CSC2516
Neural Networks + Deep Learning
Fall 2023

0 vs. 1

```
n_dark = 0
for pixel in row:
    if pixel is black:
        n_dark += 1
if n_dark > 1:
    return 0
else:
    return 1
```

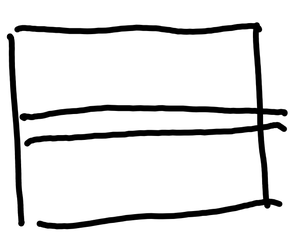
0 1 2 3 4 5 6 7 8 9

Machine Learning (ML):
program whose behavior is
"learned" from examples.

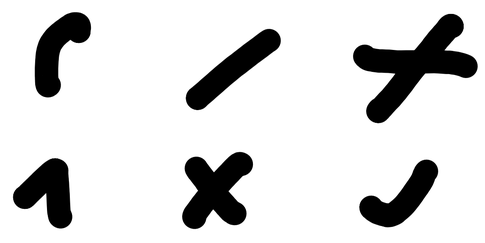
Old-fashioned ML:
1. Design features
2. Train a "simple" model

0 0 1 8 8 7

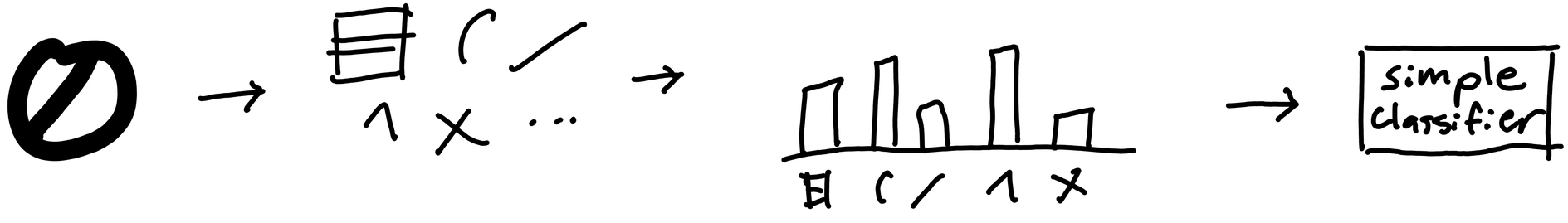
Feature: Does the image have some property (or how much)?

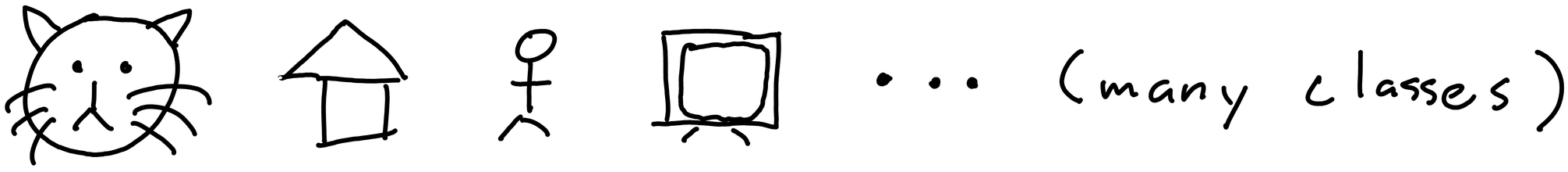


dark pixels

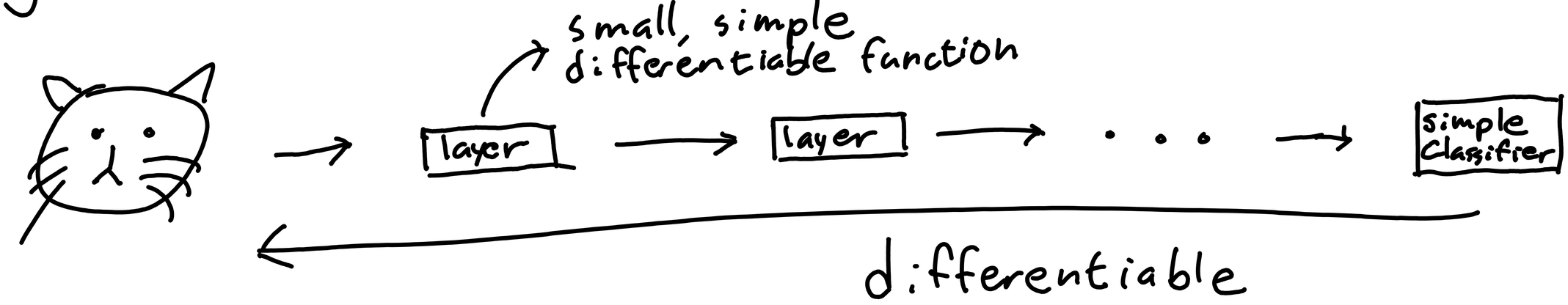


does the image have this shape?

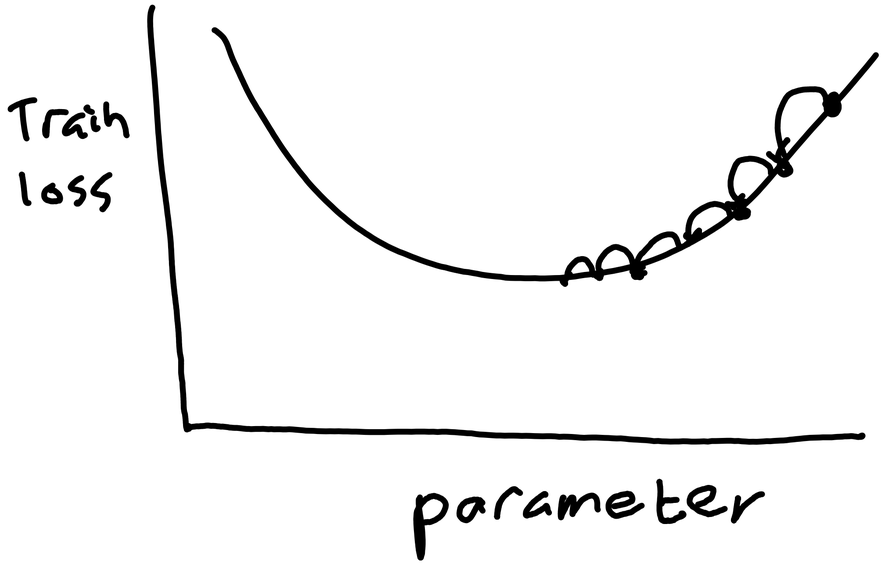




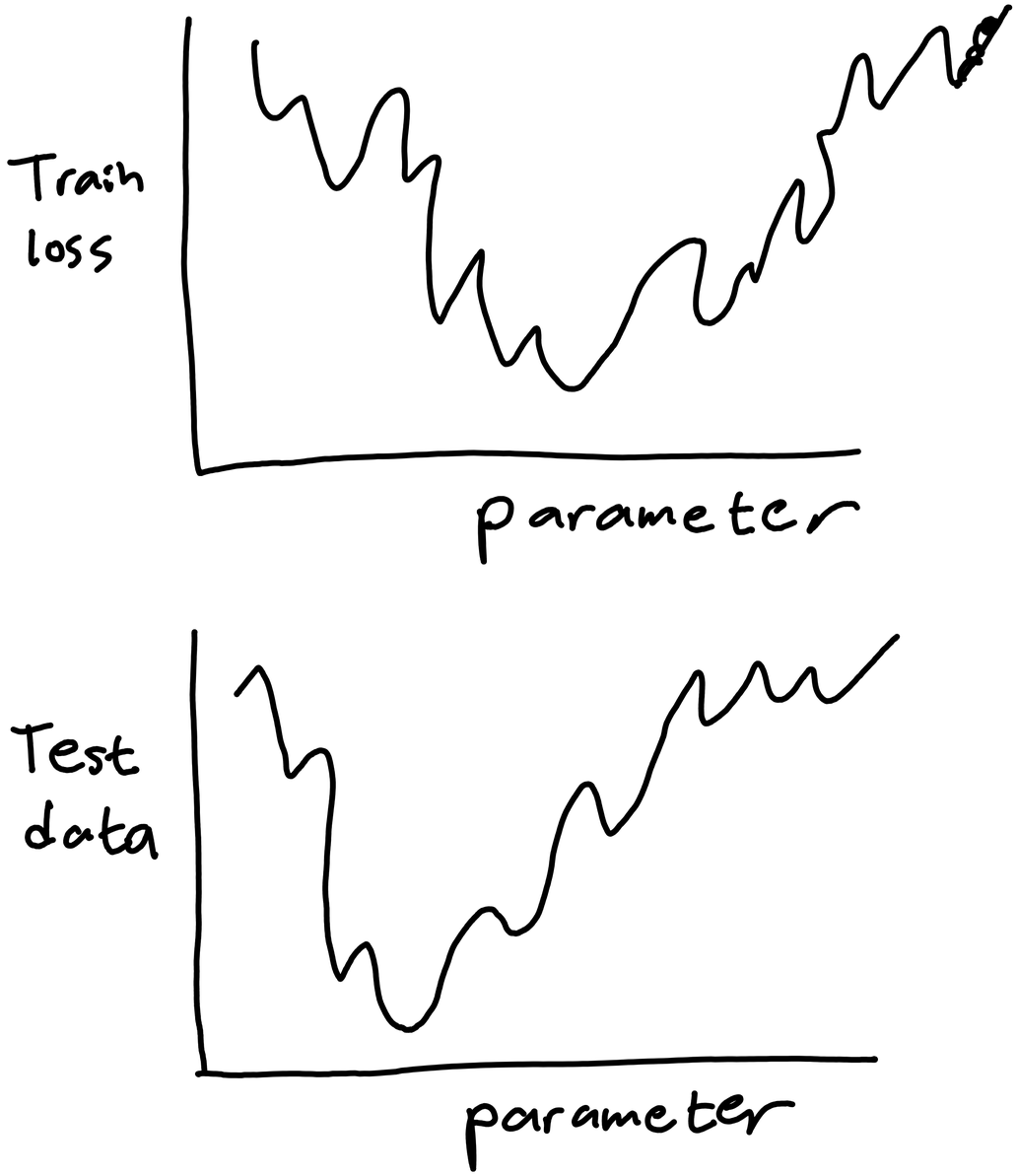
Deep learning: Learn features from data using big, complex, differentiable models that can be trained end-to-end with gradient descent.



Old-fashioned ML



Deep Learning



regularization, inductive bias, pre-training, ...

Linear Regression

Linear model that predicts a scalar (↳ a single real number) from some inputs.

ex/ Predict house value from

age, sq.ft, #bedrooms, lot size, ...

$$\text{price} = \underbrace{w_1 \text{age} + w_2 \text{sqft} + w_3 \text{nbr}}_{\text{features}} + \underbrace{b}_{\text{parameter}}$$

$$x = [\text{age}, \text{sqft}, \text{nbr}]$$

$$w = [w_1, w_2, w_3]$$

$$\text{price} = w^T x + b$$

$$\hat{y} = w^T x + b$$

$$\hat{Y} = X w + b$$

$\hat{Y} \in \mathbb{R}^{n\text{-data}}$

$X \in \mathbb{R}^{n\text{-data} \times n\text{-features}}$

training dataset

Goal: fit the parameters (w, b) using our training data.

How well are we doing?

How closely are predicted $\hat{y}$ to the true values in the training set?

Loss function:

$$\frac{1}{2}(\hat{y} - y)^2$$

prediction, target

squared error

Goal: Change parameters to minimize the loss over the training dataset.

Solve? $w^* = (X^T X)^{-1} X^T y$

Use gradient descent!

Gradient descent:

1. Initialize parameters θ (w, b)
2. Repeat:

$$\theta \leftarrow \theta - \eta \nabla_\theta L$$

θ: new param values; θ: old param values; η: learning rate; $\nabla_\theta L$: gradient of loss (L) w.r.t. θ

Until some criterion is met.

~~ex~~ loss is zero. (rare)

~~ex~~ loss stops decreasing.

~~ex~~ performance on held-out data stops improving.

~~ex~~ run out of compute or patience.

$$\hat{y} = w^T x + b$$

$$L = \tfrac{1}{2}(\hat{y} - y)^2$$

(x, y) is a single training datapoint.

We need $\nabla_\theta L$ ($\nabla_w L$ and $\nabla_b L$).

Use the chain rule!

Functions: $L(\hat{y})$, $\hat{y}(w)$ (or $\hat{y}(b)$)

$$\frac{dL}{dw} = \frac{dL}{d\hat{y}} \frac{d\hat{y}}{dw}$$

$$\frac{dL}{db} = \frac{dL}{d\hat{y}} \frac{d\hat{y}}{db}$$

$$\hat{y} = w^T x + b$$

$$L = \frac{1}{2}(\hat{y} - y)^2$$

$$\frac{dL}{d\hat{y}} = \hat{y} - y$$

$$\frac{d\hat{y}}{dw} = x$$

$$\frac{d\hat{y}}{db} = 1$$

$\nabla_w L$

$$\frac{dL}{dw} = \frac{dL}{d\hat{y}} \frac{d\hat{y}}{dw} = (\hat{y} - y)x$$

$$\frac{dL}{db} = \frac{dL}{d\hat{y}} \frac{d\hat{y}}{db} = \hat{y} - y$$

$\nabla_b L$

$$w \leftarrow w - \eta \nabla_w L$$

use 1 example → stochastic gradient descent

"batch" X, a matrix of examples

$x \in \mathbb{R}^{n\_features}$, $X \in \mathbb{R}^{n\_data \times n\_features}$

$$w \leftarrow w - \frac{n}{n\_data} \sum_{n\_data} \nabla_w L$$

→ random subset of training set

"minibatch stochastic gradient descent"

$$w \leftarrow w - \frac{\eta}{n\_data} \sum_{n\_data} \nabla_w L$$

Hyperparameters:

Values set "by hand" instead of GD.

1. η learning rate (step size)
2. "batch size" (n_data)
3. Stopping criterion
4. Initialization scheme