
Learning Nonlinear Representations

David I. Inouye

| Learned features expand **what the model can represent**

Arc 5 • Section 1 of 4

1. **Representation: build nonlinear features** ← **this section**
2. Training: learn the parameters
3. Image structure: design convolutional layers
4. Evaluation: test the architectural assumptions

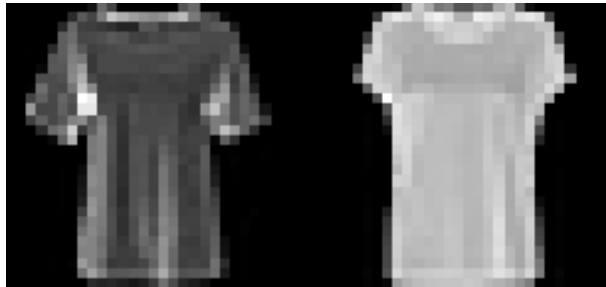
Guiding question: Can learned features improve product-image classification?

Product images → XOR and ReLU → multilayer perceptrons → held-out evidence.

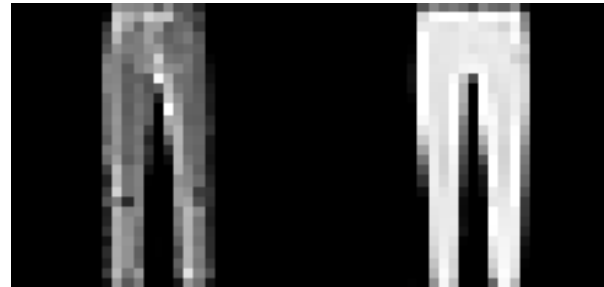
| A product image must become a **useful category**

A catalog task: assign a product image to a category such as shirt, bag, or shoe.

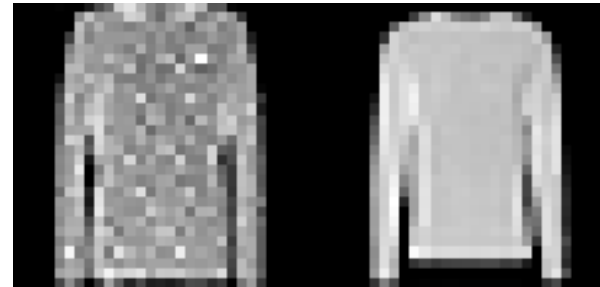
T-shirt/top



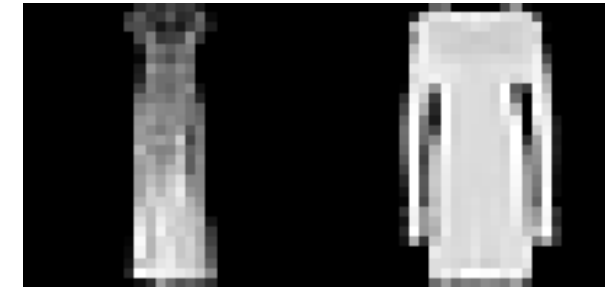
Trouser



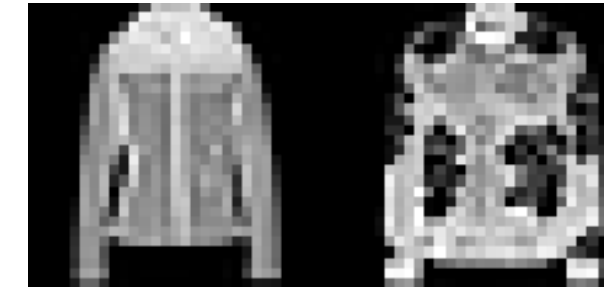
Pullover



Dress



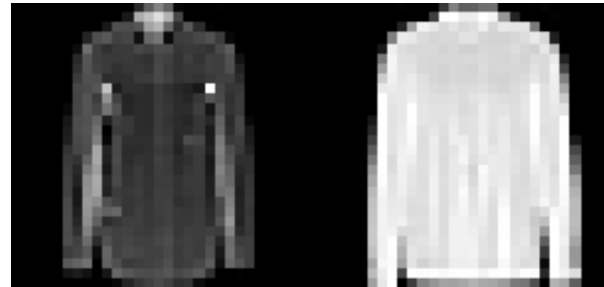
Coat



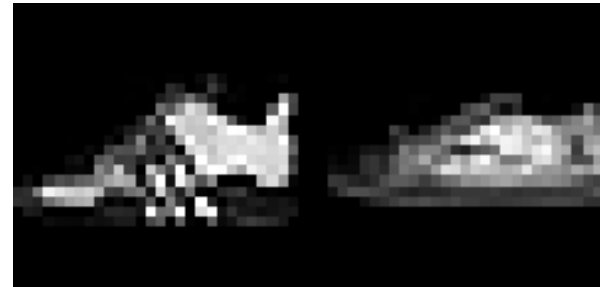
Sandal



Shirt



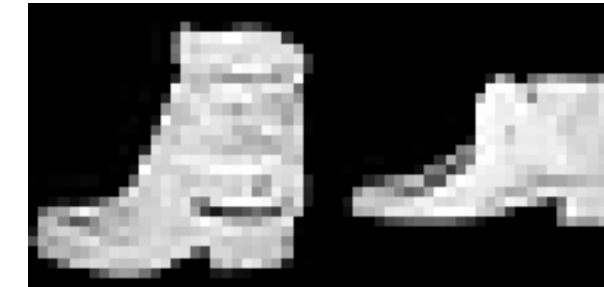
Sneaker



Bag



Ankle boot



Fashion-MNIST: 28×28 grayscale images, **784 pixel features**, ten categories.

Zalando Research, Fashion-MNIST

| Softmax and cross-entropy extend binary logistic regression

For image $x \in [0, 1]^{784}$ and category $c = 1, \dots, 10$:

1. Compute a linear score: $s_c = w_c^\top x + b_c$.

2. Apply softmax to turn the scores into probabilities that sum to one:

$$p_c = \frac{\exp(s_c)}{\sum_{j=1}^{10} \exp(s_j)}.$$

3. Use cross-entropy: if the true category is y , penalize low p_y :

$$\ell(y, p) = -\log p_y, \quad J = \frac{1}{n} \sum_{i=1}^n -\log p_{i, y_i}.$$

Softmax generalizes sigmoid; cross-entropy generalizes binary log loss.

| Regularization helps, but a nonlinear model improves validation accuracy

Same 12,000 training and 3,000 validation images; same pixel features.

Model	Training accuracy	Validation accuracy
Linear, $C = 0.01$ (stronger penalty)	84.66%	82.37%
Linear, $C = 0.1$	88.70%	84.03%
Linear, $C = 1$	92.21%	82.73%
Linear, $C = 10$ (weaker penalty)*	94.90%	80.40%
Nonlinear model	94.10%	85.70%

C is inverse regularization strength. *This fit reached its iteration limit.

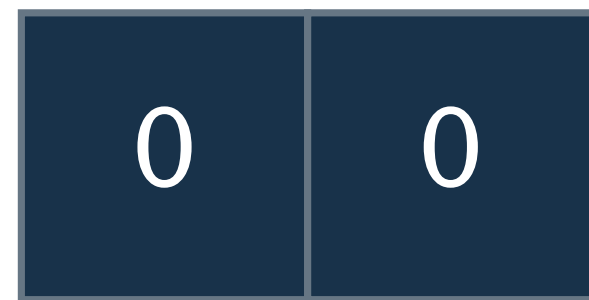
How do we go beyond linear functions, and why can that help?

A tiny image task will make the missing interaction explicit.

| An image label can depend on **relationships between regions**

To isolate the mechanism, use a toy image task: flag a mismatch between two regions.

Equal: $y = 0$



x_1

x_2

Different: $y = 1$



x_1

x_2

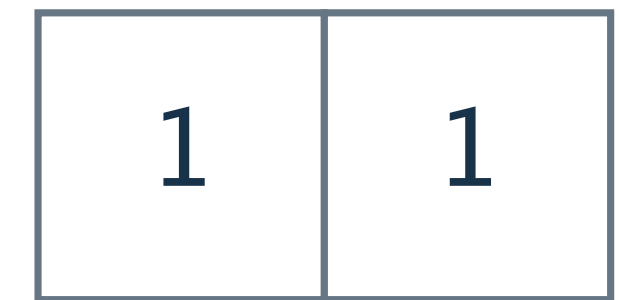
Different: $y = 1$



x_1

x_2

Equal: $y = 0$



x_1

x_2

Represent their brightness by $x = [x_1, x_2]^T \in \{0, 1\}^2$.

Predict $y = 1$ when the regions differ; otherwise predict $y = 0$.

| A raw linear score gives each pixel **one fixed contribution**

$$z = w_1x_1 + w_2x_2 + b, \quad a = \sigma(z).$$

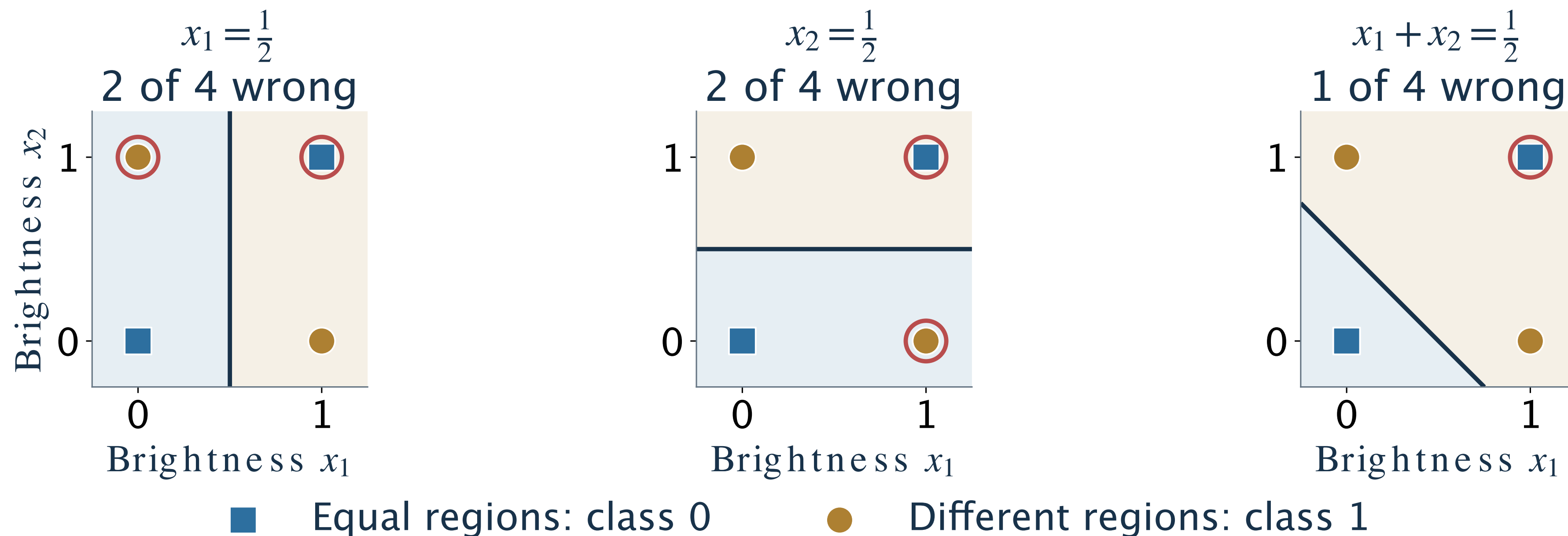
Changing x_1 from 0 to 1 changes the score by w_1 , whatever x_2 is.

Other region	What the mismatch task needs when $x_1 : 0 \rightarrow 1$
$x_2 = 0$	Class changes from 0 to 1
$x_2 = 1$	Class changes from 1 to 0

A monotone sigmoid cannot make the same score increment point both ways.

| XOR cannot be separated by **one straight decision line**

Toy image: two brightness features $x_1, x_2 \in \{0, 1\}$. Label **different regions 1**, equal regions 0.

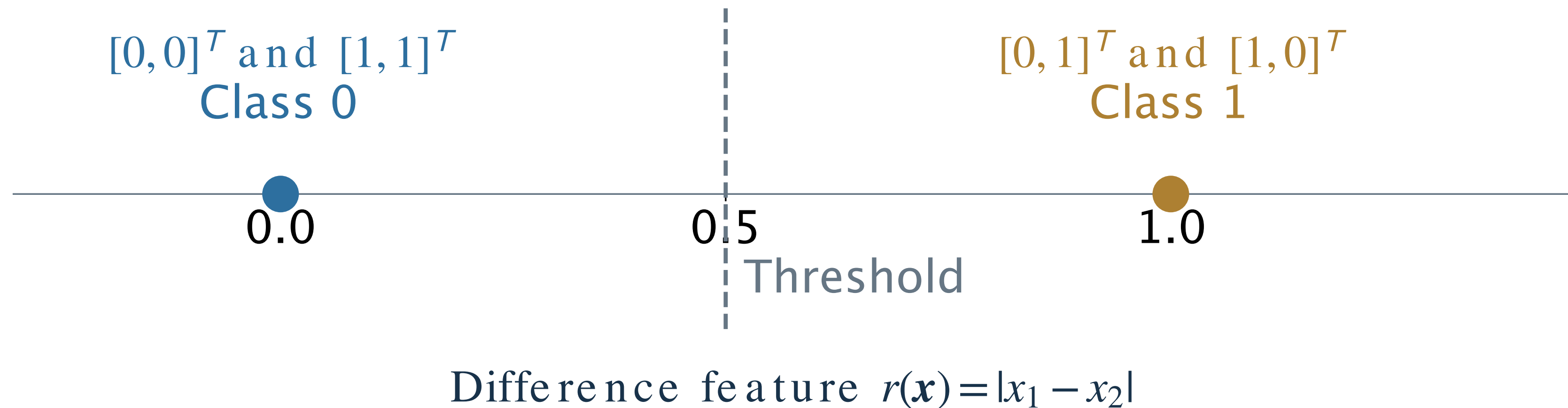


Shading shows predictions; markers show true classes. **Red rings mark errors.**

No single straight decision line separates the two classes.

| A difference feature makes **the same labels easy to separate**

Choose the feature $r(\mathbf{x}) = |x_1 - x_2|$.

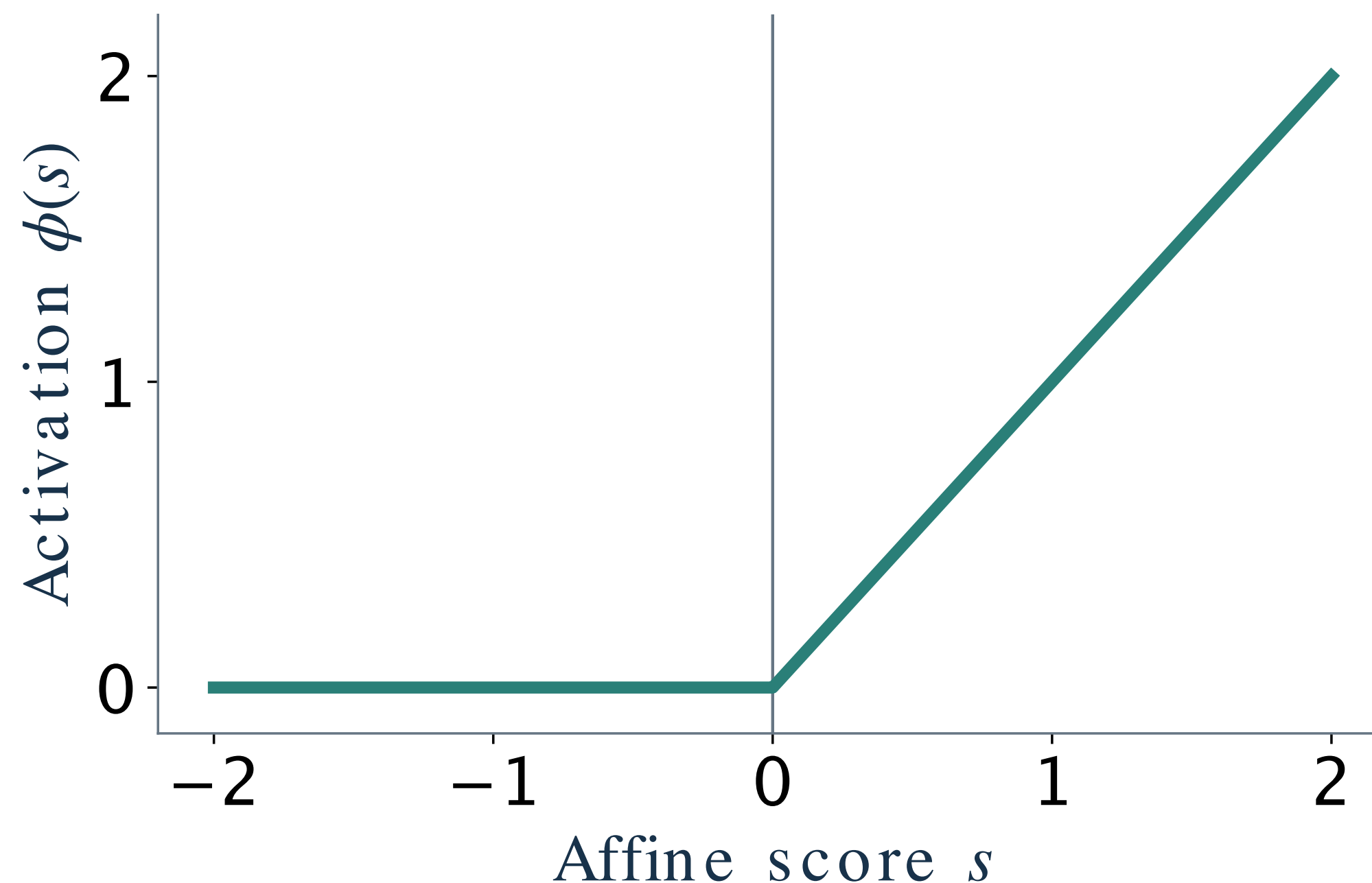


$$\hat{y} = \mathbb{1}\{r(\mathbf{x}) > \frac{1}{2}\}.$$

We changed the **representation**, not the four labels or the fitting algorithm.

| A neuron applies a **nonlinear function** to a **linear score**

$$s = \mathbf{w}^\top \mathbf{x} + b, \quad h = \phi(s).$$



Rectified linear unit (ReLU):

$$\phi(s) = \max(0, s).$$

For $\mathbf{w} = [1, -1]^\top$ and $b = 0$:

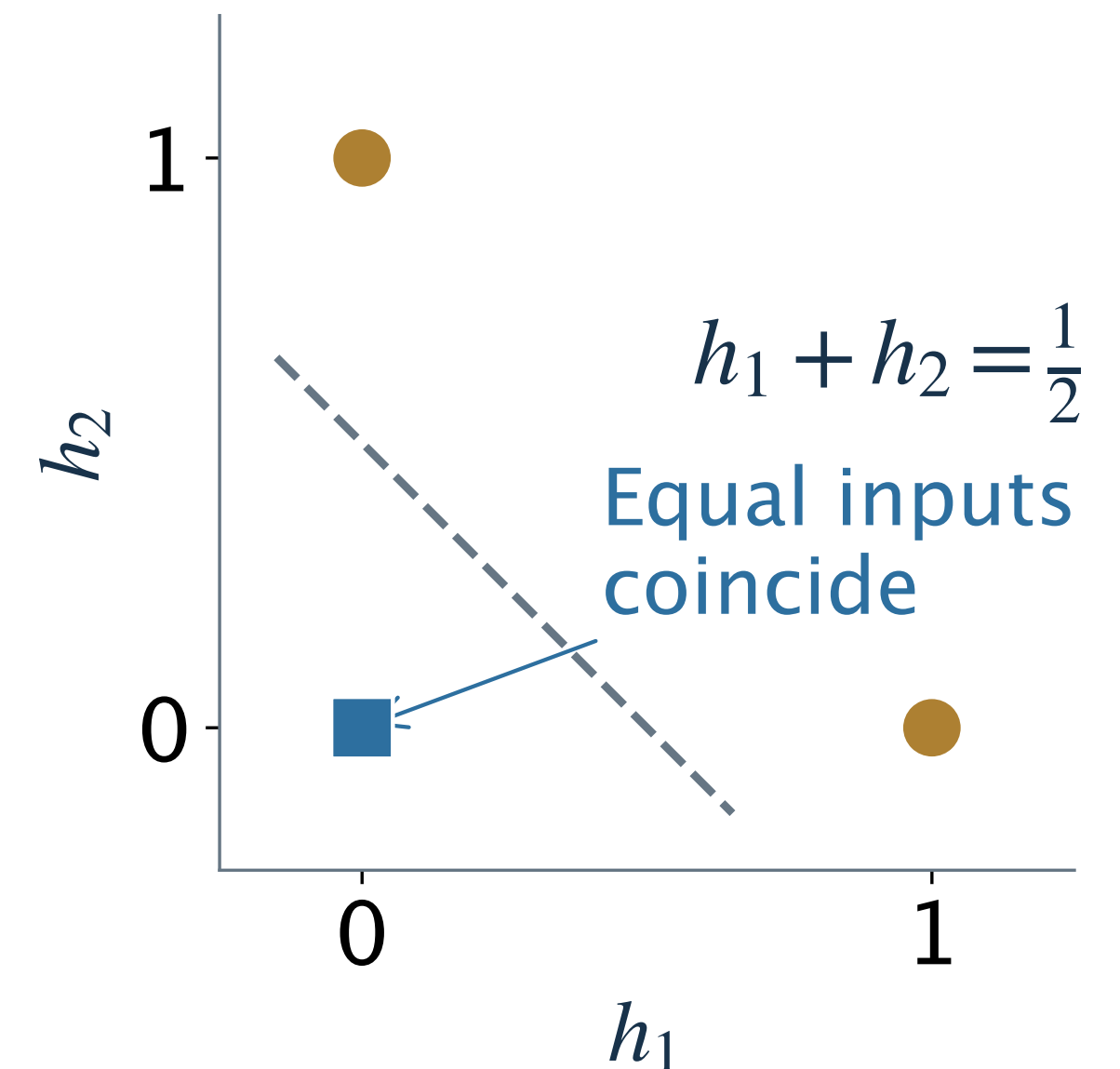
$$h = \max(0, x_1 - x_2).$$

It responds when region 1 is brighter.

| Two directional responses create a useful hidden representation

$$h_1 = \max(0, x_1 - x_2), \quad h_2 = \max(0, x_2 - x_1).$$

Input x	Hidden h	Class y
$[0, 0]^T$	$[0, 0]^T$	0
$[0, 1]^T$	$[0, 1]^T$	1
$[1, 0]^T$	$[1, 0]^T$	1
$[1, 1]^T$	$[0, 0]^T$	0



$$h_1 + h_2 = |x_1 - x_2|, \quad \hat{y} = \mathbb{1}\{h_1 + h_2 > \frac{1}{2}\}.$$

A linear decision in the hidden features solves the nonlinear input-space task.

| A hidden layer collects **several computed features into a vector**

$$h = [h_1, \dots, h_m]^T \in \mathbb{R}^m.$$

Input

x_1

x_2

Hidden representation

$$h_1 = \max(0, x_1 - x_2)$$

$$h_2 = \max(0, x_2 - x_1)$$

Decision

$$h_1 + h_2 > \frac{1}{2}$$

Each hidden unit has its own weights and bias; all see the same input.

The output layer combines the resulting features.

| Stack the neuron weights to obtain **one matrix expression**

Each row of W is one neuron's $w_j^\top$:

$$\boldsymbol{h} = \phi(W\boldsymbol{x} + \boldsymbol{b}), \quad W \in \mathbb{R}^{m \times d}, \quad \boldsymbol{b} \in \mathbb{R}^m.$$

Apply ϕ separately to each feature.

For our two-region example:

$$W = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{h} = \phi \left(\begin{bmatrix} x_1 - x_2 \\ x_2 - x_1 \end{bmatrix} \right).$$

| Dimensions account for every trainable parameter

One Fashion-MNIST image: 784 pixel features $\rightarrow$ 64 hidden units $\rightarrow$ 10 class scores.

$$h = \text{ReLU}(Wx + b), \quad s = Vh + c.$$

Layer	Weight matrix	Bias vector	Trainable parameters
Hidden layer	$W : 64 \times 784$	$b : 64 \times 1$	$64 \cdot 784 + 64 = 50,240$
Output layer	$V : 10 \times 64$	$c : 10 \times 1$	$10 \cdot 64 + 10 = 650$

Total: $50,240 + 650 = 50,890$ trainable parameters.

Activations: $x \in \mathbb{R}^{784}, h \in \mathbb{R}^{64}, s, p \in \mathbb{R}^{10}$. Softmax converts scores s to probabilities p and adds **no parameters**.

The output layer maps learned features to **multiclass probabilities**

For the product images, let $\mathbf{h} \in \mathbb{R}^{64}$ and compute

$$\mathbf{s} = V\mathbf{h} + \mathbf{c}, \quad V \in \mathbb{R}^{10 \times 64}, \quad \mathbf{c} \in \mathbb{R}^{10}.$$

Reuse softmax to normalize the ten scores into class probabilities:

$$p_c = \frac{\exp(s_c)}{\sum_{j=1}^{10} \exp(s_j)}, \quad c = 1, \dots, 10.$$

For class labels y_i , fit all parameters $\boldsymbol{\theta} = (W, \mathbf{b}, V, \mathbf{c})$ jointly using **mean cross-entropy**:

$$J(\boldsymbol{\theta}) = -\frac{1}{n} \sum_i \log p_{i,y_i}.$$

| Deeper models **compose learned representations**

$$\mathbf{h}^{(0)} = \mathbf{x},$$

$$\mathbf{h}^{(\ell)} = \phi_{\ell}(\mathbf{W}_{\ell}\mathbf{h}^{(\ell-1)} + \mathbf{b}_{\ell}), \quad \ell = 1, \dots, k.$$

$$\mathbf{x} \longrightarrow \mathbf{h}^{(1)} \longrightarrow \mathbf{h}^{(2)} \longrightarrow \text{prediction}.$$

Each layer transforms features produced by the previous layer.

An MLP—multilayer perceptron—uses dense layers with nonlinear hidden activations.

| Would extra linear layers solve **the mismatch task**?

A network has several hidden layers. Replace every hidden activation with the identity; compare the two final class scores.

Can it classify all four mismatch images correctly?

A. Yes, if the hidden layers are wide enough.

B. Yes, if gradient descent runs long enough.

C. No; the difference between its two final scores is still a linear score in the input.

C. Additional linear layers cannot supply the missing interaction.

| Composing linear layers **still gives one linear layer**

A **linear layer** computes $Wx + b$ (technically *affine* when $b \neq 0$).

$$f(x) = W_2(W_1x + b_1) + b_2$$

(two linear layers)

$$= W_2W_1x + W_2b_1 + b_2$$

(distribute)

$$= Ax + c$$

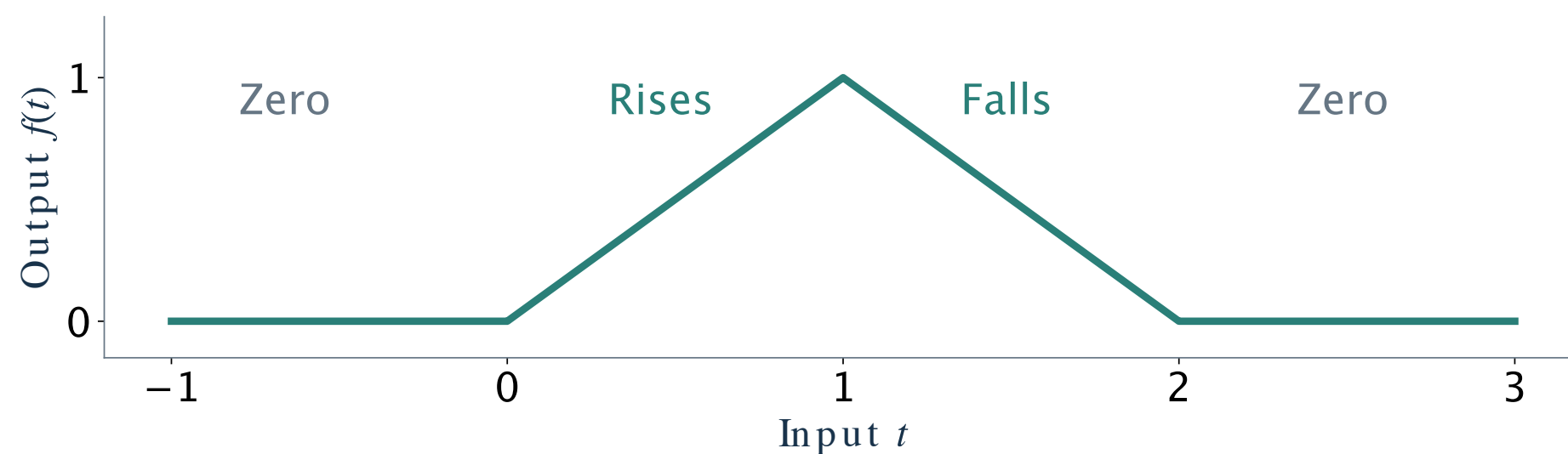
($A = W_2W_1, c = W_2b_1 + b_2$)

Repeat this argument for any number of linear layers.

Nonlinear activations let simple pieces form a new shape

$$f(t) = \phi(t) - 2\phi(t - 1) + \phi(t - 2), \quad \phi(s) = \max(0, s).$$

The three ReLU units switch on at $t = 0, 1, 2$; the output weights are $1, -2, 1$.



Input interval	Substitute the active units
$t \leq 0$	$f(t) = 0$
$0 < t \leq 1$	$f(t) = t$
$1 < t \leq 2$	$f(t) = t - 2(t - 1) = 2 - t$
$t > 2$	$f(t) = t - 2(t - 1) + (t - 2) = 0$

At $t = 1.5$: $f(t) = 1.5 - 2(0.5) + 0 = 0.5$.

Each newly active unit changes the slope: $0 \rightarrow 1 \rightarrow -1 \rightarrow 0$. Three ReLU features plus a linear output layer create a shape no single linear layer can express.

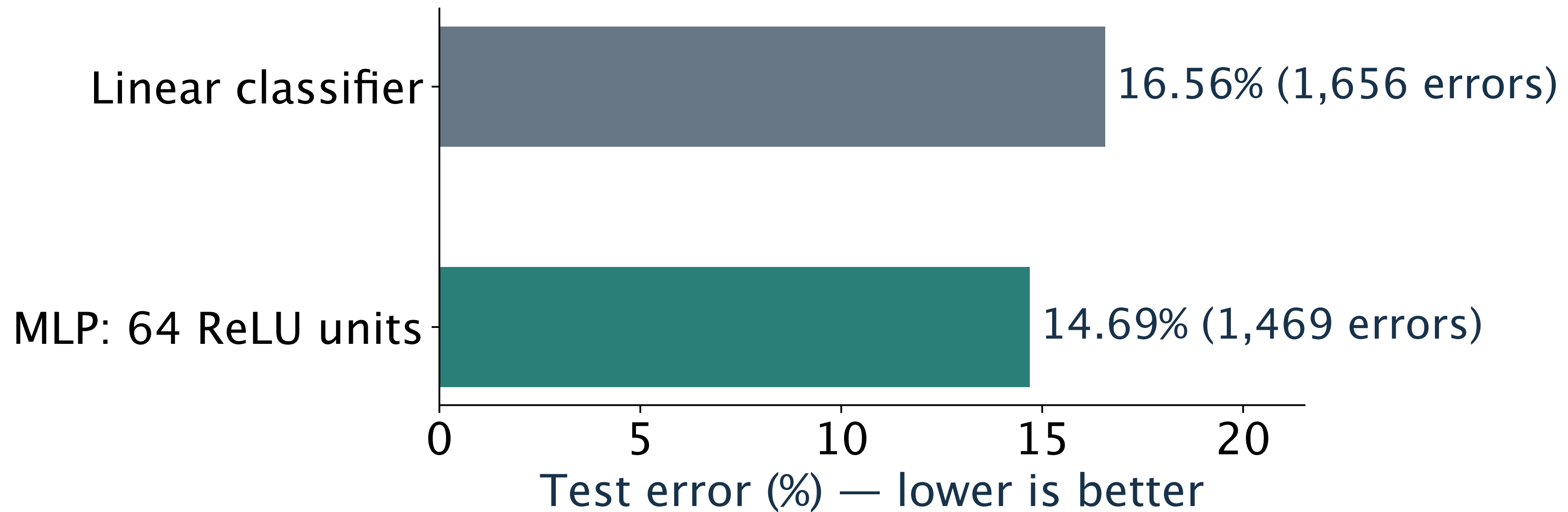
| More expressive features still need **held-out evidence**

Question	Evidence needed
Can this architecture express a useful rule?	A construction or approximation argument
Did training find useful parameters?	Training objective and optimization diagnostics
Does the fitted predictor work on new examples?	Appropriate held-out evaluation

Choose width, depth, and regularization using validation data; reserve test data for final assessment.

Our four-image construction answers only the first question.

| A small MLP makes **fewer product-image errors** in this experiment



187 fewer errors on 10,000 test images. Same inputs and split; each model selected on validation data.

| The final loss must guide **every layer's parameters**

Arc 5 • Section 2 of 4

1. Representation: build nonlinear features
2. **Training: learn the parameters ← this section**
3. Image structure: design convolutional layers
4. Evaluation: test the architectural assumptions

Guiding question: How can a final loss train earlier layers?

Gradient descent → atomic gradient rules → backpropagation → training diagnostics.

| Image structure motivates **local connections and shared weights**

Arc 5 • Section 3 of 4

1. Representation: build nonlinear features
2. Training: learn the parameters
3. **Image structure: design convolutional layers** ← **this section**
4. Evaluation: test the architectural assumptions

Guiding question: What image structure should an architecture use?

Neighborhoods → convolution → channels and shapes → receptive fields.

Preview of a later section of the arc.

| Architectural assumptions need empirical evidence

Arc 5 • Section 4 of 4

1. Representation: build nonlinear features
2. Training: learn the parameters
3. Image structure: design convolutional layers
4. **Evaluation: test the architectural assumptions ← this section**

Guiding question: When does architectural bias help?

Controlled MLP/CNN comparison → errors and cost → translation stress test.

Preview of a later section of the arc.