GDPP

Learning Diverse Generations using Determinantal Point Process

Mohamed Elfeki, Camille Couprie, Morgane Rivière and Mohamed Elhoseiny

* https://github.com/M-Elfeki/GDPP
What’s wrong with Generative models?
What’s wrong with Generative models?

GAN

Real Sample
Fake Sample
What’s wrong with Generative models?

GAN

GDPP-GAN

- Real Sample
- Fake Sample
Determinantal Point Process (DPP)

\[ \mathcal{P}(S \subseteq Y) \propto \det(L_S) \]

\[ \mathcal{P}(S \subseteq Y) \propto \det(\phi(S)^\top \phi(S)) \]

\(\phi\) is feature representation of subset \(S\) sampled from ground set \(Y\)
Determinantal Point Process (DPP)

\[ \mathcal{P}(S \subseteq Y) \propto \det(L_S) \]

\[ \mathcal{P}(S \subseteq Y) \propto \det(\phi(S)^\top \phi(S)) \]

\( \phi \) is feature representation of subset \( S \) sampled from ground set \( Y \)

\( L_S \): DPP kernel, models the diversity of a mini-batch \( S \)
What is GDPP?

Fake Data

Real Data

Generation Loss
What GDPP?

Fake Data

Real Data

Generation Loss

Diversity Loss: Eigen Values/Vectors

$\lambda^1_{fake}$

$\lambda^2_{fake}$

$\lambda^1_{real}$

$\lambda^2_{real}$
What GDPP?

\[ \mathcal{L}_{g_{DPP}} = \mathcal{L}_m + \mathcal{L}_s = \sum_i \| \lambda_{\text{real}}^i - \lambda_{\text{fake}}^i \|_2 - \sum_i \hat{\lambda}_{\text{real}}^i \cos(v_{\text{real}}^i, v_{\text{fake}}^i) \]
How GDPP?
How GDPP?

Fake Non-Diverse Batch

Real Diverse Batch
How GDPP?

Z_B

G

Fake/Real \( \Phi(\cdot) \)

D/E

S_B

Fake Non-Diverse Batch

OR

OR

D_B

Real Diverse Batch
How GDPP?

$S_B = G(z_B) \quad D_B \sim \text{Real}$

$L_B = \phi(B)^T \phi(B)$

Fake/Real $\phi(.)$

D/E

$Z_B$

$G$

Fake Non-Diverse Batch

Real Diverse Batch

OR

Diversity Loss
Does it work? (Synthetic)
Does it work? (Synthetic)
Does it work? (Real)
What else?

2D Grid

High quality samples

Batch Size

Data Efficient

GDPP-GAN
Unrolled-GAN
VEE-GAN
WGAN-GP
What else?

Data Efficient

Time Efficient
What else?

2D Grid

- Data Efficient
  - GDPP-GAN
  - Unrolled-GAN
  - VEE-GAN
  - Reg-GAN
  - WGAN
  - WGAN-GP

- Time Efficient
  - GDPP-GAN
  - Unrolled-GAN
  - VEE-GAN
  - WGAN
  - WGAN-GP

<table>
<thead>
<tr>
<th></th>
<th>DCGAN</th>
<th>Unrolled-GAN</th>
<th>VEE-GAN</th>
<th>Reg-GAN</th>
<th>WGAN</th>
<th>WGAN-GP</th>
<th>GDPP-GAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg. Iter. Time (s)</td>
<td>0.0674</td>
<td>0.2467</td>
<td>0.1978</td>
<td>0.1357</td>
<td>0.1747</td>
<td>0.4331</td>
<td><strong>0.0746</strong></td>
</tr>
</tbody>
</table>
What else?

- Data Efficient
- Fast Training Time
- Stabilizes Adversarial Training
- Time Efficient
What else?

- Data Efficient
- Time Efficient
- Fast Training Time
- Stabilizes Adversarial Training
- Robust to poor Initialization

**Fast Training Time**
Why GDPP?

1. No extra trainable parameters (cost-free)
Why GDPP?

1. No extra trainable parameters (cost-free)
2. Unsupervised Setting (No labels)
Why GDPP?

1. No extra trainable parameters (cost-free)
2. Unsupervised Setting (No labels)
3. Stabilizes Adversarial Training
Why GDPP?

1. No extra trainable parameters (cost-free)
2. Unsupervised Setting (No labels)
3. Stabilizes Adversarial Training
4. Time and Data efficient
Why GDPP?

1. No extra trainable parameters (cost-free)
2. Unsupervised Setting (No labels)
3. Stabilizes Adversarial Training
4. Time and Data efficient
5. Architecture & Model Invariant (GAN & VAE)
Why GDPP?

1. No extra trainable parameters (cost-free)
2. Unsupervised Setting (No labels)
3. Stabilizes Adversarial Training
4. Time and Data efficient
5. Architecture & Model Invariant (GAN & VAE)

Yet, Consistently outperforms state-of-the-art
For many more, join us in