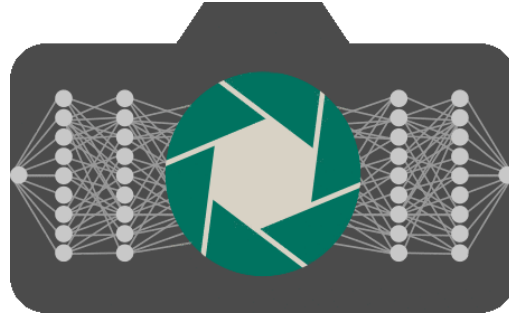


# CSE 559A: Computer Vision



Fall 2018: T-R: 11:30-1pm @ Lopata 101

Instructor: Ayan Chakrabarti (ayan@wustl.edu).

Course Staff: Zhihao Xia, Charlie Wu, Han Liu

<http://www.cse.wustl.edu/~ayan/courses/cse559a/>

December 4, 2018

# GENERAL

---

- PSET 3 grades posted, PSET 4 to be posted shortly.
- PSET 5 due today.
- No office hours on Friday.

- Project reports due December 9.
- Submit through git

```
git clone cse559@euclid.seas.wustl.edu:wustl.key/project
```

- Include only your report PDF. Do not submit your code or other files.
  - But hang on to your code in case we ask for it later.

# GENERATIVE ADVERSARIAL NETWORKS

---

- So far, we have looked at networks that given an input  $x$ , produce an output  $y$ .
- All  $(x, y)$  pairs come from some joint distribution  $p(x, y)$ .
- A function that maps  $x \rightarrow y$  is then reasoning with the distribution  $p(y|x)$
- And producing a single guess  $\hat{y}$  which minimizes  $\mathbb{E}_{p(y|x)}L(y, \hat{y})$ .
- But if  $p(y|x)$  is not deterministic, this expected loss won't go to zero: Bayes Error
- What if I didn't want my network to produce a "best" guess, but tell me about this distribution ?

# GENERATIVE ADVERSARIAL NETWORKS

---

- One option: choose a parametric form for  $p(y|x) = f(y; \theta)$ .
- Have a network  $g$  that predicts  $\theta = g(x)$  for a specific  $x$ .
- The other option is to train a sampler. A network that given an input  $x$ , produces "samples" from  $p(y|x)$ .
- How do you produce samples, or how do you produce multiple outputs for the same input ?
- You give your network access to a random generator: a noise source.

# GENERATIVE ADVERSARIAL NETWORKS

---

- Let's ignore conditional distributions. Consider the task of generating samples from  $p_x(x)$ .
- You don't know  $p(x)$ , but you have training examples that are samples from  $p_x(x)$ .
- You want to learn a "generator" network  $G(z; \theta)$  which
  - Takes in random inputs  $z$  from a known distribution  $p_z(z)$
  - And produces outputs  $x$  from  $p(x)$
  - Has learnable parameters  $\theta$
- You want to select  $\theta$  such that the distribution of  $\{G(z; \theta) : z \sim p_z(z)\}$  to match  $p_x(x)$ .
- But you don't have the data distribution, only samples from it.

# GENERATIVE ADVERSARIAL NETWORKS

---

- Set this up as a min-max objective with a second network, a "discriminator"  $D(x, \phi)$
- The discriminator is a binary classifier. Tries to determine if
  - The input  $x$  is "real", i.e., it came from the training set.
  - Or "fake", i.e., it was the output of  $G$
- Train both networks simultaneously, against the following loss:

$$L(\theta, \phi) = -\mathbb{E}_{z \sim p_z} \log(1 - D(G(z; \theta); \phi)) - \mathbb{E}_{x \sim p_x} \log D(x; \phi)$$

- This is the cross-entropy loss on the discriminator saying outputs of  $G$  should be labeled 0.
- What about examples that should be labeled 1 ?

# GENERATIVE ADVERSARIAL NETWORKS

---

$$L(\theta, \phi) = -\mathbb{E}_{z \sim p_z} \log(1 - D(G(z; \theta); \phi)) - \mathbb{E}_{x \sim p_x} \log D(x; \phi)$$

- Expectation  $\mathbb{E}_{x \sim p_x}$  is just average over training set.
- Expectation  $\mathbb{E}_{z \sim p_z}$  is based on sampling  $z$  from known  $p_z$  at each iteration.

$$\theta = \arg \max_{\theta} \min_{\phi} L(\theta, \phi)$$

- You are optimizing the discriminator to succeed in telling real and fake samples. To minimize the loss.
- And training the generator to fool the discriminator, i.e., *maximize the same loss*
- How do you solve this optimization problem ?
- Turns out, it is reasonable to use back-prop and gradient descent.
  - Just compute gradients with respect to the loss for both the discriminator and generator.
  - Subtract them from the discriminator, add them to the generator.

# GENERATIVE ADVERSARIAL NETWORKS

---

$$G = \arg \max_G \min_D -\mathbb{E}_{z \sim p_z} \log(1 - D(G(z))) - \mathbb{E}_{x \sim p_x} \log D(x)$$

## Theoretical Analysis

- Let's say your discriminator and generator had infinite capacity and you had infinite training data.
- For a given input  $x$ , what should the optimal output of your discriminator  $D(x)$  be?
  - Say you know  $p_x(x)$ .
  - You also know  $p_g(x)$  and  $G$ , and therefore  $p_g(x)$ : probability of  $x$  being an output from the generator.

$$q = D(x) = \arg \min_q -p_g(x) \log(1 - q) - p_x(x) \log q$$

- What  $q$  minimizes this, for  $q \in [0, 1]$ ?

$$q = \frac{p_x(x)}{p_g(x) + p_x(x)}$$

- Let's replace  $D$  with this optimal value in the loss function, and figure out what  $G$  should do.



# GENERATIVE ADVERSARIAL NETWORKS

---

$$G = \arg \max_G \min_D -\mathbb{E}_{z \sim p_z} \log(1 - D(G(z))) - \mathbb{E}_{x \sim p_x} \log D(x)$$

$$G = \arg \min_G \mathbb{E}_{z \sim p_z} \log \frac{p_g(G(z))}{p_x(G(z)) + p_g(G(z))} + \mathbb{E}_{x \sim p_x} \log \frac{p_x(x)}{p_x(x) + p_g(x)}$$

- Remember that  $p_g$  also depends on  $G$ . In fact, you can replace this as an optimization on  $p_g$ .

$$p_g = \arg \min \int_x \left[ p_g(x) \log \frac{p_g(x)}{p_x(x) + p_g(x)} + p_x(x) \log \frac{p_x(x)}{p_x(x) + p_g(x)} \right] dx$$

- You can relate this to KL-divergences:

$$KL \left( p_g \parallel \frac{p_g + p_x}{2} \right) + KL \left( p_x \parallel \frac{p_g + p_x}{2} \right)$$

- Called the "Jensen-Shannon" Divergence
- Minimized when  $p_g$  matches  $p_x$ .

# GENERATIVE ADVERSARIAL NETWORKS

---

$$L(\theta, \phi) = -\mathbb{E}_{z \sim p_z} \log(1 - D(G(z; \theta); \phi)) - \mathbb{E}_{x \sim p_x} \log D(x; \phi)$$

## Practical Concerns

- So the procedure is, set up your generator and discriminator networks. Define the loss.
- At each iteration, pick a batch of  $z$  values from a known distribution (typically a vector of uniformly or Gaussian distributed values)
- And a batch of training samples
- Compute gradients for the discriminator and update.
- Compute gradients for the generator, by back-propagating **through** the discriminator, and update.
- A common issue is that the discriminator has a much "easier" task
  - In the initial iterations, your generator will be producing junk.
  - Very easy for the discriminator to identify fake samples with high confidence.
  - At that point,  $\log 1 - D(G(z))$  will saturate.
  - No gradients to generator.

# GENERATIVE ADVERSARIAL NETWORKS

---

$$L(\theta, \phi) = -\mathbb{E}_{z \sim p_z} \log(1 - D(G(z; \theta); \phi)) - \mathbb{E}_{x \sim p_x} \log D(x; \phi)$$

## Practical Concerns

- One common approach: minimize  $G$  with respect to a different loss
  - Instead of  $\max -\log(1 - D(G(z)))$
  - Do  $\min -\log D(G(z))$
- View as minimizing cross-entropy wrt wrong label, rather than maximizing wrt true label.
- Other approaches:
  - Reduce capacity of discriminator
  - Make fewer updates to discriminator, or have lower learning rate
  - Provide additional losses to generator to help it train: e.g., separate network that predicts intermediate features of the discriminator.
  - Other losses: See Wasserstein GANs.
- Also need to be careful how you use Batch Normalization. (Don't let the discriminator use batch statistics to tell real and fake apart!)

# GENERATIVE ADVERSARIAL NETWORKS

---

- Conditional GANs: now want to sample from  $p(x|s)$  for a given  $s$
- Same adversarial setting but  $s$  is given as an input to both generator and discriminator
  - $G(z, s)$  and  $D(s)$
- Sometimes a noise source is simply replaced by dropout
- Or no noise at all: called an "Adversarial Loss". The generator is producing a deterministic output, but being trained with a distribution matching loss rather than  $L_1$  or  $L_2$ .
  - Can be useful when the true  $p(x|s)$  is multi-modal.
  - Regular networks would average the modes, adversarial loss promotes picking one of the modes.