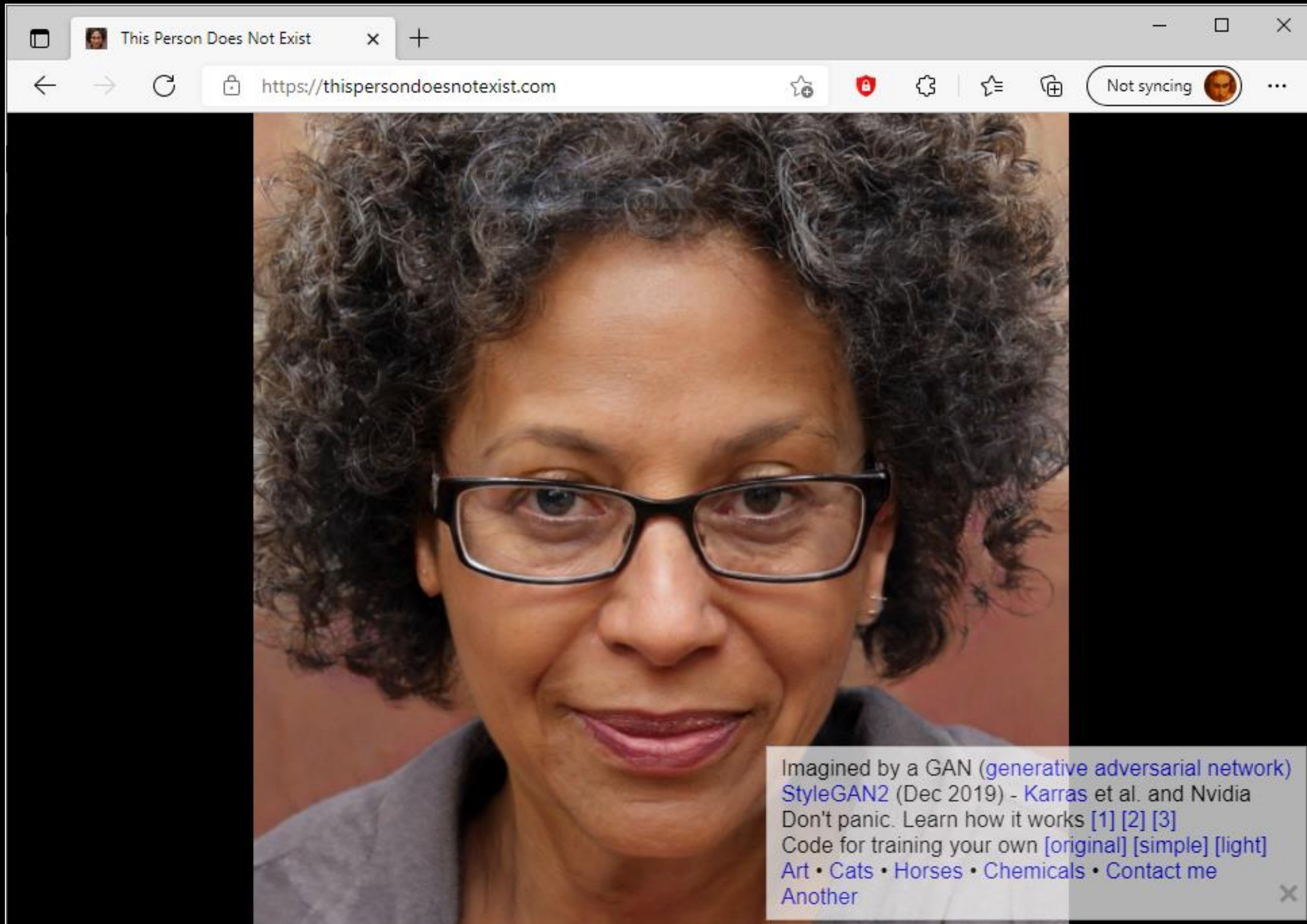
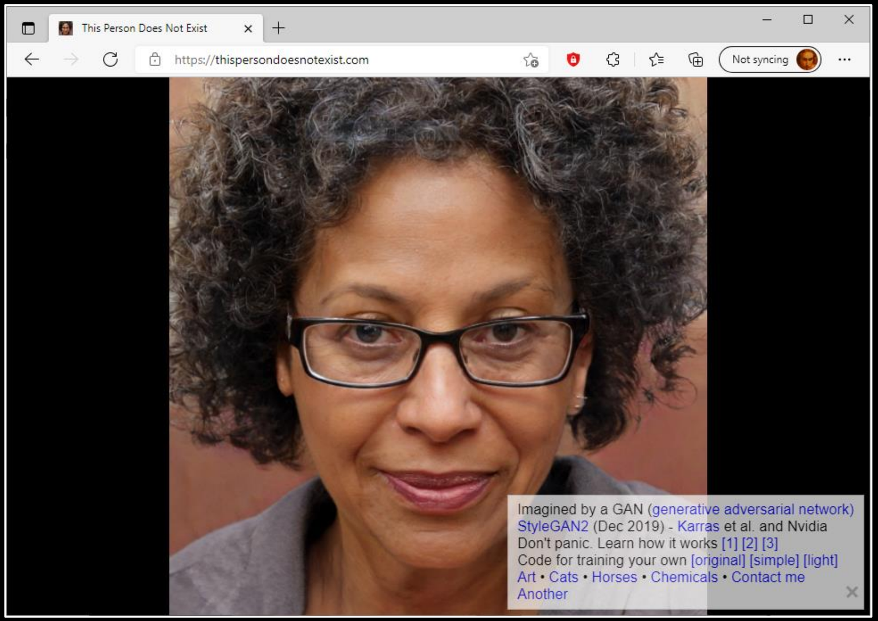
GANs and other optimizations


This Person Does Not Exist
https://thispersondoesnotexist.com
Not syncing
Imagined by a GAN (generative adversarial network)
StyleGAN2 (Dec 2019) - Karras et al. and Nvidia
Don't panic. Learn how it works [1] [2] [3]
Code for training your own [original] [simple] [light]
Art • Cats • Horses • Chemicals • Contact me
Another

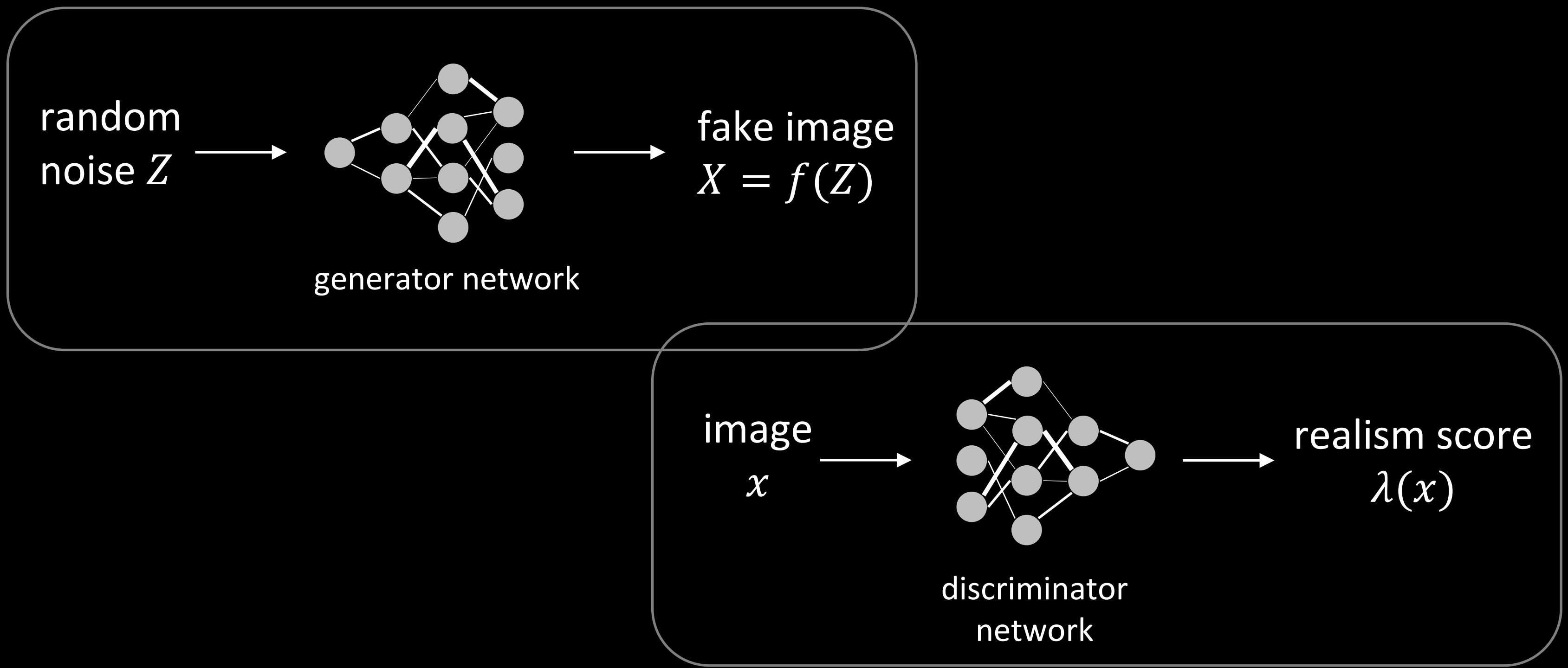
Given a dataset of images, how can we train a neural network to be able to generate realistic fakes?

Flickr-Faces-HQ Dataset (FFHQ)
https://github.com/NVlabs/ffhq-dataset



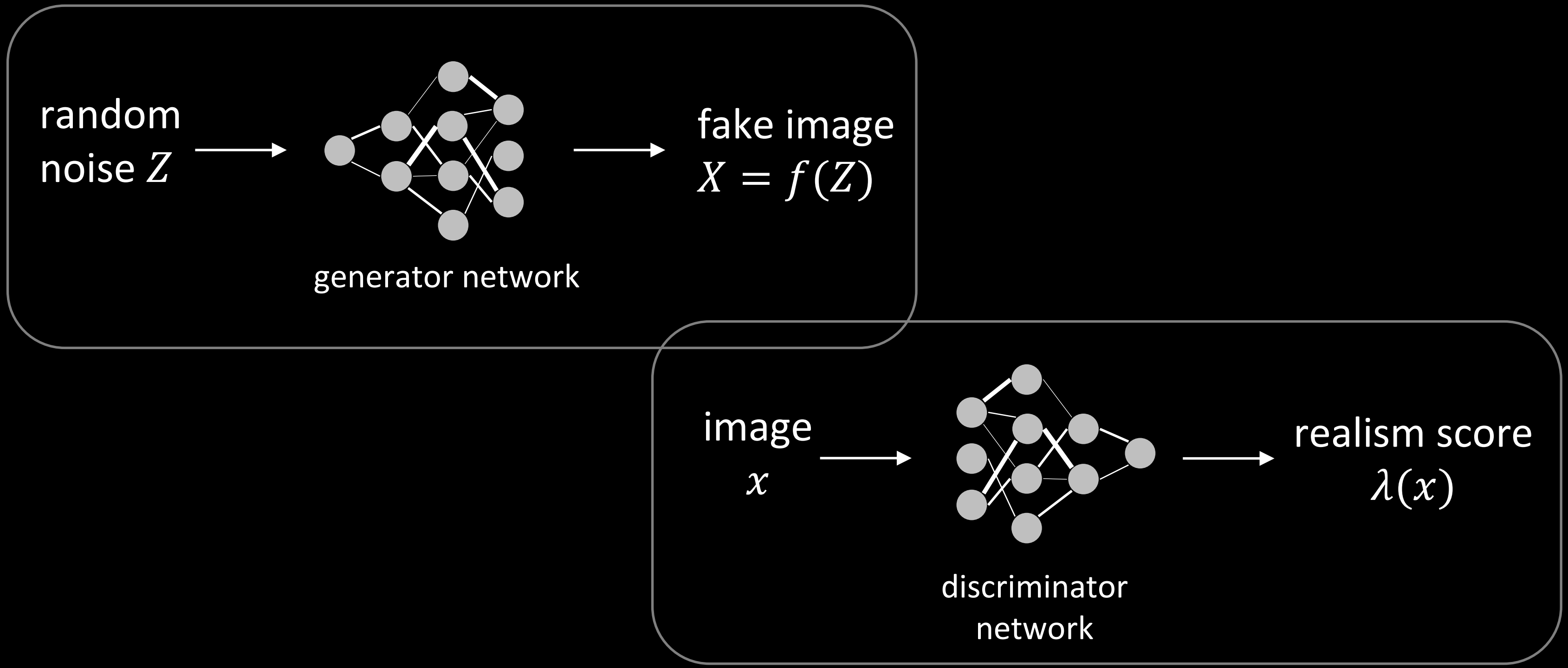
StyleGan2 by Nvidia labs

Given a dataset of images, how can we train a neural network to be able to generate realistic fakes?



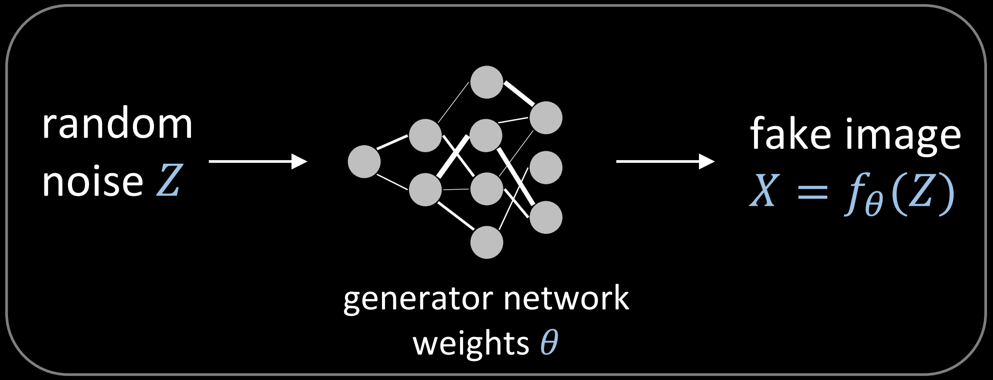
- train the discriminator to distinguish genuine images from fake
- train the generator to fool the discriminator

Given a dataset of images, how can we train a neural network to be able to generate realistic fakes?



To understand what's going on, it's useful to restate the problem using in the language of random variables.

Given a dataset of images, how can we train a neural network to be able to generate realistic fakes?




```
def gen_face(θ):
    Z = random.random()
    X = f_θ(Z)
    return X
```

To understand what's going on, it's useful to restate the problem using in the language of random variables.

WHAT IS A RANDOM VARIABLE?

```
def gen_geom(p):
    Z = random.random()
    λ = - math.log(1-p)
    X = math.ceil(-math.log(Z)/λ)
    return X
```

1 Figure out the probability mass function

$$\mathbb{P}(X=x) = (1-p)^{x-1} p$$

for $x \in \{1, 2, 3, \cdots\}$

HOW DO WE LEARN FROM DATA?

Suppose we're given a dataset $\{x_1, x_2, \ldots, x_n\}$ and we want to tune our random variable generator by choosing p to match the dataset as closely as possible. How can we do this?

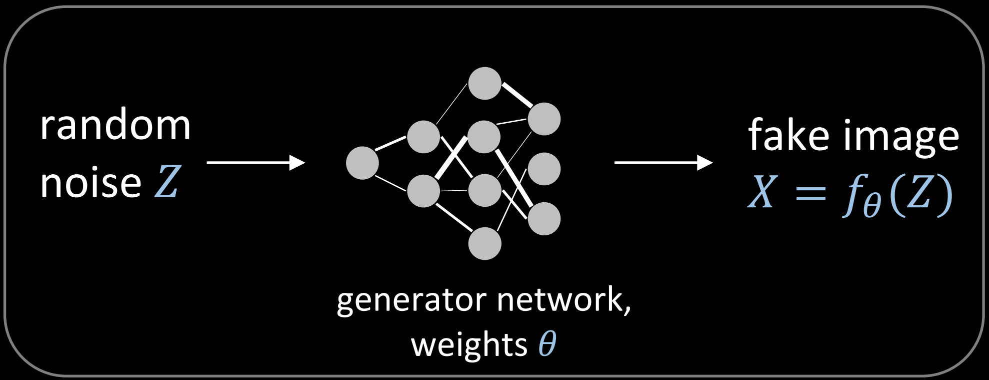
2 what's the probability that our rng generated $\{x_1, \cdots, x_n\}$?

$$\mathbb{P}(\text{gen. this dataset}) = \mathbb{P}(X=x_1) \times \mathbb{P}(X=x_2) \times \cdots \times \mathbb{P}(X=x_n)$$

$$= \left(\frac{p}{1-p}\right)^n (1-p)^{\sum x_i}$$

3 Choose p to make this probability as large as possible. $\hat{p} = \frac{n}{\sum x_i}$

Given a dataset of images, how can we train a neural network to be able to generate realistic fakes?




```
def gen_face(θ):
    Z = random.random()
    X = f_θ(Z)
    return X
```

We have no hope of calculating a formula for this. Nevertheless...

X is a random variable. Let its prob. mass func. be $\mathbb{P}(X = x) = p_\theta(x)$.

We're given a dataset $\{x_1, x_2, \dots, x_n\}$ of faces, and we our generator's output to match this dataset as closely as possible. We should find the θ that maximizes

$$p_\theta(x_1) \times p_\theta(x_2) \times \cdots \times p_\theta(x_n)$$

Or, equivalently, let's pick θ to maximize

$$\mathcal{V}(\theta) := \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(x_i)$$

TRAINING GOAL

pick θ to maximize $\mathcal{V}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x_i) = \sum_{x \in \mathcal{X}} q_x \log p_\theta(x)$

$q_x = \text{\#times } x \text{ occurs}/n$

$\mathcal{X} = \text{set of all possible images}$

EQUIVALENT TRAINING GOAL (YOU'LL NEVER BELIEVE THIS ONE WEIRD TRICK)

maximize $\sum_x q_x \log r_x$

over $r \in \mathbb{R}^{\mathcal{X}}, \theta$

such that $r_x = p_\theta(x)$ for all x

LAGRANGIAN

$$\mathcal{L}(\theta, r; \lambda) = \sum_{x \in \mathcal{X}} q_x \log r_x - \sum_{x \in \mathcal{X}} \lambda_x \big(r_x - p_\theta(x)\big)$$

LAGRANGIAN WEAK DUALITY

For any θ and any λ,

with some simple optimization over r

$$\mathcal{V}(\theta) = \mathcal{L}(\theta, p_\theta; \lambda) \le \max_{r \in \mathbb{R}^{\mathcal{X}}} \mathcal{L}(\theta, r; \lambda) = \begin{cases} \text{const} - \underbrace{\sum_x q_x \log \lambda_x}_{\lambda_x \text{ should be larger where } q_x \text{ is larger}} + \underbrace{\sum_x p_\theta(x) \lambda_x}_{\lambda_x \text{ should be smaller where } p_\theta(x) \text{ is larger}}, & \underbrace{\text{if } \lambda_x > 0 \text{ for all } x}_{\lambda_x \text{ should be } > 0 \text{ for all } x} \\ \infty, & \text{otherwise} \end{cases}$$

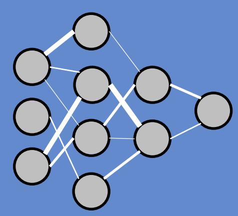
What values of λ would make this upper bound on $\mathcal{V}(\theta)$ as tight as possible?

TRAINING GOAL

pick θ to maximize $\mathcal{V}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x_i) = \sum_{x\in\mathcal{X}} q_x \log p_\theta(x)$

$q_x = {}^{\#\text{times } x \text{ occurs}}/_{n}$

$\mathcal{X} = \text{set of all possible images}$

image x →  → realism score $\lambda(x)$

discriminator network

We aimed to train the discriminator so that $\lambda(x)$ larger if x is genuine, smaller if it's faked.

LAGRANGIAN WEAK DUALITY

For any θ and any λ,

with some simple optimization over r

$$\mathcal{V}(\theta) = \mathcal{L}(\theta, p_\theta; \lambda) \le \max_r \mathcal{L}(\theta, r; \lambda) = \begin{cases} \text{const} - \sum_x q_x \log \lambda_x + \sum_x p_\theta(x)\lambda_x, & \text{if } \lambda_x > 0 \text{ for all } x \\ \infty, & \text{otherwise} \end{cases}$$

What values of λ would make this upper bound on V(θ) as tight as possible?

λ_x should be larger where q_x is larger

λ_x should be smaller where $p_\theta(x)$ is larger

λ_x should be > 0 for all x

FURTHER DETAILS

The score

$$\mathcal{V}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(x_i)$$

measures how well our generator performs. We have shown that for any θ and any $\lambda > 0$

$$\mathcal{V}(\theta) \le \text{const} - \sum_x q_x \log \lambda_x + \sum_x p_\theta(x)\lambda_x\,.$$

Let's propose a discriminator neural network $x \mapsto d_\phi(x) > 0$ for computing $\lambda_x = d_\phi(x)$, and try to find good neural network weights ϕ. Since our Lagrangian bound is true for any θ and any $\lambda > 0$, it follows that

$$\max_\theta \mathcal{V}(\theta) \le \text{const} + \max_\theta \min_\phi \left\{-\sum_x q_x \log d_\phi(x) + \sum_x p_\theta(x) d_\phi(x)\right\}$$

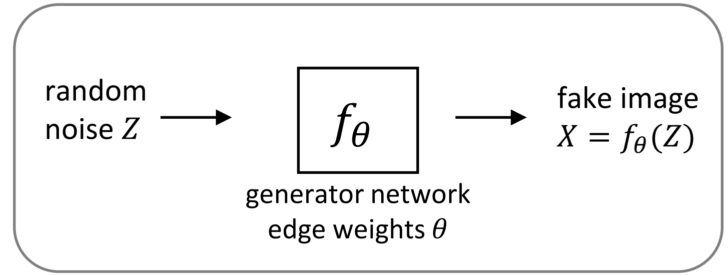
We can approximate the $\{\cdot\}$ term by

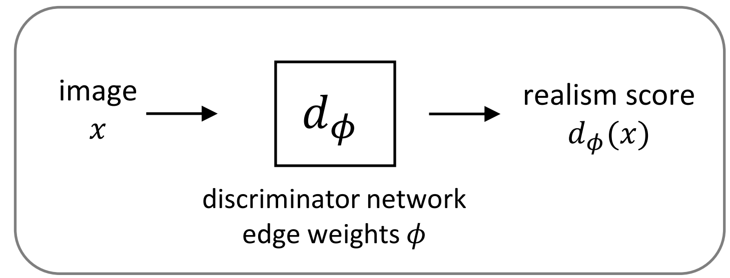
$$L(\theta,\phi) \coloneqq -\frac{1}{n}\sum_{i=1}^{n} \log d_\phi(x_i) + \frac{1}{n}\sum_{i=1}^{n} d_\phi(f_\theta(z_i))$$

where $\{x_1, \dots, x_n\}$ are the images in the training dataset and $\{z_1, \dots, z_n\}$ are randomly generated noise terms. Training consists in using gradient descent to compute

$$\max_\theta \min_\phi L(\theta,\phi)\,.$$

This doesn't actually compute $\max_\theta \mathcal{V}(\theta)$, it only computes an upper bound, but hopefully the upper bound is reasonably tight and we end up learning a generator with a good score.

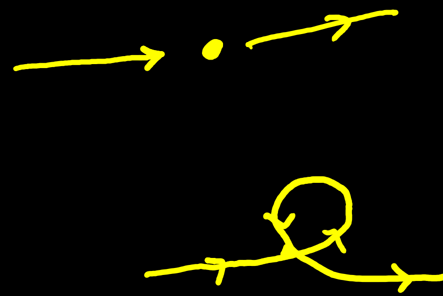




Other optimizations

Shortest path problem

Given a directed graph where the weight of edge $v \to w$ is c_{vw}, find the minimum-weight path from a start vertex s to a destination vertex t



minimize $\quad \sum_{v \to w} x_{vw}\, c_{vw}$

over $\quad x \in \{0,1,2,\ldots\}^E$

not $x \in \{0,1\}^E$, (since paths might re-use edges)

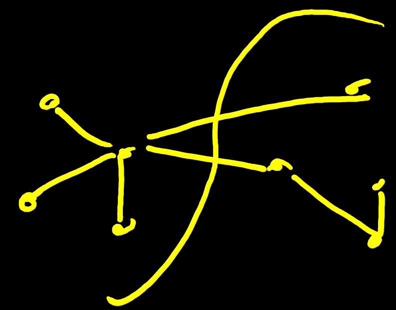
such that $\quad$ net flow out of s is 1

and $\quad$ flow is conserved at all vertices in $V \setminus \{s, t\}$

Minimum spanning tree problem

Given an undirected graph where the weight of edge $v \leftrightarrow w$ is c_{vw}, find a spanning tree of minimum weight

A tree is an acyclic connected graph



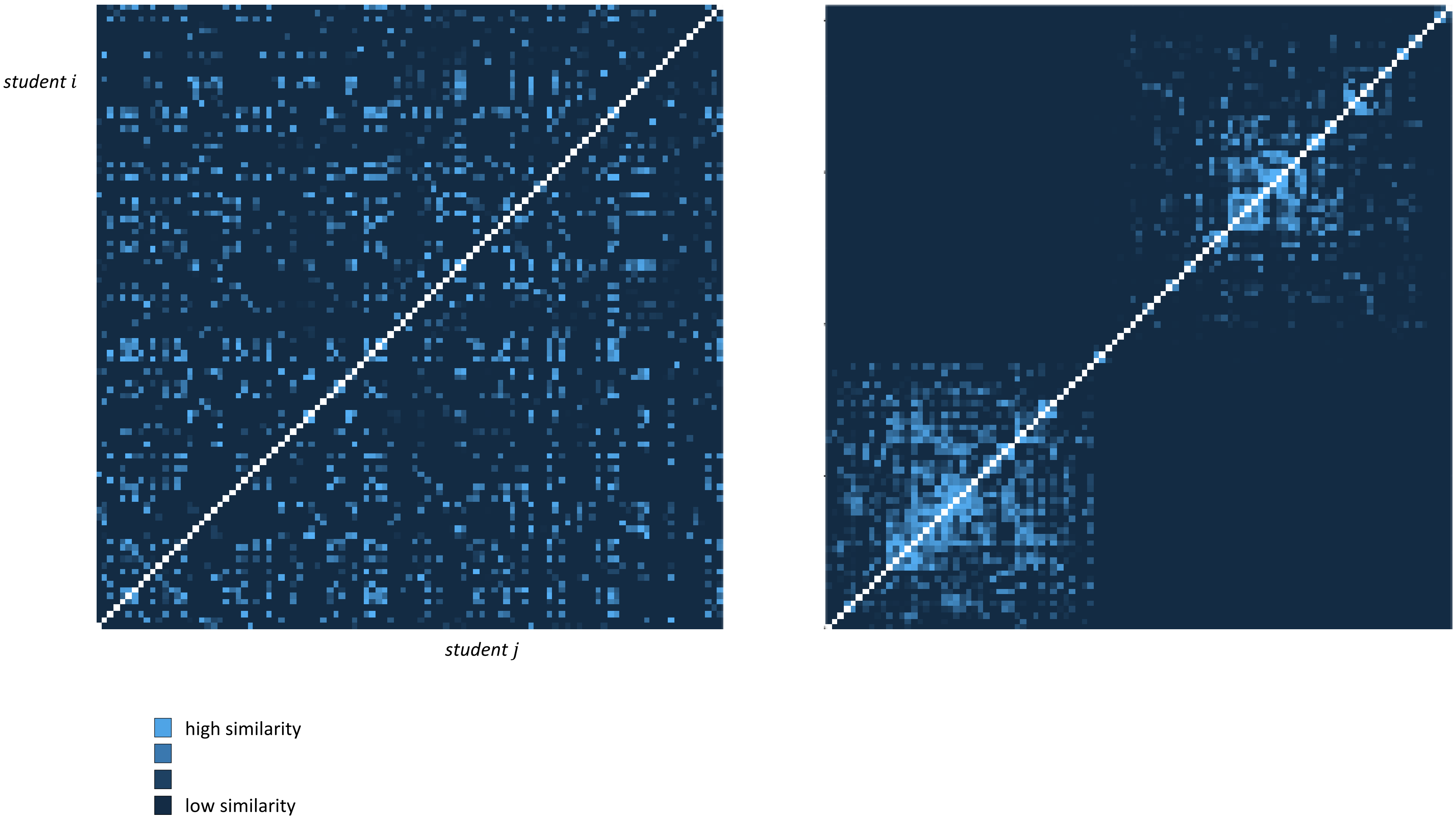
$$
\begin{array}{ll}
\text{minimize} & \sum_{v \leftrightarrow w} x_{vw}\, c_{vw} \\
\text{over} & x \in \{0,1\}^E \\
\text{such that} & \sum_{v \leftrightarrow w} x_{vw} = |V| - 1 \\
\text{and} & \sum_{\substack{v \leftrightarrow w: \\ v \in S, w \in \bar{S}}} x_{vw} \geq 1 \text{ for all sets } S \subset V, S \neq V, S \neq \emptyset
\end{array}
$$

implies acyclic

connected

Challenge: find an ordering for all students so that students with similar Tick 1 code are close to each other

Similarity matrix of Tick 1 code



	score on training data (2021 tick1)	score on holdout data (2022 tick1)
Tunan Shi (Sidney Sussex)	77.4	62.7
Kuba Bachurski (Trinity)	76.9	62.4
Mark Li (Corpus Christi)	76.6	61.8
Andy Zhou (Queens’)	73.0	60.2
Cheuk Kit Lee (Downing)	77.2	timeout
Jiayou Song (Robinson)	75.0	timeout

Software 1.0 is code we write. Software 2.0 is code written by the optimization based on an evaluation criterion (such as "classify this training data correctly"). It is likely that any setting where the program is not obvious but one can repeatedly evaluate the performance of it (e.g. — did you classify some images correctly? do you win games of Go?) will be subject to this transition, because the optimization can find much better code than what a human can write.

- Can I express my task as an optimization problem?
- … that can be solved with off-the-shelf optimizers?
- If I can’t, is there an adjacent problem that’s more amenable?