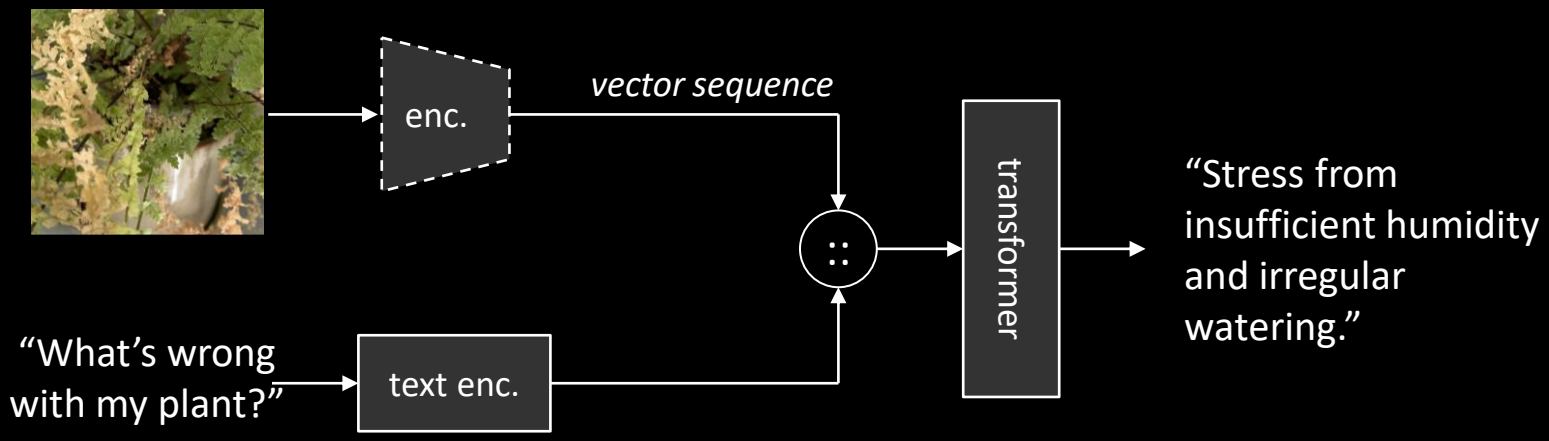
# Three applications of critics

1. Training generators
   adversarial training
2. Image encoders
   contrastive learning
3. Out-of-distribution inputs

**If we can encode an image as a vector sequence, an LLM will be able to 'see' it.**
We want to convert the pixels into a useful condensed format, while retaining their relevant information.




It's simpler and cheaper to build it standalone, and easier to debug

**Design choices.**
- Train the whole system jointly from scratch, or train a standalone image captioner?
- Supervised learning with manually labelled data, or unsupervised with scraped images + surrounding text?

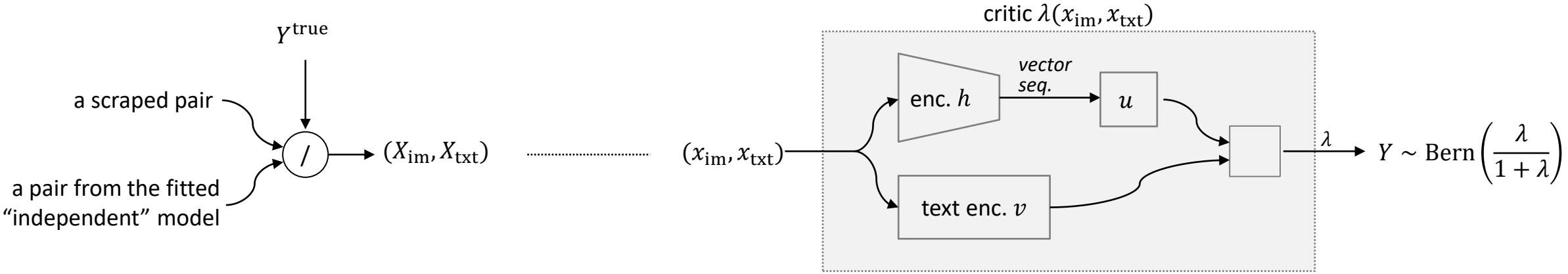
with this data, we can train "extract features relevant for textual description", perhaps not "image captioner"

**Problem formulation.** We want to
- learn an encoder function $h$: image $\mapsto$ vector sequence
- that retains the information relevant for textual description

Let $(X_{\text{im}}, X_{\text{txt}}) \sim q$ be an image and its surrounding text, as sampled from scraped web data.

***Cunning idea:*** *to measure how image and text are related, propose a model in which they're independent and ask a critic to judge it.*

What is the best-fitting model **such that** $X_{\text{im}}$ and $X_{\text{txt}}$ are independent? How can we sample a pair from it?




- h is the encoder we want to learn
- we want the critic to consider $h(x_{im})$ with $x_{txt}$ together
- simplest possible linkage: dot-product them, after making them equal-dim vectors

**Train the critic** as a classifier, to classify well when applied to a random pair:

$$\max_{\lambda}\{\pi \mathbb{E}_{\text{true}} \log D + (1-\pi)\mathbb{E}_{\text{model}} \log(1-D)\}$$

where $D := \Lambda/(1+\Lambda)$ and $\Lambda := \lambda(X_{\text{im}}, X_{\text{txt}})$ and $\pi = \mathbb{P}(Y^{\text{true}} = 1)$

**Design the critic's internals** so that it's forced to learn an encoder $h$ that we can reuse:

$$\lambda(x_{\text{im}}, x_{\text{txt}}) = \exp\left[u\left(h(x_{\text{im}})\right) \cdot v(x_{\text{txt}}) + b\right]$$

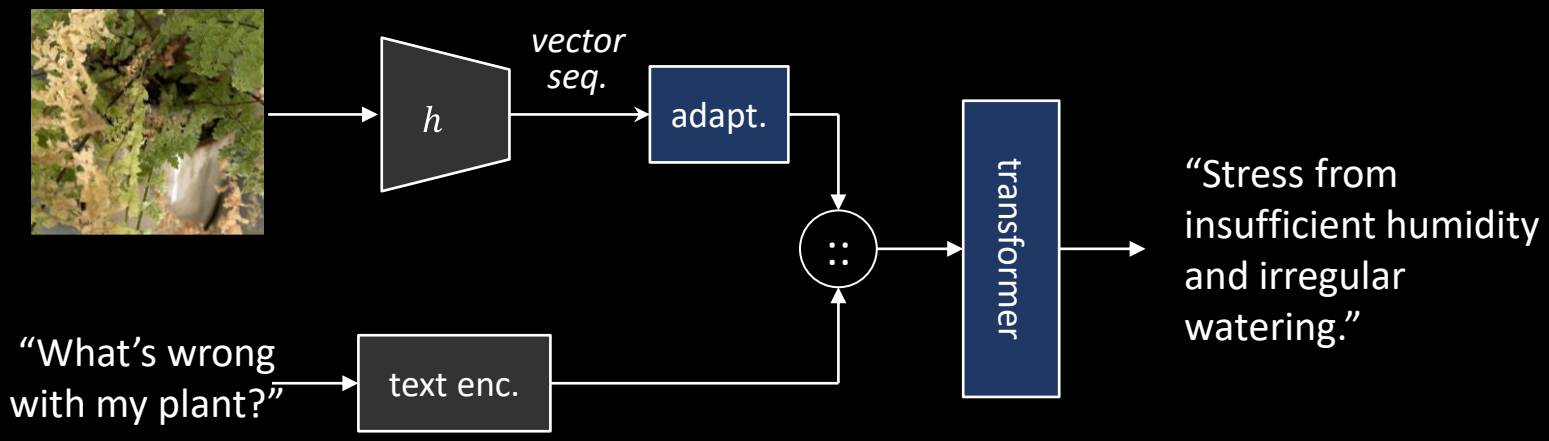
This is SigLIP (Zhai et al. 2023); the dot product makes it cheap. CLIP (Radford et al. 2021) is a variant whose critic answers "which of these pairings is the real one?" rather than "is this pairing real?".

Sample $(x_{\text{im}}, x_{\text{txt}})$ pairs for $q$, and $(x_{\text{im}}, x'_{\text{txt}})$ pairs for the "independent" model.

| **image $x_{\text{im}}$** | **its own alt-text $x_{\text{txt}}$** | **shuffled alt-text $x'_{\text{txt}}$** |
|---|---|---|
|  | Bailey loving the beach today | the view from our hotel in Como |
|  | Biscuit finally gets a taste of his own medicine 😂 | Bailey loving the beach today |
|  | Nike Air Max 90, size 10 | Biscuit finally gets a taste of his own medicine 😂 |
|  | the view from our hotel in Como | Nike Air Max 90, size 10 |

**If we can encode an image as a vector sequence, an LLM will be able to 'see' it.**

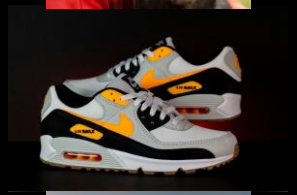
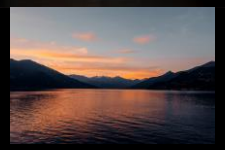
We want to convert the pixels into a useful condensed format, while retaining their relevant information.

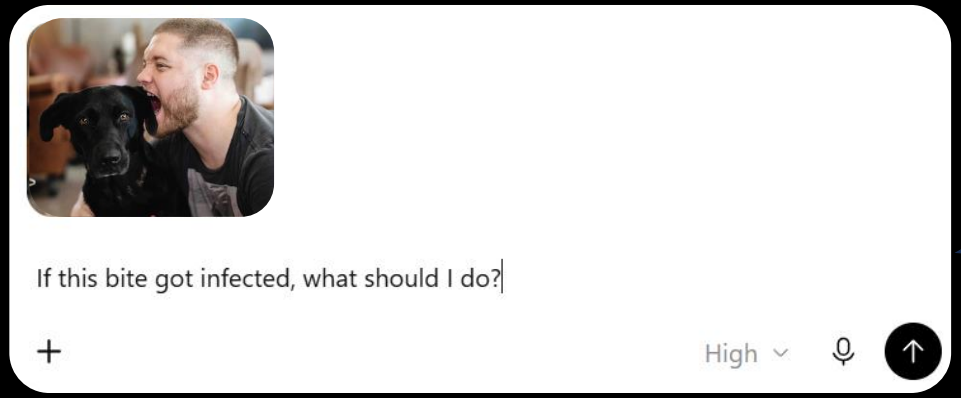



***LLaVA*** *(Liu et al. 2023).*
*Freeze $h$. Train the adapter and the transformer by supervised fine-tuning, on a dataset of conversations about images.*

This is called *contrastive learning.*
It's a general design pattern for extracting structure from unlabelled data

1. Collect data that has the structure you're interested in images with their alt-text
2. Construct a version **such that** the structure is destroyed shuffle the alt-text column
3. Train a critic to tell the difference
4. Wire the critic so that it has a module we can reuse the representation h

| image $x_{\text{im}}$ | its own alt-text $x_{\text{txt}}$ | shuffled alt-text $x'_{\text{txt}}$ |
|---|---|---|
|  | Bailey loving the beach today | the view from our hotel in Como |
|  | Biscuit finally gets a taste of his own medicine 😂 | Bailey loving the beach today |
|  | Nike Air Max 90, size 10 | Biscuit finally gets a taste of his own medicine 😂 |
|  | the view from our hotel in Como | Nike Air Max 90, size 10 |



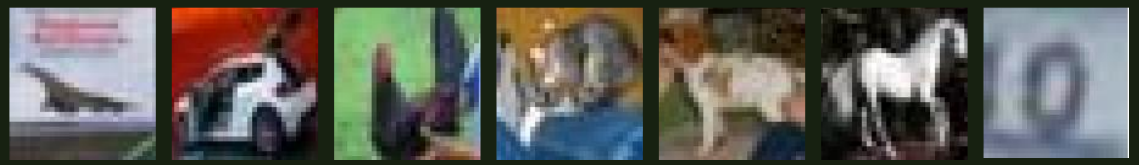

Which reply is the agent more likely to give, and why? (1) “See your GP.” (2) “See your vet.”

Caveat: the critic is blind to structure that it doesn’t need to learn, for the contrastive task you gave it.

Yuksekgonul et al. 2023, *When and why vision-language models behave like bags-of-words*

# Application 3: detecting out-of-distribution inputs

**Task:** we've trained a classifier on one dataset. Can we tell when an input is out-of-distribution (OOD), and decline to classify it?



**Evil Task:** Someone has trained a classifier on one dataset. Can we construct an adversarial input to fool it?

"35 mph" (95.9%)




"85 mph" (25/43)




"cheetah" (99.6%)



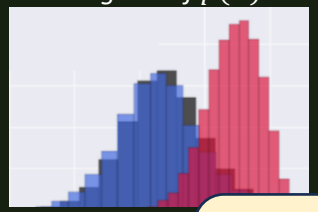
**Answer 1: it's all just likelihood.**

Fit a model to the true distribution, $\hat{p} \approx q$.

If $\hat{p}(x_{\text{new}})$ is very small, flag $x_{\text{new}}$ as OOD.

*This doesn't work – OOD data may well have systematically higher $\hat{p}(\cdot)$*

*histogram of $\hat{p}(X)$*



- *$X \sim$ original dataset (CIFAR10)*
- *$X \sim$ OOD (SVHN)*

*A toy example*: let $X \sim N(0,1)$, and suppose we have fitted $\hat{p} \approx q$ very accurately. Invent a $q_{\text{out}}$ with a larger $\mathbb{E}_{X \sim q_{\text{out}}} \log \hat{p}(X)$.

**Answer 2: look downstream.**

Use the classifier you trained on $q$.

If it's very uncertain on $x_{\text{new}}$ then flag $x_{\text{new}}$ as OOD.

*This doesn't work – classifiers can be confidently wrong.*



"cheetah" (99.6%)

What do we even mean by "OOD"?

**Answer 3: rethink the question.**

OOD means "it comes from some other distribution $q_{\text{out}}$".

Use a critic $\lambda(x) = q_{\text{out}}(x)/\hat{p}(x)$ and flag $x_{\text{new}}$ as OOD if $\lambda(x_{\text{new}})$ is large.

*This begs the question: where does $q_{\text{out}}$ come from?*

*How can we train the critic?*

*We want $\lambda(x) \approx q_{\text{out}}(x)/\hat{p}(x)$, thus $q_{\text{out}}(x) \approx \lambda(x)\hat{p}(x)$.*

*This suggests fitting $q_{\text{out}}$:*

minimize $\mathrm{KL}(q_{\text{out}} \,\|\, \overbrace{\lambda \odot \hat{p}}^{p_{\text{out}}})$ over $\lambda$

such that $p_{\text{out}}$ is a distribution.

# OOD detectors are often presented as algorithms, but they are probability models in disguise.

*"If $q_{\text{out}}$ did not exist, it would be necessary to invent it." – VoltaireBot. This work is from Zhang (2024), "Anomalous Inputs in Deep Learning".*

| ALGORITHMIC MINDSET | | PROBABILITY MODELLING MINDSET |
|---|---|---|
| Compute a score $s(x)$<br>Flag $x$ as OOD if $s(x) > \text{threshold}$ | $\Longleftrightarrow$ | Define the implied OOD distribution<br>$p_{\text{out}}(x) \propto s(x)\,\hat{p}(x)$<br>Flag $x$ as OOD if $\lambda(x) > \text{threshold}$ |

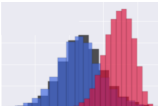
this is called tilting $\hat{p}$, with tilt $s$

Algorithm 1: flag as OOD if $\hat{p}(x) < \kappa$ — $s(x) = 1/\hat{p}(x)$ — $p_{\text{out}}(x) \propto 1$ so most samples look like static 

Algorithm 2: OOD if $H(\text{class}(x)) > \kappa$ — $s(x) = H(\text{class}(x))$ — $p_{\text{out}}(x) \propto \hat{p}(x)\,H(\text{class}(x))$ Zhang gives examples

The entropy H measures the uncertainty in a distribution

## A critic only learns what its training task requires: to distinguish $p_{\text{out}}$ from $\hat{p}$

Algorithm 1 looks out for static – so things that are smooth are seen as "very in-distribution"  

Algorithm 2 looks out for ambiguities – so confident misclassifications are seen as "very in-distribution" 


Zhang (2024) constructed a $p_{\text{out}}(x) \equiv s(x)$ that is blind to textural differences, and therefore detects only macro differences

# Flipping the problem: constructing adversarial inputs to a classifier Zhang 2024

*Consider the following optimization problem:*

given a probabilistic classifier, find the distribution $p_{\text{adv}}$ that minimizes

$$\mathrm{KL}\left(p_{\text{adv}} \parallel p_{x_0}\right) - \beta\, \mathbb{E}_{X \sim p_{\text{adv}}} \underbrace{\Pr_{\text{Class}}(k_0 | X)}_{\substack{\textit{classifier's probability} \\ \textit{of class } k_0 \textit{ for input } X}}$$

We need a distribution $p_{x_0}(\cdot)$ for "items perceived (by humans) as similar to $x_0$". How would you design this distribution?

The solution is a tilted distribution: $p_{\text{adv}}(x) \propto p_{x_0}(x)\, e^{\beta \Pr_{\text{Class}}(k_0|x)}$

Derive this yourself, for a discrete distribution. It's easy!

To sample from $p_{\text{adv}}$ defined this, we can use Langevin dynamics (outside this course)

Langevin dynamics is for models where we know $p$ but don't know how to sample,
whereas GANs are for models where we can sample but don't know $p$

# Gaming a metric

*"When a measure becomes a target, it ceases to be a good measure" — Goodhart's law*

EXAMPLE

You have a generative model $X \sim p$ that you're fond of, but it doesn't pass an evaluation test that the field likes. The field expects your model to match the statistics of a reference distribution $q$:

$$\text{dist}\big(\mathbb{E}_{X\sim p}\, t(X),\ \mathbb{E}_{X\sim q}\, t(X)\big) < \varepsilon.$$ (FID is of this form.)

What's the smallest change you can make to your model, to pass this test?

**Answer 1:** Write $m_r$ for $\mathbb{E}_{X\sim r} t(X)$. Then create a new generator $\tilde{p}$ to solve

$$\text{minimize } \mathrm{KL}(\tilde{p} \parallel p) + \beta \text{dist}(m_{\tilde{p}}, m_q) \text{ over } \tilde{p}$$

**Answer 2: data selection.** Choose a reweighting of the dataset $\lambda$ to solve

$$\text{minimize } \mathrm{KL}(\lambda \odot p \parallel p) + \beta \text{dist}(m_{\tilde{p}}, m_q) \text{ over } \lambda\colon \mathcal{X} \to \mathbb{R}_{>0} \text{ such that } \tilde{p} = \lambda \odot p \text{ is a distribution}$$

Conveniently, $m_{\tilde{p}} \coloneqq \mathbb{E}_{X\sim\tilde{p}} t(X) = \mathbb{E}_{X\sim p} \lambda(X) t(X)$, so it can be easily estimated with samples from $p$.

Kynkäänniemi et al. 2023. They applied Answer 2 to StyleGAN2, a face generator, dramatically improving its FID score, but barely noticeable by humans. What got the weight? ImageNet has no class for faces, only things like "microphone" and "seatbelt", and the FID features are late-layer features; so it chose the faces that happen to classify as those objects, in the proportions that real faces have them. Real game-the-metric sneakiness!

# Lecture 2 optional exercise: training a critic

D.J. Wischik

Let $p$ be the supposed true distribution from which MNIST digits are drawn, and let $q$ be the restriction to even digits. Make yourself a bag of samples from each, using disjoint images, and split each bag into training, validation and holdout sets.

1. Train a classifier $d(x) \approx \Pr(\text{from } q \mid x)$ to tell the bags apart, and let $\lambda(x) = d(x)/(1-d(x))$.
2. Use holdout data to estimate

$$L(q, p, \lambda) := 1 + \mathbb{E}_{X\sim q} \log \lambda(X) - \mathbb{E}_{X\sim p}\lambda(X).$$

   What does this value tell you? Answer with maths, not words.

3. Overtrain your classifier. What is $L$ now? Comment.
4. Write $\hat{q}^{\text{tr}}$ and $\hat{q}^{\text{ho}}$ for the empirical distributions of the training and holdout bags from $q$, and likewise for $p$. Let $\lambda$ be some classifier fitted on the training sets. Discuss the difference between $L^{\text{tr}} := L(\hat{q}^{\text{tr}}, \hat{p}^{\text{tr}}, \lambda)$ and $L^{\text{ho}} := L(\hat{q}^{\text{ho}}, \hat{p}^{\text{ho}}, \lambda)$. If $\lambda$ is overtrained, what happens to $L^{\text{tr}}$ and $L^{\text{ho}}$?
5. What is $\mathrm{KL}(q \,\|\, p)$ for these specific datasets?
6. State a bound on $\mathrm{KL}(p \,\|\, q)$ that uses your classifier from step 1. What is $\mathrm{KL}(p \,\|\, q)$ for these specific datasets?

No submission, not assessed—this is just for you to test your understanding.

If you want to discuss, come and chat with me.

# Most of machine learning can be seen as optimization problems involving probability distributions

Today we looked at constrained optimization
(though sometimes the constraint is sneaky):

$$\text{minimize } \mathrm{KL}(q \parallel r) \text{ over } r, \theta \text{ such that } r = p_\theta$$

When you see a density ratio, it is often linked with this sort of constrained optimization.

$$\lambda(x) = \frac{q(x)}{p(x)}$$

- *GAN: learn $\lambda$, use it to train $p$*
- *Contrastive: fix $p$, learn $\lambda$ for the sake of a module inside it*
- *OOD detection: based on $\lambda = p_{out}/\hat{p}$*
- *Adversarial generation: $p_{\text{adv}} = p_{x_0}\lambda$*

We also saw some klunky optimization with a policy knob:

$$\text{minimize} \quad \mathrm{KL}(q \parallel p_\theta) + \beta \,\text{penalty}(p_\theta) \quad \text{over } \theta$$

Next week: LLM training as KL-based optimization,
alignment via a density ratio.

# Most of machine learning ~~can~~ *should* be seen as optimization problems involving probability distributions (IMHO)

- We can reason about what we want to achieve separately from how we achieve it (it's declarative).
  *GAN failure modes*
- The goal itself suggests algorithms, via gradient descent.
  *GAN minimax; adversarial image generation*
- It's easier to extend safely, without side-effects (it's declarative).
  *adversarial images, incorporating human-perceived similarity*
- It's more interpretable and debuggable.
  *contrastive learning's blind spots; OOD blind spots*
- It helps make sense of the field: "it's all just KL"!