1. Image Classification
- Input: ảnh (tensor pixel giá trị 0 - 255, RGB HxWx3)
- Ouput: label trong một tập nhãn cho trước
- Sematic gap: máy chỉ thấy được số pixel, không phải khái niệm giống con người

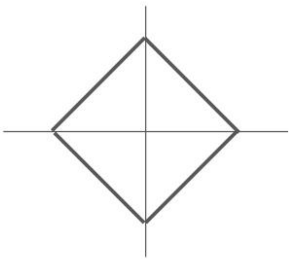
2. Các thách thức
- Đổi góc chụp -> pixel bị đổi - Viewpoint variation
- Ánh sáng khác nhau -> pixel khác nhau - Illumination
- Nền rối gây nhiễu - Background clutter
- Bị che - Occlusion
- Vật biến dạng hoặc có tư thế khác - Deformation
- Cùng chung một lớp nhưng có các kiểu khác nhau(size, màu,…) - Intraclass variation
- Ngữ cảnh ảnh đánh lừa - Context

3. Data-Driven Approach
- Thu thập dataset + label
- Train model
- Evaluate trên ảnh mới

4. KNN (K - Nearest Neighbor)
- Ý tưởng: ghi nhớ toàn bộ training set, dự đoán bằng K ảnh gần nhất
-> có 2 phần chính train và predict
- Độ phức tạp: train O(1) vì chỉ lưu và predict O(n) vì so với tất cả các train
- Distance metrix có 2 loại:
    + L1(Mahattan) tổng |chênh lệch| theo từng pixel - cái này sẽ giống việc di chuyển trong 1 thành phố với từng ô vuông hơn
    + L2(Euclidean) khoảng cách thẳng

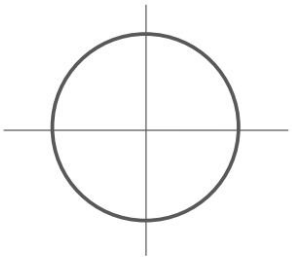
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



=> L1 sẽ có nhiều cạnh vuông hơn L2

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

K = 1

L2 (Euclidean) distance

$$d_2(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

K = 1

5. Hyperparameters và cách chọn đúng

- Hyperparameters: K, loại distance, …
- Cách chọn:

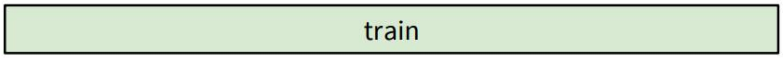
+ Cách 1: Chỉ chọn hyperparameter hoạt động tới trên tập training -> overfitting(vd K = 1 luôn luôn đúng hết cho mọi tập data)

+ Cách 2: Chọn theo test: "leak data" -> không biết performance thật

+ Cách 3: train/val/test, tune trên val, chỉ dùng test 1 lần cuối -> hợp lý nhất

Idea #1: Choose hyperparameters that work best on the training data

BAD: K = 1 always works perfectly on training data



Idea #2: choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

Idea #3: Split data into train, val; choose hyperparameters on val and evaluate on test

Better!

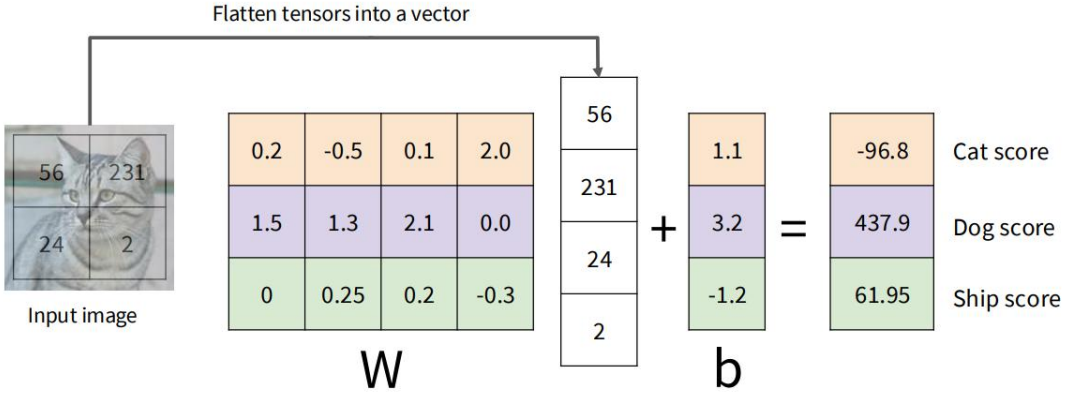
- Có thể dùng cross - validation để chia nhỏ dữ liệu: Chia data thành các folds, mỗi folds sẽ được validation và phần còn lại sẽ là training test -> tính average accuracy

6. Linear Classifier

- Model: ảnh được "flatten" thành vector x; dự đoán score cho từng lớp:

$$\mathbf{f(x, W) = Wx + b}$$

- Output: vector score(xác suất chính xác cho từng lớp)



- Linear Layer thường được sử dụng ở bước cuối của các mô hình phân tích ảnh

7. Muốn học W tốt: cần Loss + Optimization

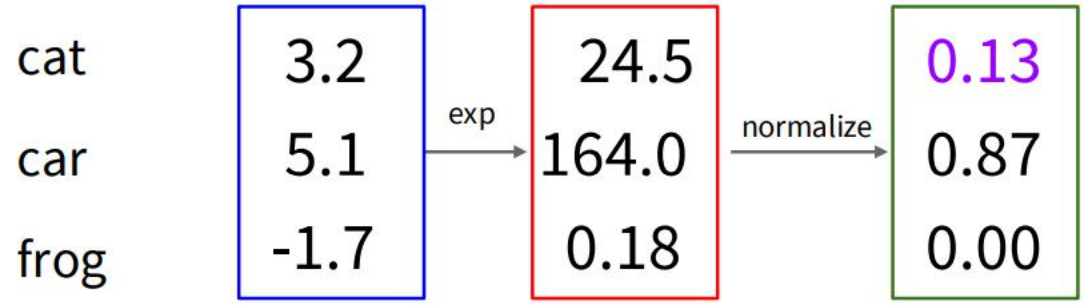
- 2 bước chính:
  + Define loss function đo sai số của scores trên train data
  + Optimize để tìm W làm loss nhỏ nhất
- Loss toàn dataset thường là trung bình loss từng mẫu

$$L = \frac{1}{N}\sum_i L_i(f(x_i, W), y_i)$$

Where: xi là image, yi là label

8. Softmax Classifier

- Mục tiêu: biến scores thành probabilities bằng softmax (exp + normalize)



  + Bước exp: Biến các xác suất lớn hơn 0
  + Bước normalize: Biến tổng các xác suất bằng 1

-> Công thức Loss cuối cùng:

$$L_i = -\log P(Y = y_i \mid X = x_i) = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

-> Dùng thêm log ở đây để phạt mạnh nếu như class đúng quá nhỏ giúp gradient học nhanh và hội tụ nhanh hơn