GANs and other optimizations
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?

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

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

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?

1. Figure out the probability mass function:
   \[
   P(X = x) = (1-p)^{x-1} p 
   \]
   for \( x \in \{1, 2, 3, \ldots\} \)

2. What’s the probability that our rng generated \( \{x_1, \ldots, x_n\} \)?

   \[
   P(\text{gen. this dataset}) = P(X = x_1) \times P(X = x_2) \times \cdots \times P(X = x_n)
   \]
   \[
   = \left(\frac{p}{1-p}\right)^{x_1} \times (1-p)^{x_2} \times \cdots \times (1-p)^{x_n}
   \]

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?

A random noise $Z$ is fed into a generator network with weights $\theta$, which outputs a fake image $X = f_\theta(Z)$.

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

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

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

Or, equivalently, let's pick $\theta$ to maximize

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

```python
def gen_face(θ):
    Z = random.random()
    X = f_θ(Z)
    return X
```
**TRAINING GOAL**

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

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

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

**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**

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

**LAGRANGIAN WEAK DUALITY**

For any $\theta$ and any $\lambda$,

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

What values of $\lambda$ would make this upper bound on $V(\theta)$ as tight as possible?
TRAINING GOAL

pick $\theta$ to maximize

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

$\lambda(x)$

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 $\theta$ and any $\lambda$,

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

What values of $\lambda$ would make this upper bound on $V(\theta)$ as tight as possible?
FURTHER DETAILS

The score

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

measures how well our generator performs. We have shown that for any \( \theta \) and any \( \lambda > 0 \)

\[ V(\theta) \leq \text{const} - \sum_x q_x \log \lambda_x + \sum_x p_{\theta}(x) \lambda_x. \]

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 \( \phi \). Since our Lagrangian bound is true for any \( \theta \) and any \( \lambda > 0 \), it follows that

\[ \max_{\theta} V(\theta) \leq \text{const} + \max_{\theta} \min_{\phi} \left\{ - \sum_x q_x \log d_\phi(x) + \sum_x p_{\theta}(x) d_\phi(x) \right\} \]

We can approximate the \( \{ \} \) term by

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

where \( \{x_1, ..., x_n\} \) are the images in the training dataset and \( \{z_1, ..., z_n\} \) 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 \( \max_{\theta} V(\theta) \), 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 $t$

\[
\begin{align*}
\text{minimize} & \quad \sum_{v \to w} x_{vw} c_{vw} \\
\text{over} & \quad x \in \{0,1,2,\ldots\}^E, \quad \text{not } x \in \{0,1\}^E, \\
\text{such that} & \quad \text{net flow out of } s \text{ is 1} \\
\text{and} & \quad \text{flow is conserved at all vertices in } V \setminus \{s,t\}
\end{align*}
\]
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

\[
\text{minimize} \quad \sum_{v \leftrightarrow w} x_{vw} c_{vw} \\
\text{over} \quad x \in \{0,1\}^E \\
\text{such that} \quad \sum_{v \leftrightarrow w} x_{vw} = |V| - 1 \\
\text{and} \quad \sum_{v \in S, w \in \overrightarrow{S}} x_{vw} \geq 1 \text{ for all sets } S \subset V, S \neq \emptyset, S \neq V}

A tree is an \underline{acyclic connected graph}

\[
\begin{array}{c}
\text{A tree is an} \\
\text{acyclic connected graph}
\end{array}
\]
Challenge: find an ordering for all students so that students with similar Tick 1 code are close to each other.
<table>
<thead>
<tr>
<th>Name</th>
<th>Score on training data (2021 tick1)</th>
<th>Score on holdout data (2022 tick1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tunan Shi (Sidney Sussex)</td>
<td>77.4</td>
<td>62.7</td>
</tr>
<tr>
<td>Kuba Bachurski (Trinity)</td>
<td>76.9</td>
<td>62.4</td>
</tr>
<tr>
<td>Mark Li (Corpus Christi)</td>
<td>76.6</td>
<td>61.8</td>
</tr>
<tr>
<td>Andy Zhou (Queens’)</td>
<td>73.0</td>
<td>60.2</td>
</tr>
<tr>
<td>Cheuk Kit Lee (Downing)</td>
<td>77.2</td>
<td>timeout</td>
</tr>
<tr>
<td>Jiayou Song (Robinson)</td>
<td>75.0</td>
<td>timeout</td>
</tr>
</tbody>
</table>
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?