbf{w}_r} = \sum_{a \in [k], b \in [l]} p_{a,b} \left(\sum_{i \neq a} (V_{i,r} - V_{a,r}) o_i(\mathbf{x}_{alb}, W) \right) \cdot \mathbf{x}_{a,b} \cdot 1(\mathbf{w}_{r,\mathrm{init}}^\top \mathbf{x}_{a,b} \geq 0)$$

Can Deep Nets Learn Structured Data?

Proof Ideas:

Lemma 1: Throughout the training process, the "pseudo-gradients" are "almost" the same as the true gradients.

Lemma 2: As the pseudo gradients resemble gradients of a convex function, it can be shown that the "pseudo-gradients" are zero only at the global minimiser of the loss.

Can Deep Nets Learn Linear Separators?

Take home points:

1. Overparameterisation enables analysis using a “convex-like” loss function instead of the loss function.

Future directions open:

1. What if the matrix “V” was also learnt using SGD?

2. Proof requires prohibitive amount of overparameterisation, is that necessary?

Weight Tying

Data Model

Data point: $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k) \in \mathbb{R}^{dk}$

Label: $y = g^*(\mathbf{h}^*(\mathbf{x}))$

$$\mathbf{h}^*(\mathbf{x}) = [\sigma(\mathbf{u}_0^\top \mathbf{x}_1), \ldots, \sigma(\mathbf{u}_0^\top \mathbf{x}_k)]$$

Example g^*:

- $g^*_{\text{low}}(\mathbf{z}) = z_1$
- $g^*_{\text{high}}(\mathbf{z}) = \cos(\sum_i \pi z_i)$
- $g^*_{\text{mid}}(\mathbf{z}) = g^*_{\text{high}}(\mathbf{z}) + g^*_{\text{low}}(\mathbf{z})$

Shai Shalev-Shwartz et al. Weight sharing is crucial to succesful optimization. arXiv:1706.00687, 2017

Learning Hypothesis Class

$$\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_k] \in \mathbb{R}^{dk}$$

Prediction: $$p_{\mathbf{w}}(\mathbf{x}) = c_k \mathbf{w}_1^\top \mathbf{x}_1 + \cos\left(\sum_{i=1}^{k} \mathbf{w}_i^\top \mathbf{x}_i\right)$$

Truth: $$p_{\mathbf{u}_0}(\mathbf{x}) = c_k \mathbf{u}_0^\top \mathbf{x}_1 + \cos\left(\sum_{i=1}^{k} \mathbf{u}_0^\top \mathbf{x}_i\right)$$

Loss: $$F(\mathbf{w}) = \mathbb{E}_{\mathbf{x}}\left[\frac{1}{2}\left(p_{\mathbf{w}}(\mathbf{x}) - p_{\mathbf{u}_0}(\mathbf{x})\right)^2\right]$$

Weight-sharing model: $\mathbf{w}_i = \mathbf{w}_0$ (Thus, k times lesser parameters)

Shai Shalev-Shwartz et al. Weight sharing is crucial to succesful optimization. arXiv:1706.00687, 2017

Gradients Vanish For Standard Model

Lemma 1:

If $\|(\mathbf{w}_2,\ldots,\mathbf{w}_k)\| \in \left[\frac{\sqrt{k-1}}{3}\cdot\|\mathbf{u}_0\|, \frac{\sqrt{k-1}}{2}\cdot\|\mathbf{u}_0\|\right]$

then $\left\|\frac{\partial}{\partial(\mathbf{w}_2,\ldots,\mathbf{w}_k)}F(\mathbf{w})\right\| \;\leq\; c_1\sqrt{k}\|\mathbf{u}_0\|\exp\left(-c_2 k\|\mathbf{u}_0\|^2\right)$

Lemma 2:

$$\textit{for any } \mathbf{w} \textit{ such that } \|(\mathbf{w}_2,\ldots,\mathbf{w}_k)\| \leq \frac{\sqrt{k-1}}{2}\|\mathbf{u}_0\|,$$

$$F(\mathbf{w}) - F(\bar{\mathbf{u}}_0) \;\geq\; 1 - c_3\exp(-c_4 k\|\mathbf{u}_0\|^2)$$

Gradients Vanish For Standard Model

Theorem 1: For the standard model, gradient descent will take exponential (in k) iterations to get to the solution.

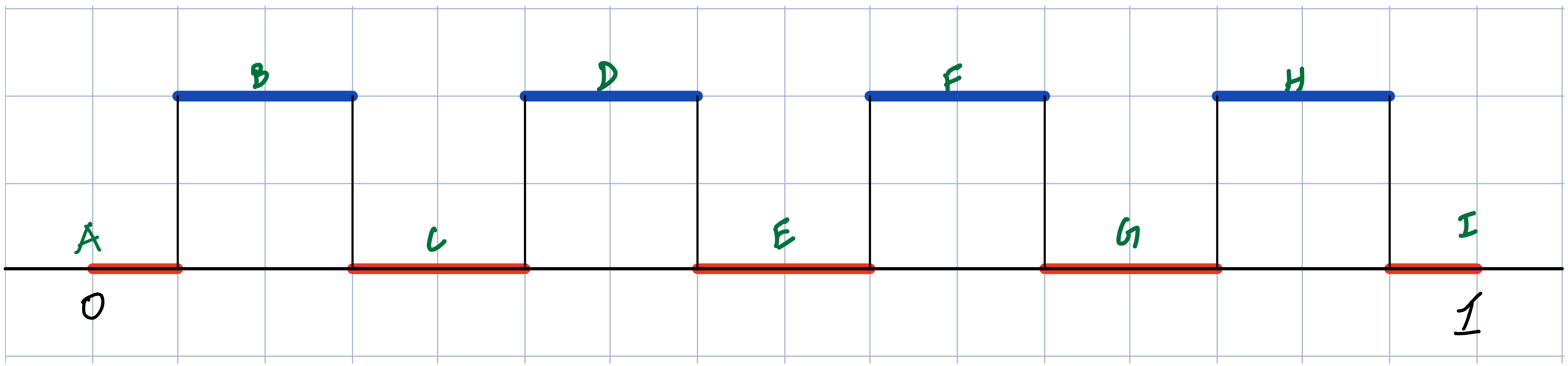
Weight Sharing Model Objective is Convex!

Theorem 2: For the weight-sharing model, the objective becomes strongly convex!

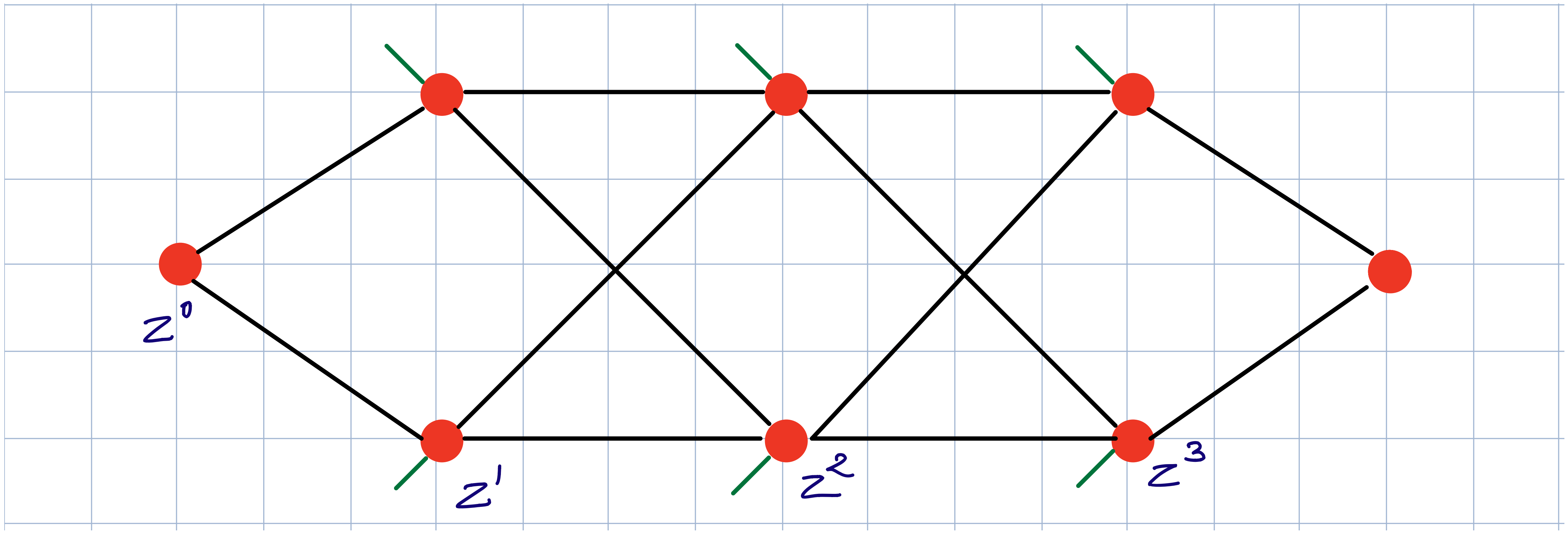
Illustration of Deep Network Learning and Weight Tying

Square Wave Data

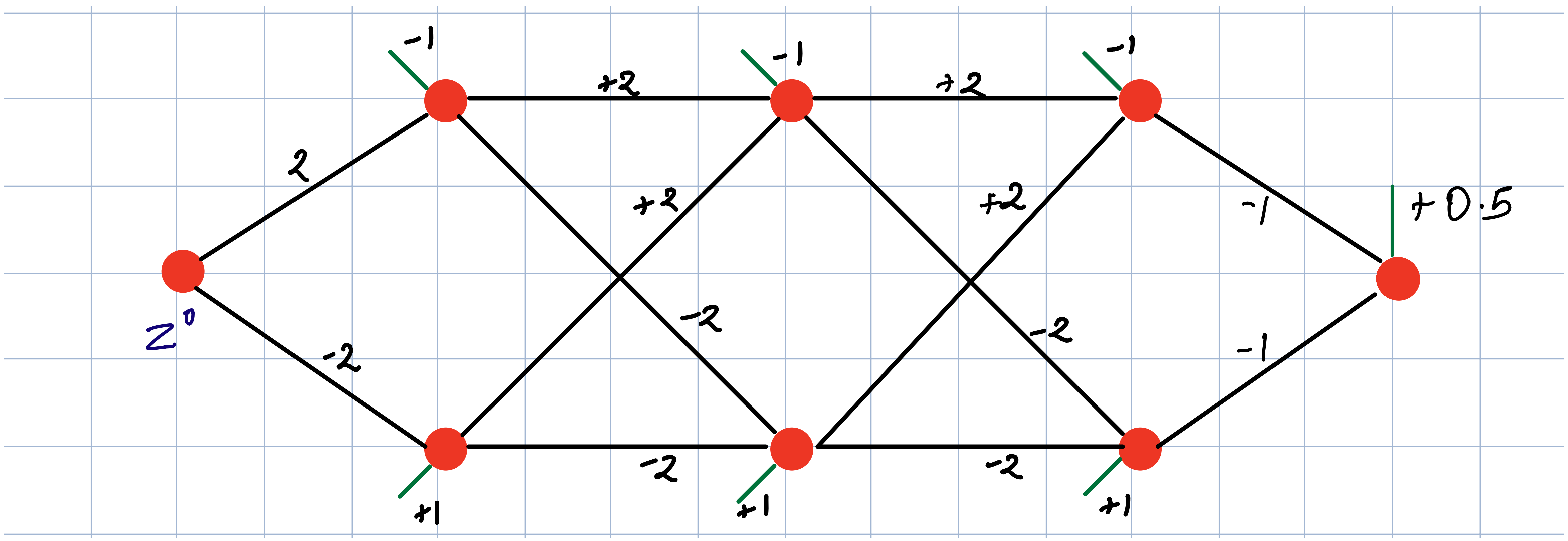
$$x \in [0, 1], y \in \{0, 1\}$$



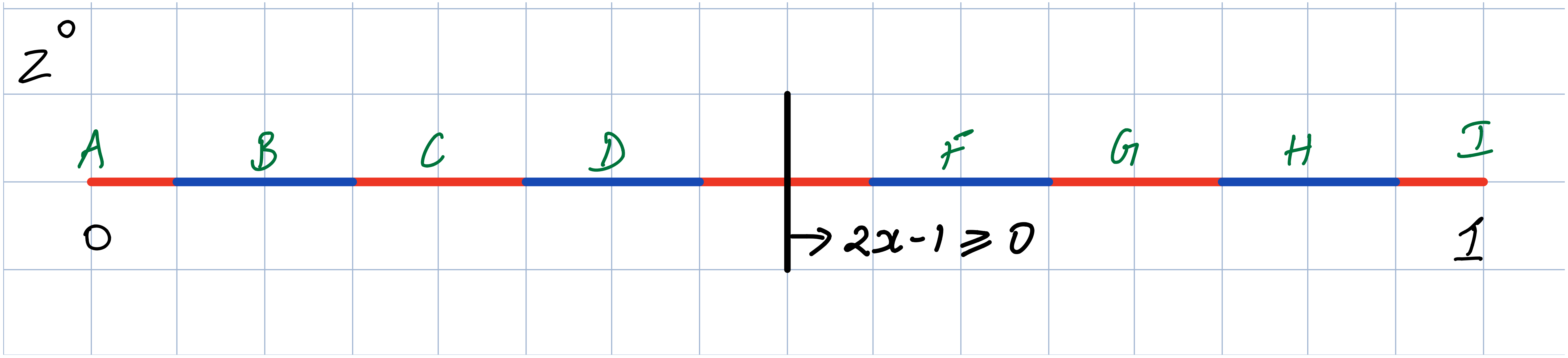
Fully Connected ReLU Network Model



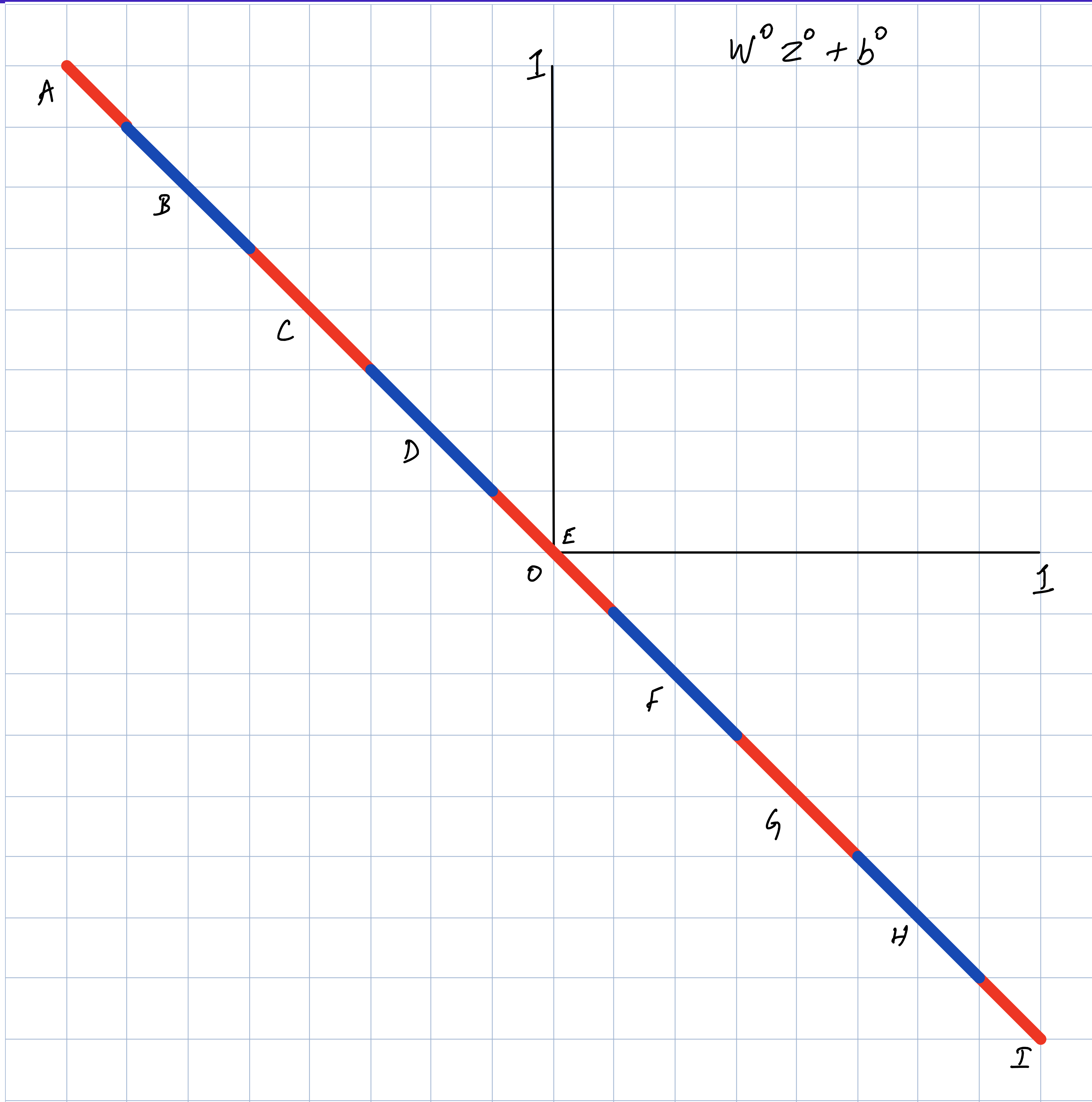
"Correct" Parameters



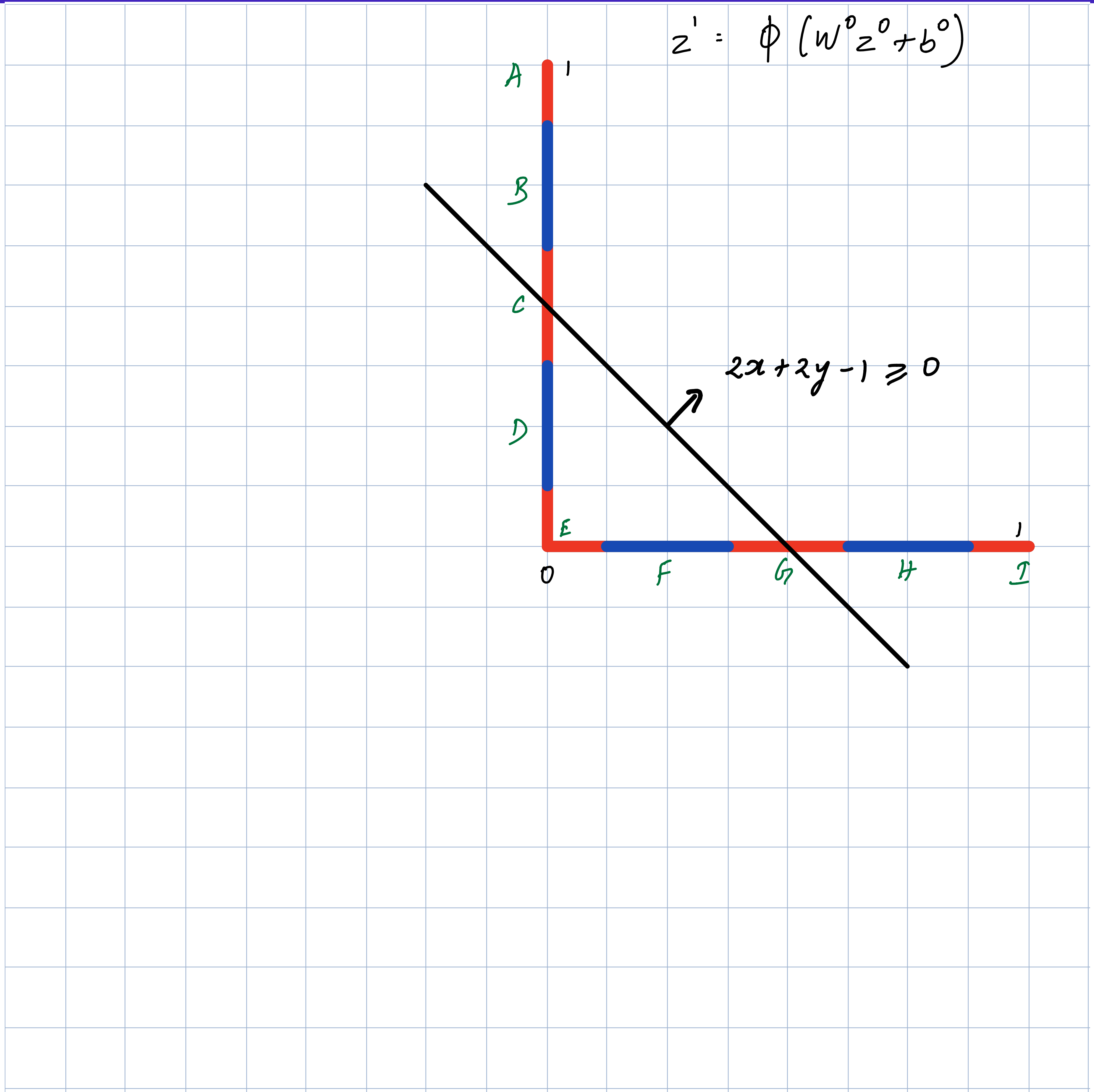
Zeroth Layer Outputs



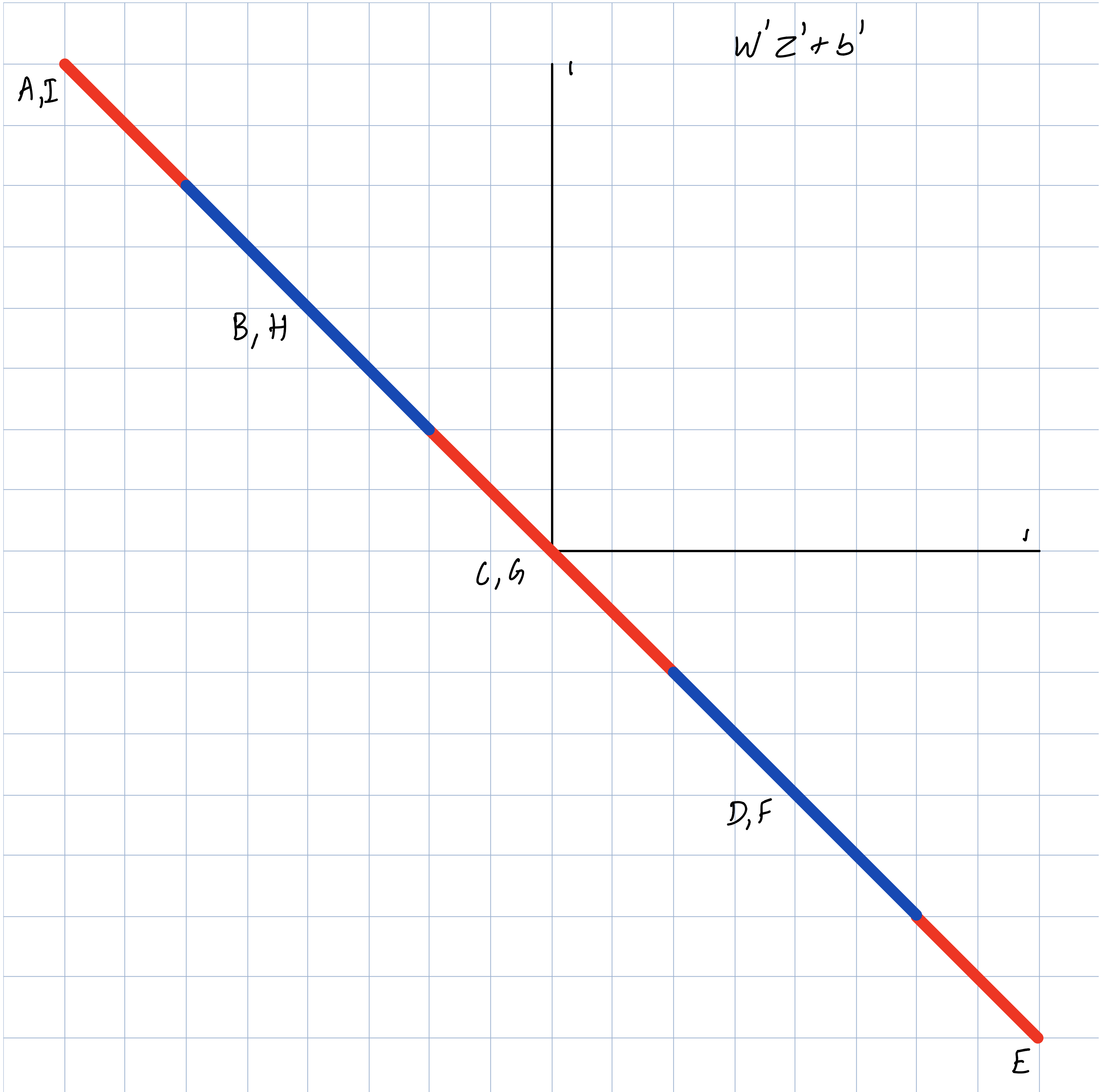
First Layer Pre-Activation



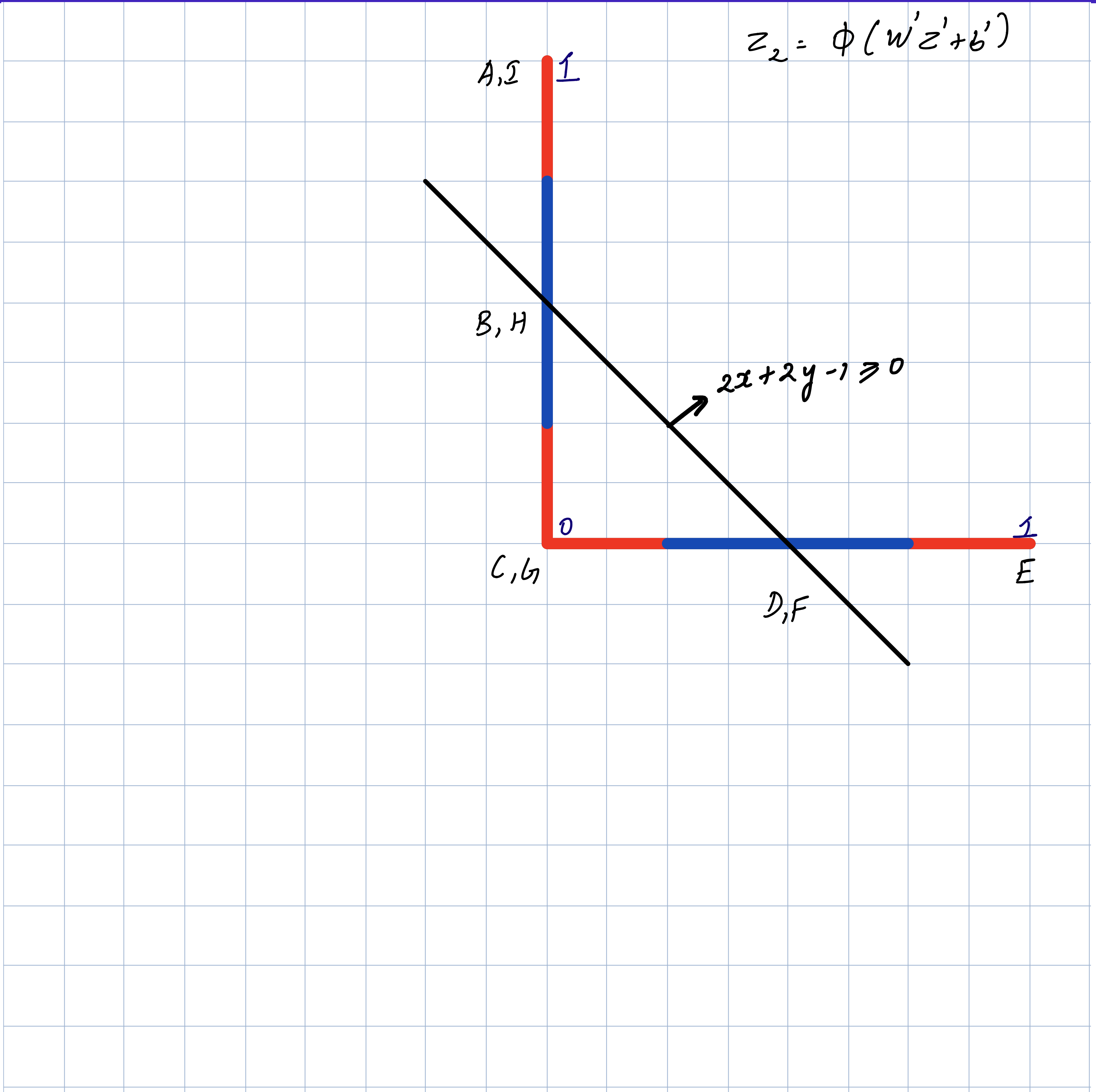
First Layer Output



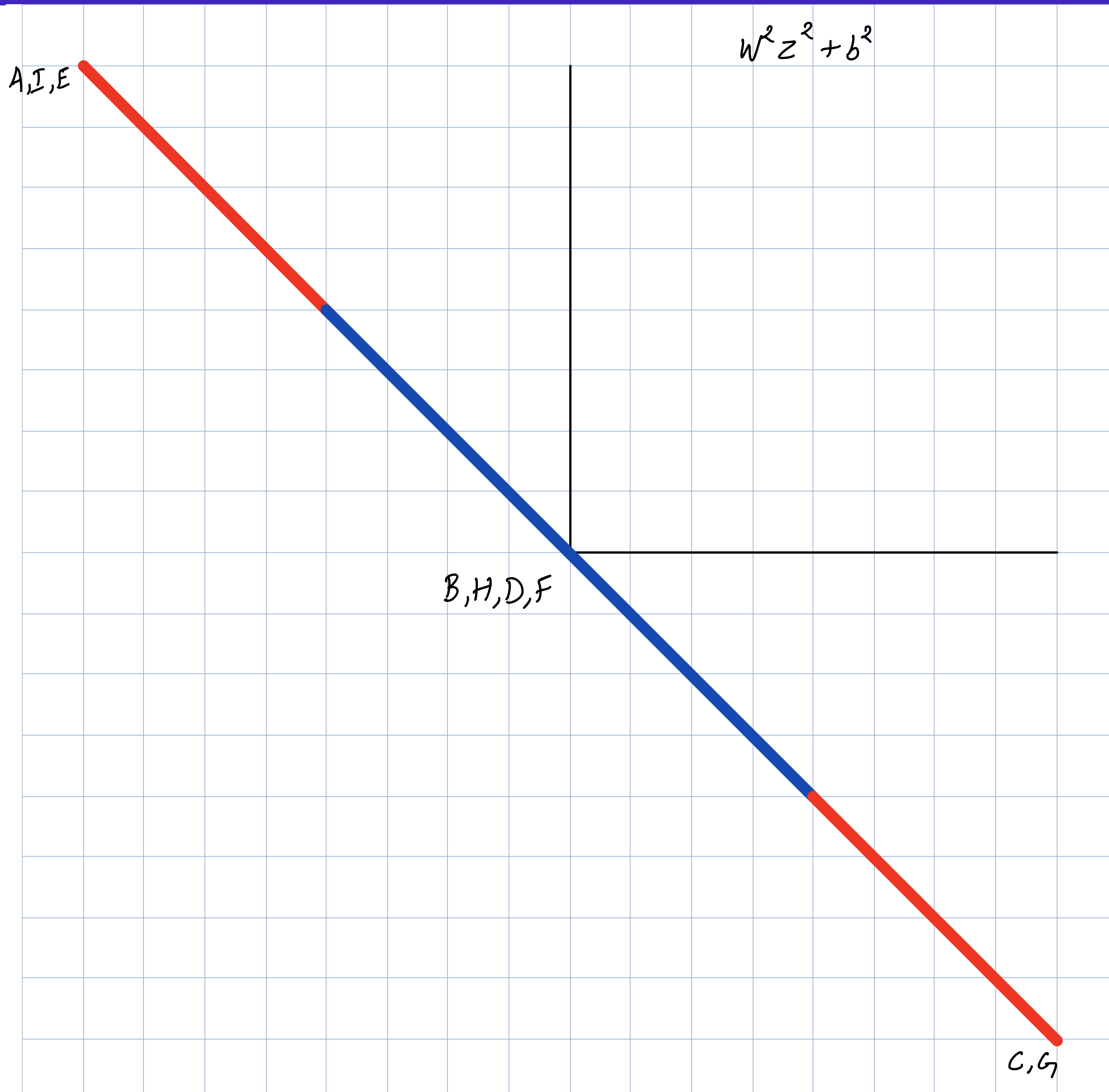
Second Layer Pre-Activation



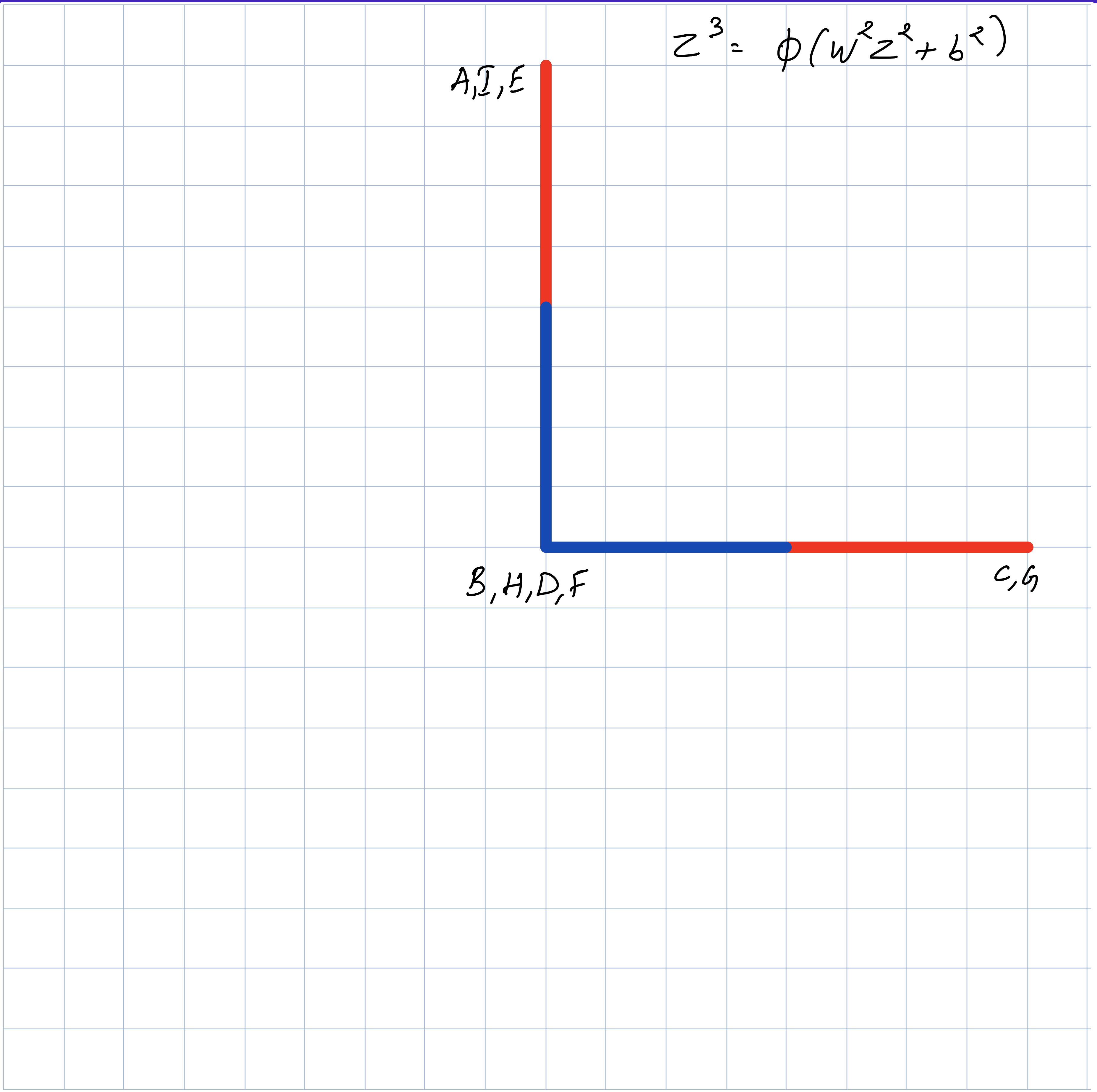
Second Layer Output



Third Layer Pre-Activation



Third Layer Output



Square Wave Learning Tricks

- The objective is highly non-convex and direct minimisation of all parameters fails.
- Observing the symmetry of the “right” answer, weights can be tied “negatively”: i.e. weights going into each node is forced to be the negative of the weights going into the other node in the same layer.
- Number of parameters only halve, but chances of success in stochastic gradient descent improves dramatically.

Theoretical Advantages of Weight Tying

Take Home Points:

- The benefits of weight tying go beyond simply reducing parameter count.

- Symmetries to be exploited should be expressed via tying.
    - Convolutional weight tying: Location agnostic object detection
    - Negative weight tying: Negation symmetry at multiple scales

Theoretical Advantages of Weight Tying

Future Directions:

- Discover the “inductive bias” for various mechanisms of weight tying.
- New weight tying schemes
- Show benefits of weight tying with multiple layers.

Generalisation in Deep Networks

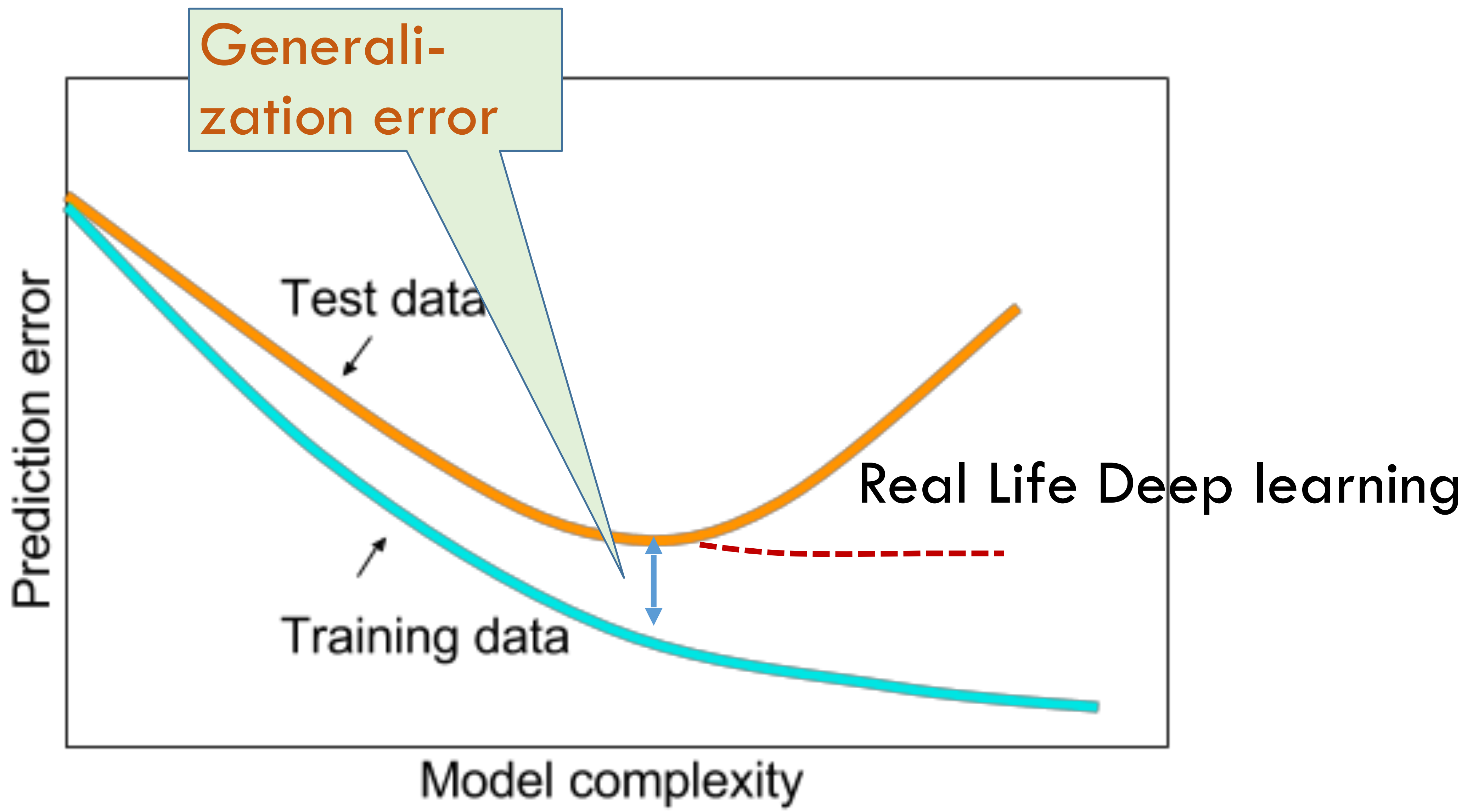
The Parameter Deluge

Training state of the art ConvNets for CIFAR10 (60k images):

	# Parameters	Train Accuracy	Test Accuracy
Inception	1.6M	100	89
Inception v2	1.6M	100	83
Alexnet	1.4M	100	81
MLP 3	1.7M	100	52

Source: Zhang et al. 2017

Large Models Can Overfit



Longtime belief: SGD + regularization eliminates “excess capacity” of the net

Source: Sanjeev Arora ICML 2018 Tutorial

Models that can Fit Anything, Even Noise

Training state of the art ConvNets for CIFAR10 with random labels:

	# Parameters	Train Accuracy	Test Accuracy
Inception	1.6M	100	10
Inception v2	1.6M	100	10
Alexnet	1.4M	100	10
MLP 3	1.7M	100	10

Better Measures of Complexity?

1. A very similar problem arose in the early 2000s for a popular algorithm known as AdaBoost.

2. AdaBoost also fit extremely complex models with very small training data, and had good generalisation power.

3. The key to explaining that was a better notion of complexity than parameter counting: Margin distributions

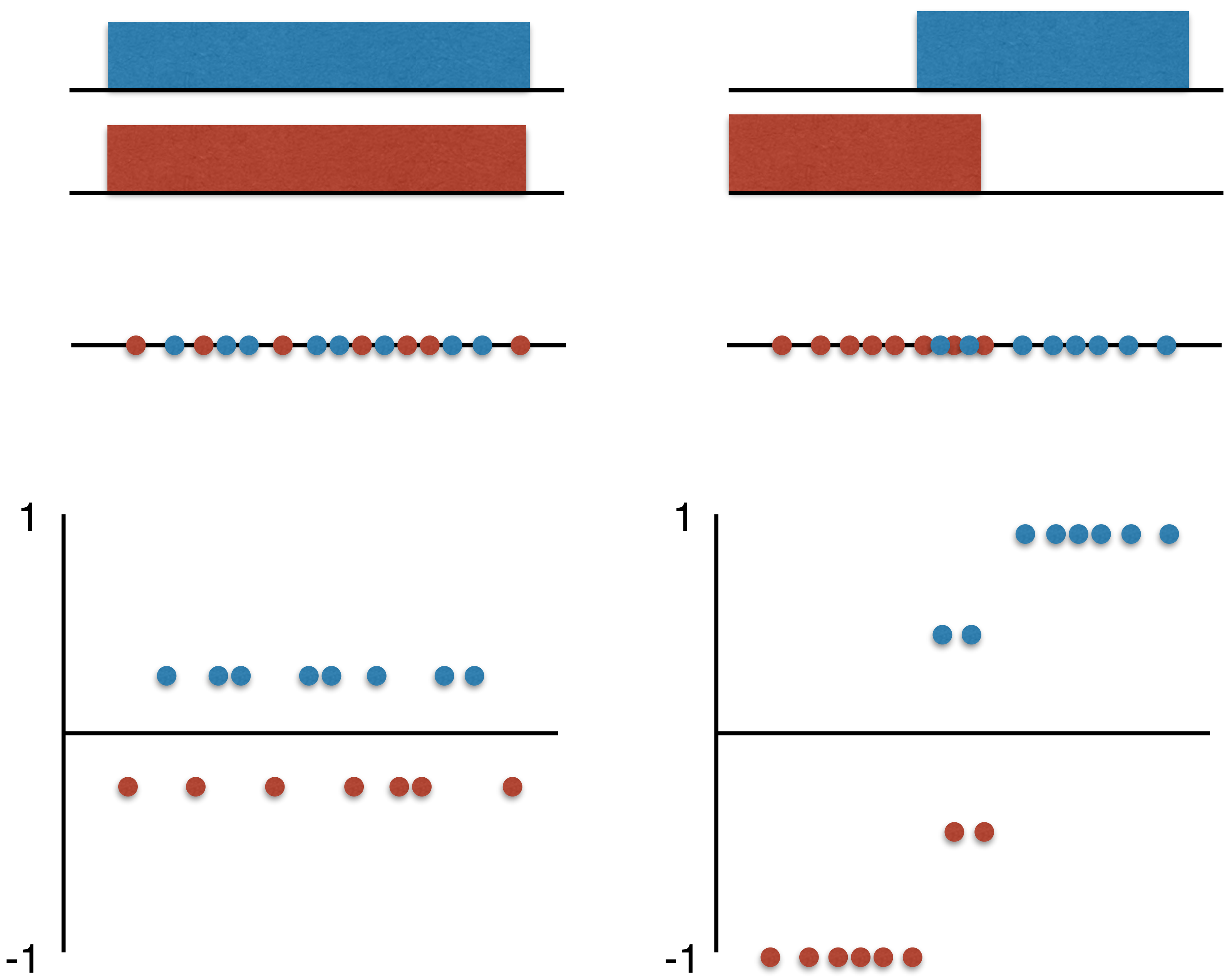
Margin Distributions

Problem: Learn a weighted majority of 10 decision stumps

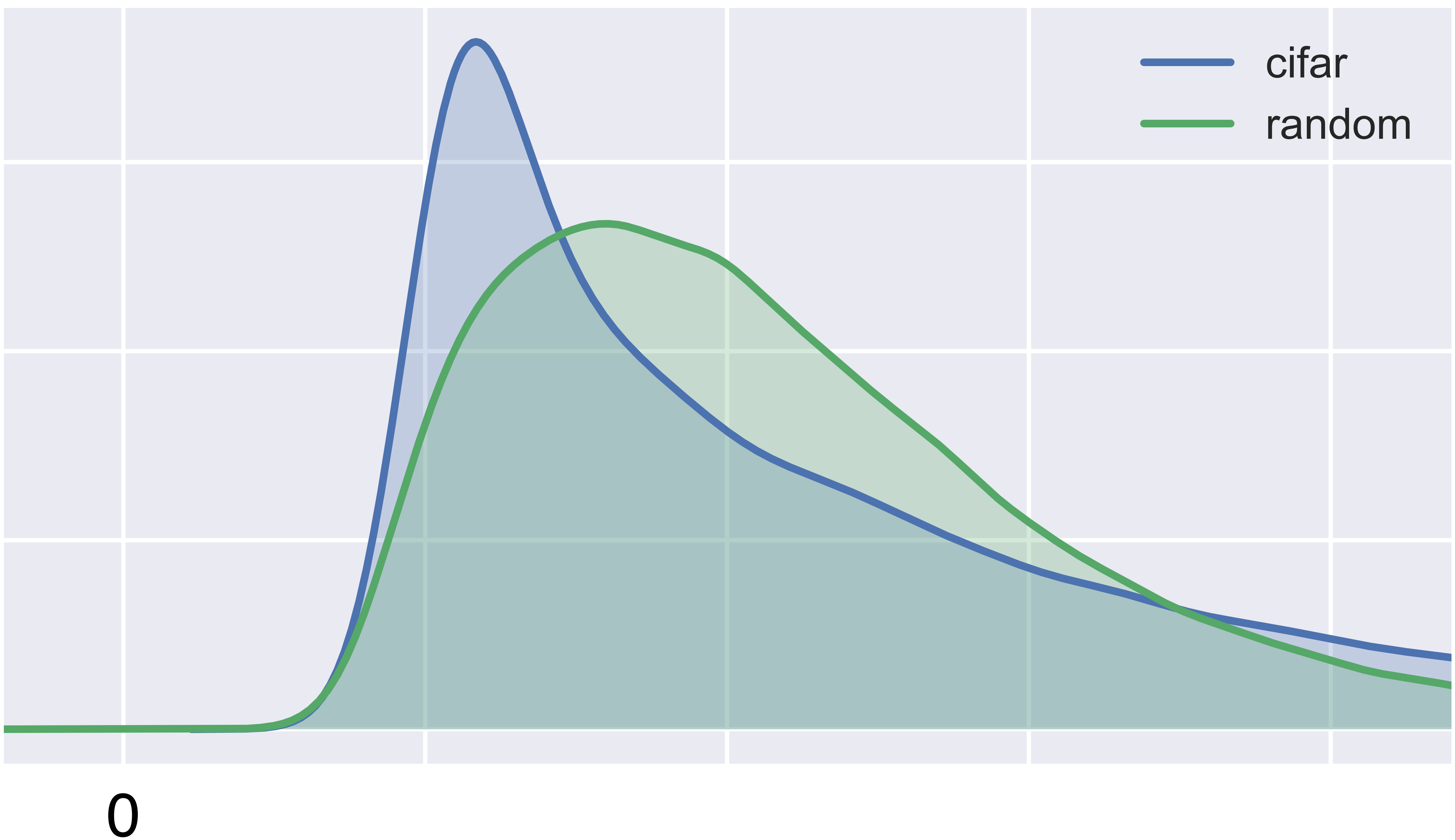
Distribution

Training data

Learned classifier

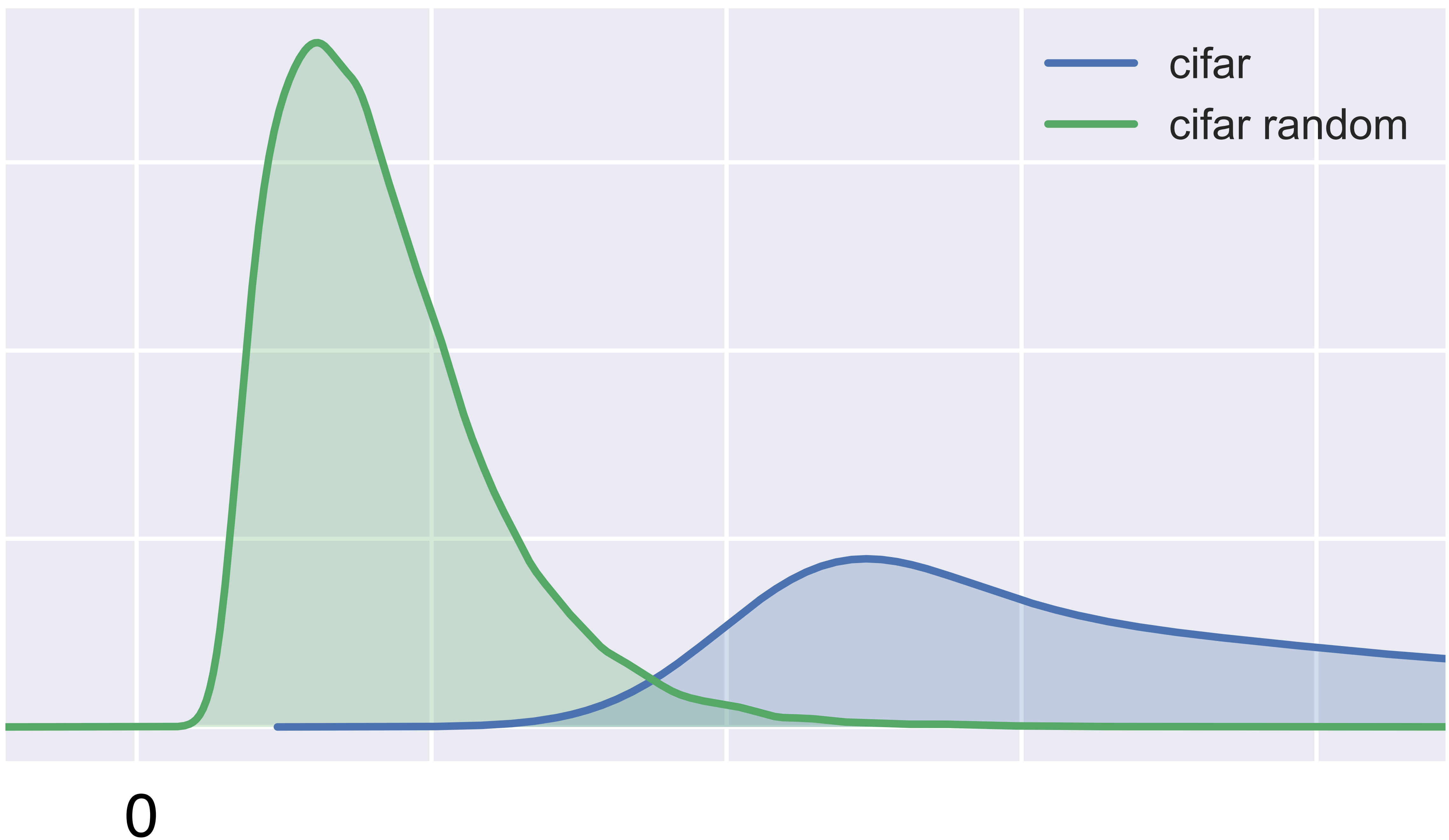


Margin Distributions



Bartlett et al. Spectrally-normalized margin bounds for neural networks.arXiv:1706:08498

Normalised Margin Distributions



Source: Bartlett et al.

Normalising Constant?

$$R_{\mathcal{A}} := \left(\prod_{i=1}^{L} \rho_i \|A_i\|_\sigma \right) \left(\sum_{i=1}^{L} \frac{\|A_i^\top - M_i^\top\|_{2,1}^{2/3}}{\|A_i\|_\sigma^{2/3}} \right)^{3/2}.$$

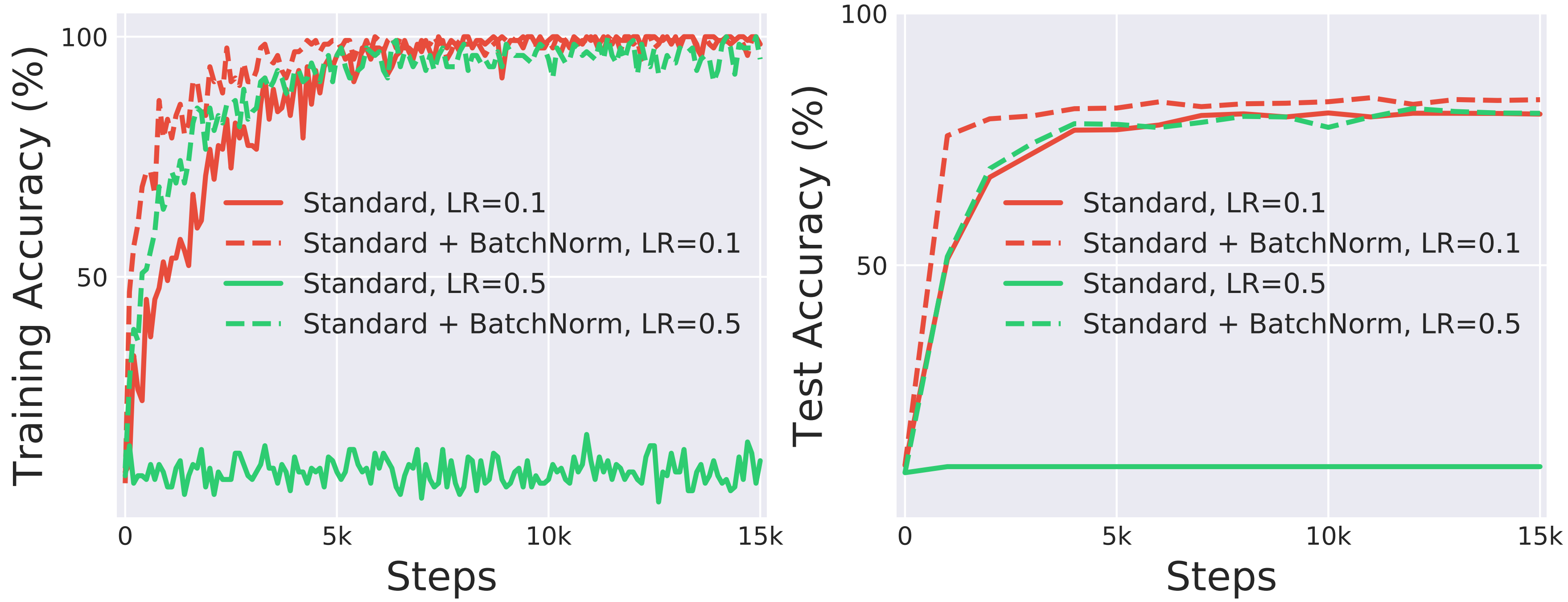
1. Usual log loss does maximise margins.

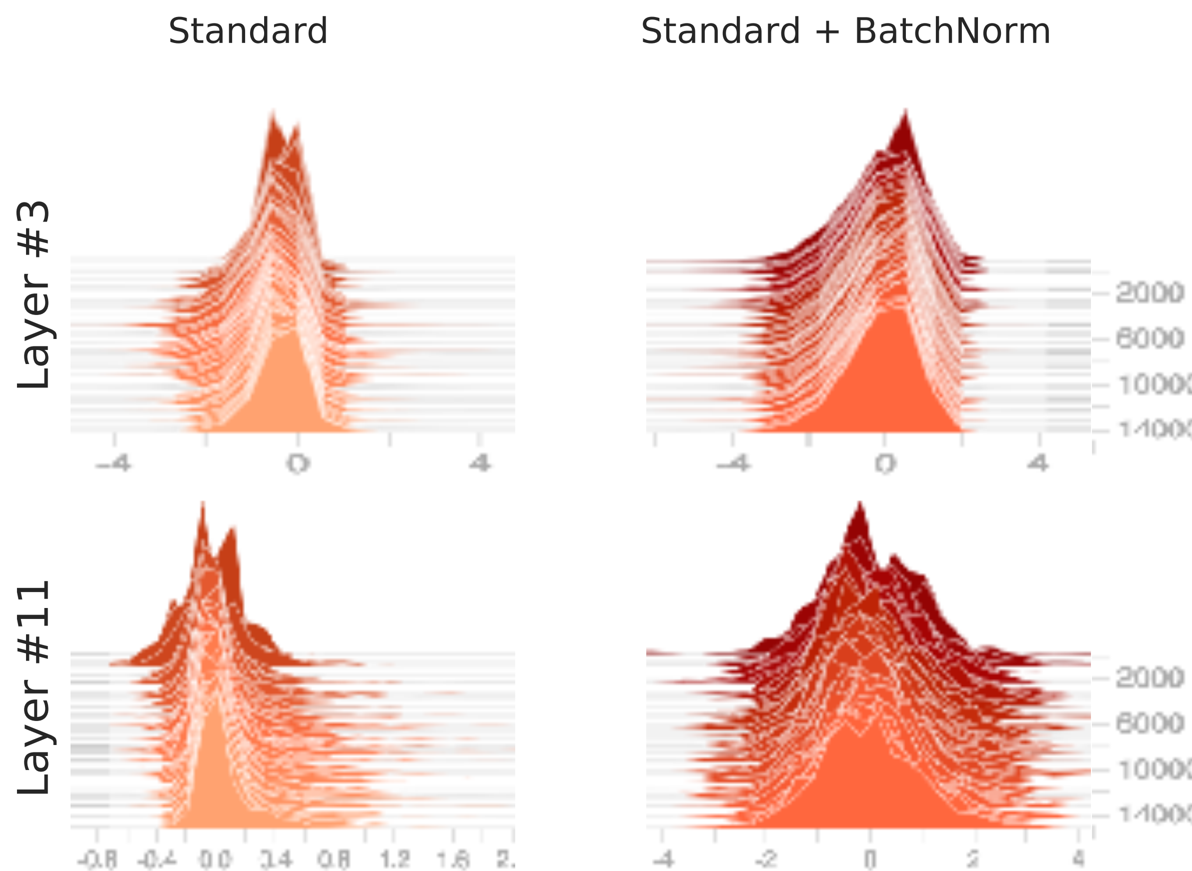
2. But not, “normalised” margins.

3. This immediately suggests a regulariser: the Lipschitz parameter above.

Layer Magic!

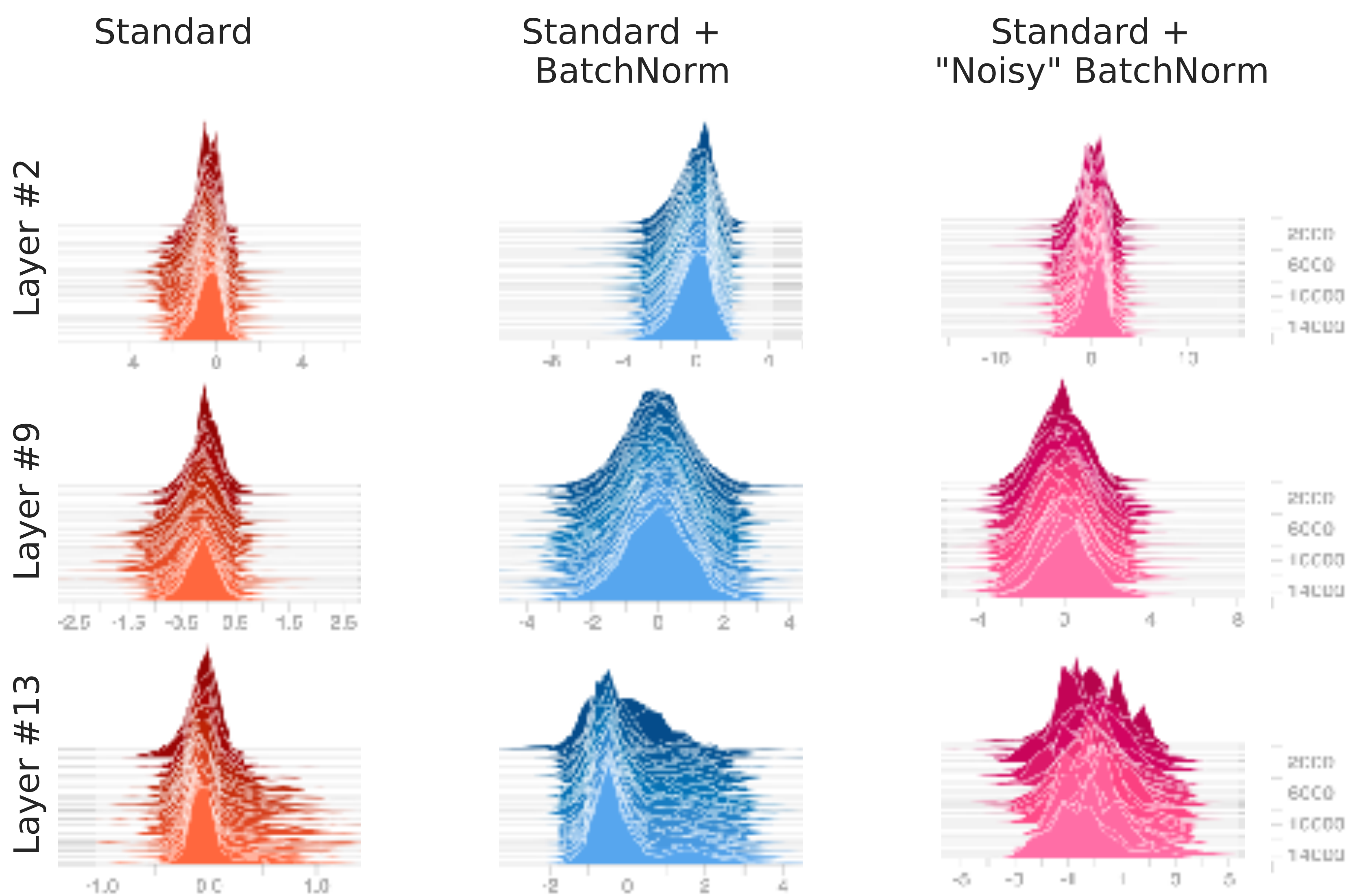
Batch Normalisation: Internal Covariate Shift





Santurkar et al. How Does Batch Normalization Help Optimization?. NIPS 2018

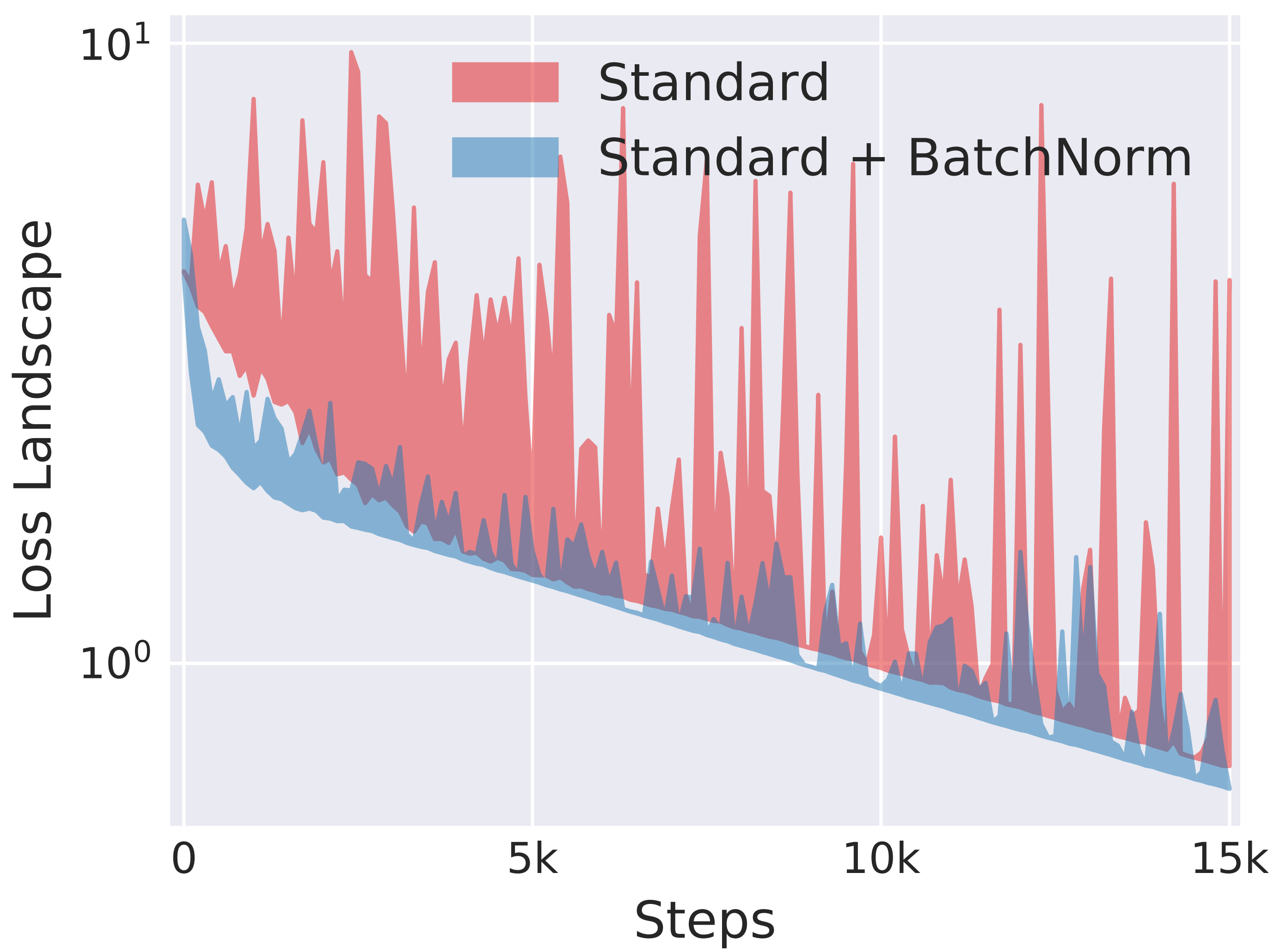
Batch Normalisation: Internal Covariate Shift



Source: Santurkar et al.

Batch Normalisation: Loss Smoother?

How much does the loss change in a direction,
with and without Batch-Norm?



Source: Santurkar et al.