Generative Adversarial Networks - Overview

- How do GANs work?

```
random noise values
↓
G
↓ generator network
→ D
→ discriminator
→ 'Real Image?'
\[\checkmark\]
\[\square\]
guides the generator
```

* "D" is shown real images half of the time, and fake "gen" images the other half. "D" is trained to output close to 1 prob. for real images.

Meanwhile, "G" tries to generate images that "D" will classify as real.

So, over time, "G" learns to produce more real images in order to fool "D".
G and D are at a competition.

* Choosing a good architecture

* lrelu = leaky relu

```
D
matmul + lrelu
matmul + lrelu
matmul + tahn
matmul + lrelu
```

```
G
matmul + sigmoid
matmul + lrelu
```

« a simple GAN »

D and G should have at least 1 hidden layer

- In GANs there are two optimizations:
  - optim. Adam (D. parameters (), lr)
  - optim. Adam (G. parameters (), lr)

\[ \boxed{\text{D} \rightarrow \text{logits} \rightarrow \text{sigmoid} \rightarrow \text{prob} \frac{0}{1}} \] (binary classification task)

\[ d - \text{loss} = \text{nn} \cdot \text{BCEWithLogitsLoss}(\text{logits}, \text{label} \ast 0.9) \]

- numerically stable cross-entropy
- it help to generalize better

\[ g - \text{loss} = \text{nn} \cdot \text{BCEWithLogitsLoss}(\text{logits, flipped_labels}) \]
Alternative GAN Architecture

D

matmul + sigmoid

reshape

Conv + ReLU

G

Conv + tanh

Conv + ReLU

reshape

random factor \( Z \)