Adversarially learned transferable representations

Project Presentation for EE392A - Undergraduate Project

Abhinav Agrawal (14011)

Under the Guidance of:
Prof. Vinay P. Namboodiri
Prof. Tanaya Guha

April 19, 2018

Table of Contents

Introduction

Motivation - Domain Adaptation

- Large amount of data is usually unlabeled
- Annotations are expensive
- Large datasets available but can not represent every situation
- Domain adaptation allows us to learn generalizable functions for desired task

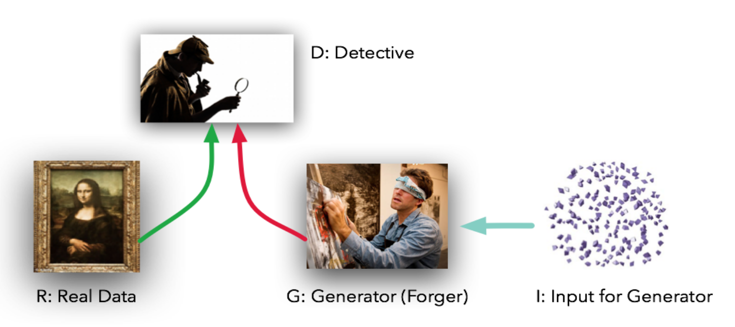
Problem Description - Domain Adaptation

- Attracted researches from different domains
- Setting assumes:
  - A labeled dataset in Source Domain with distribution $P(S)$
  - An unlabeled dataset in Target Domain with distribution $P(T)$
  - Since $P(S) \neq P(T)$ what we can have $P(z|X_s) \sim P(z|X_t)$
- Hence, the task often converts to the one of finding generalizable representations

Lab to Real World

Previous Work

- A generative modeling technique introduced by Goodfellow et al. 2014

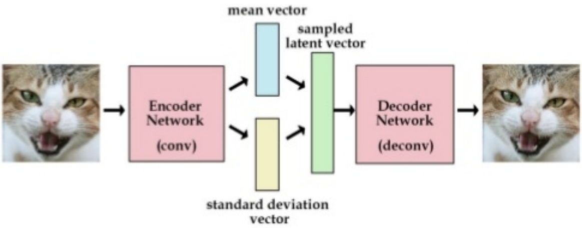


- Hard to train(mode collapse)

Picture - Dev Nag, Medium Blog, 2017

Preliminaries-VAE

- More mathematically understood method of generative modeling

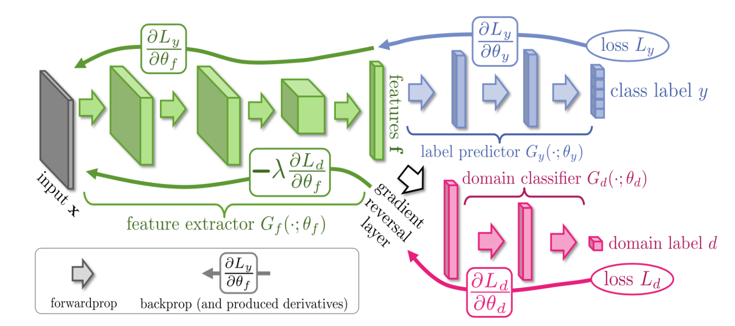


Kevin Frans, "Variational Autoencoders explained" (2016)

- Image visual quality inferior to GANs

Previous Approaches

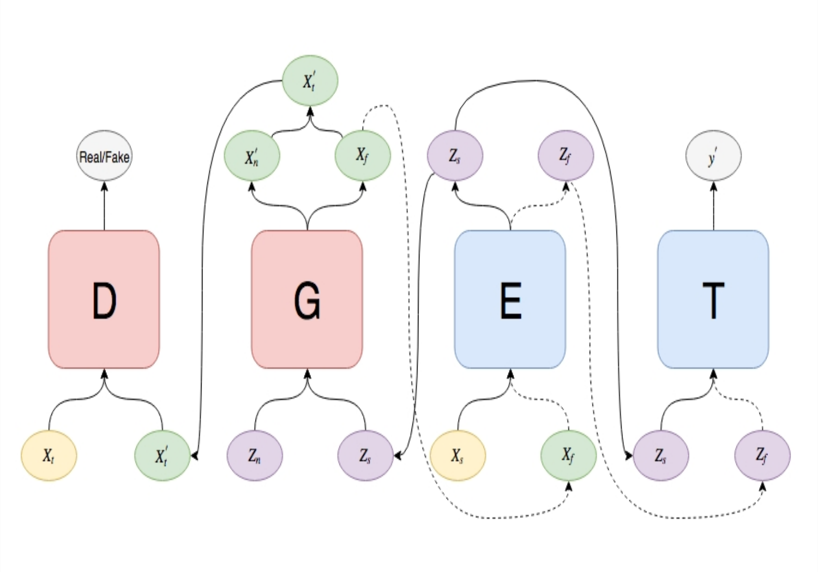
- Deep Adversarial Neural Network(DANN) based approaches were instantiated by the Ganin et al. 2014



- Different variants which worked one reducing the distance of representations(Long et al. 2015, Tzeng et al. 2015)
- Google recently approached the problem with a conditioned-GAN approach(Bousmalis et al. 2017)

Our Approach

Model architecture



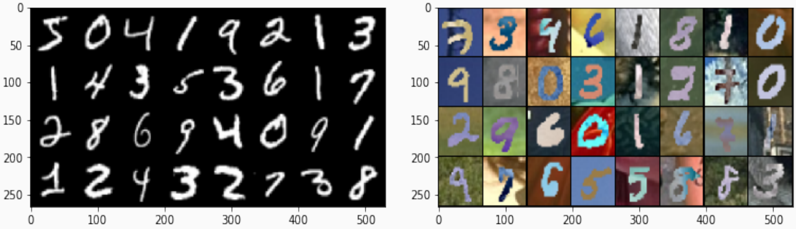
Different Loss functions

$$\begin{aligned}\mathcal{L}_{GAN}(D, G) =& \mathbb{E}_{\mathbf{x}_t}[\log D(\mathbf{x}_t; \theta_D)] + \alpha \mathbb{E}_{\mathbf{x}_s}[\log(1 - D(G(E(\mathbf{x}_s; \theta_G); \theta_G); \theta_D)] \\ & + \beta \mathbb{E}_{\mathbf{z}_n}[\log(1 - D(G(\mathbf{z}_n; \theta_G); \theta_D)]\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{Task}(E, T) =& \mathbb{E}_{\mathbf{x}_s, \mathbf{y}_s, \mathbf{x}_f}[-\mathbf{y}_s^\top \log(T(E(\mathbf{x}_f; \theta_E); \theta_T)) \\ & - \mathbf{y}_s^\top \log(T(E(\mathbf{x}_s; \theta_E); \theta_T))]\end{aligned}$$

$$\mathcal{L}_{prior} = D_{KL}(q(\mathbf{z}_s|\mathbf{x}_s)||p(\mathbf{z}_s))$$

$$\mathcal{L}_C = D_{KL}(q(\mathbf{z}_s|\mathbf{x}_s)||q(\mathbf{z}_f|\mathbf{x}_f))$$



(a) MNIST (b) MNISTM

Figure 1

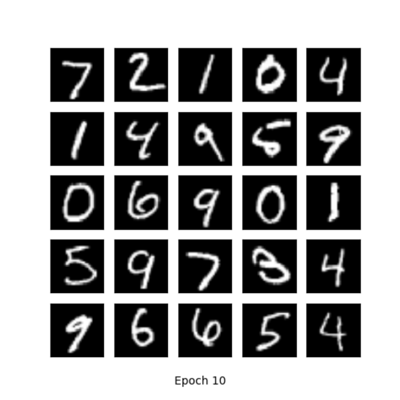
Experiments and Observations

Methods of Training

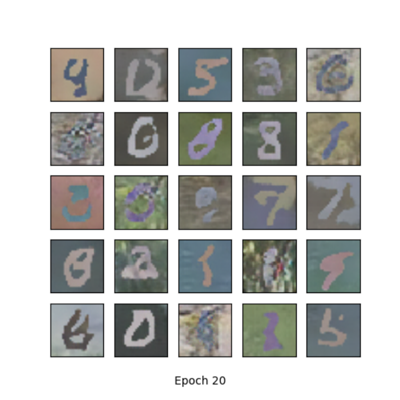
- End-to-end Training :
  - Train the entire model simultaneously(introduce some penalty to fake images)
  - Generates fake images(uncontrolled) but failed to perform on the task
- Sequential Training :
  - Train the encoder first and keep it fixed for the rest of the generation process
  - Performs on the task but failed to generate images

Experiments to Debug

- Training without the task supervision :
  - Generates fake images but no control on generation
  - No negative alignments
  - Possible explanation : Source images not encoded



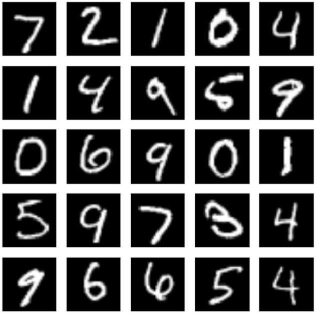
(a) Encoder Input



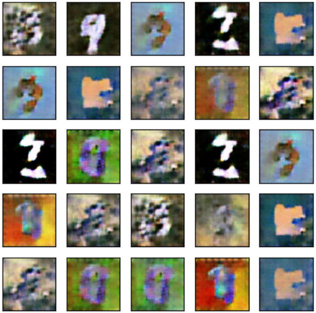
(b) Generator Output

Experiments to Debug

- Training without prior end-to-end :
  - Remove the prior requirement from the setting
  - Generates fake images but no control on generation
  - Modal collapse
  - Fails on classification task
  - Possible explanation : Stronger prior that satisfies both the tasks



Epoch 10



Epoch 18

Experiments to Debug

After finding hints that having a weak prior might be the culprit, we tested how does imposing this strong prior requirement affect the pure classification task.
To do this we used the same encoder-task network combination that we had been using but introduced and removed sampling process to create different losses. We also experimented with inclusion of the ”Prior Loss” function.

Some more tests

- Training the standard Le-Net Classifier :

Table 1: Accuracy on MNISTM dataset(w/o VAE architecture)

Method	Accuracy
Target Only	96.3
Source Only	58.9
Target and Source Both	96.4

- Training with sampling(Prior Loss not included):

Table 2: Accuracy on MNISTM dataset(w/ VAE architecture)

Method	Accuracy
Target Only	96.2
Source Only	58.7
Target and Source Both	93.8

Some more tests

- Training with sampling(Prior Loss included):

Table 3: Accuracy on MNISTM dataset(w/ VAE architecture)

Method	Accuracy
Target Only	95.2
Source Only	55.4
Target and Source Both	11.8

Inferences drawn/Future work

Inferences drawn/Future work

- Inferences:
  - We need a different set of priors than can model such complex distributions
  - Encode more information for controllable generation
- Future Work:
  - Stay with the same architecture
    - Use stronger Gaussian priors
    - Use transformations such as normalizing flows and real valued non-volume preserving(r-nvp) flows
  - Change to different architectural designs
    - Inspired from Cycle GAN and Bi-GAN approach for encoding more information

Acknowledgements

- I express my gratitude for **Prof. Vinay P. Namboodiri** for providing me with valuable support and guidance as and when needed
- I am also equally grateful to **Prof. Tanaya Guha** for agreeing to be the co-advisor on this project.
- A special thanks to **Mr. Vinod Kumar Kurmi** for essential discussions and getting me introduced to the field of domain adaptation.

Questions?