

Hierarchical Probabilistic Image Inpainting

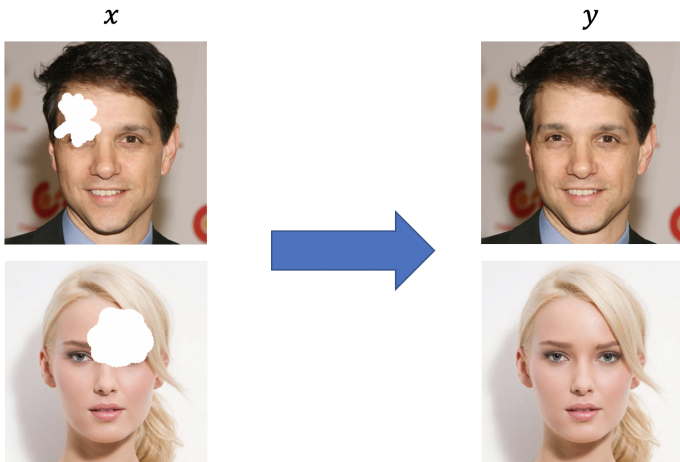
Alex Hung(lingyuh) Zhiqing Sun(zhiqings)

Carnegie Mellon University

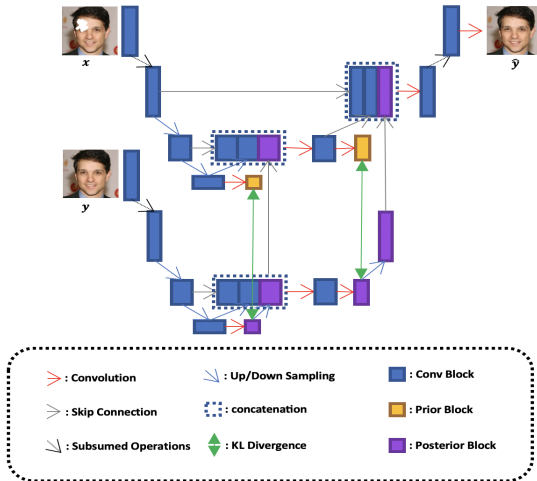
December, 2020

Introduction

In what manner should we model the uncertainty in $x \rightarrow y$?



Network Structure



PGM: Learning

In the learning phase, the incomplete image \mathbf{x} and the complete image \mathbf{y} are observed. We aim to learn n posterior distribution

$$q_{\phi_i}(\mathbf{z}_1|\mathbf{x}, \mathbf{y}), q_{\phi_i}(\mathbf{z}_i|\mathbf{x}, \mathbf{y}, \mathbf{z}_{<i}), \quad (1)$$

n prior distribution

$$p_{\theta_1}(\mathbf{z}_1|\mathbf{x}), p_{\theta_i}(\mathbf{z}_i|\mathbf{x}, \mathbf{z}_{<i}), \quad (2)$$

and a final projection function

$$p_{\theta}(\mathbf{y}|\mathbf{x}, \mathbf{z}_1, \dots, \mathbf{z}_n), \quad (3)$$

such that the variational lower bound (ELBO) is maximized:

$$\begin{aligned} & \mathbb{E}_{\{q_{\phi_i}(\mathbf{z}_i|\mathbf{x}, \mathbf{y}, \mathbf{z}_{<i})\}} [\log p_{\{\theta_i\}}(\mathbf{y}|\mathbf{x}, \{\mathbf{z}_i\})] \\ & - \sum_i D_{KL}(q_{\phi_i}(\mathbf{z}_i|\mathbf{x}, \mathbf{y}, \mathbf{z}_{<i}) \| p_{\theta_i}(\mathbf{z}_i|\mathbf{z}_{<i})) \end{aligned} \quad (4)$$

PGM: Inference

In the inference stage, only the incomplete image \mathbf{x} are observed. Therefore, we can first sample latent variables from the priors:

$$\begin{aligned}\tilde{\mathbf{z}}_1 &\sim p_{\theta_i}(\mathbf{z}_1|\mathbf{x}) \\ \tilde{\mathbf{z}}_i &\sim p_{\theta_i}(\mathbf{z}_i|\mathbf{x}, \mathbf{z}_{<i})\end{aligned}\tag{5}$$

And then perform the inference using $\{\tilde{\mathbf{z}}_i\}$:

$$\tilde{\mathbf{y}} \sim p_{\theta}(\mathbf{y}|\mathbf{x}, \tilde{\mathbf{z}}_1, \dots, \tilde{\mathbf{z}}_n)\tag{6}$$

Results on CelebAMask-HQ Dataset

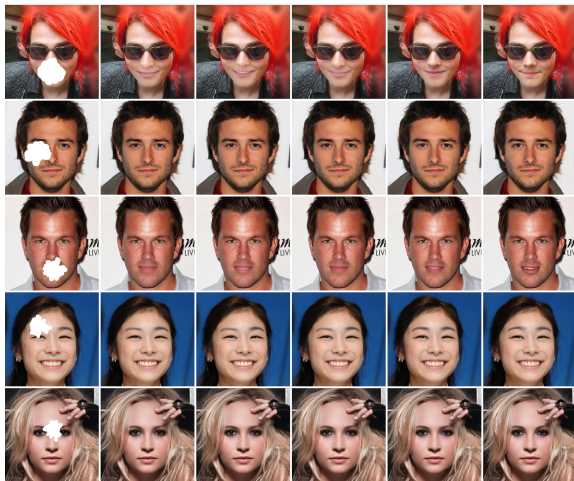


Figure: Input Prior1 Prior2 Prior3 Posterior Original

Conclusion

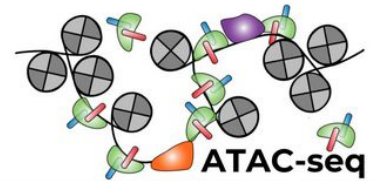
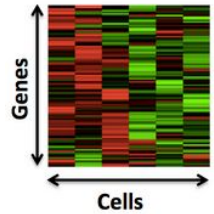
- ▶ We propose a hierarchical probabilistic approach for image inpainting, which utilizes latent variables to model the uncertainty in images.
- ▶ Our empirical results show that our hierarchical probabilistic model is able to successfully complete an up to 256×256 face image with fine-grained details.

Joint Clustering of scRNA-seq and scATAC-seq Data

— Euxhen Hasanaj —

A little bit of biology

- scRNA-seq: captures gene expression profiles of individual cells
- scATAC-seq: captures chromatin accessibility profiles of individual cells
- SNARE-seq: captures both at the same cell level



The Problem

- We can cluster and analyze scRNA-seq and scATAC-seq separately



- When we jointly profile chromatin accessibility and RNA on the same cell, we can utilize this fact to perform joint clustering

The Solution

- Assume underlying distributions for each data modality
 - Gaussian for scRNA-seq
 - Bernoulli for scATAC-seq
- For cells of the same type, require open chromatin regions corresponding to the top differential genes to be open simultaneously

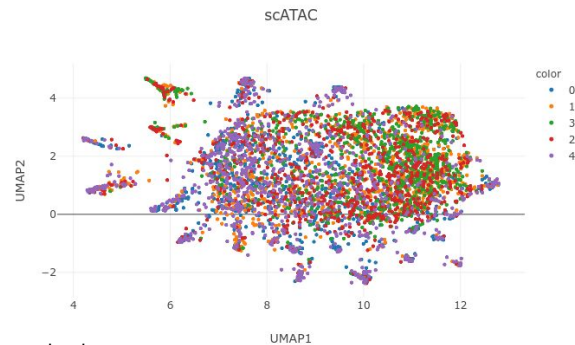
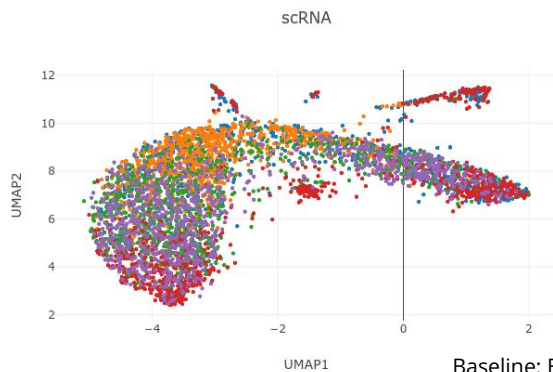
The Solution

- View this as a latent variable PGM
- Likelihood

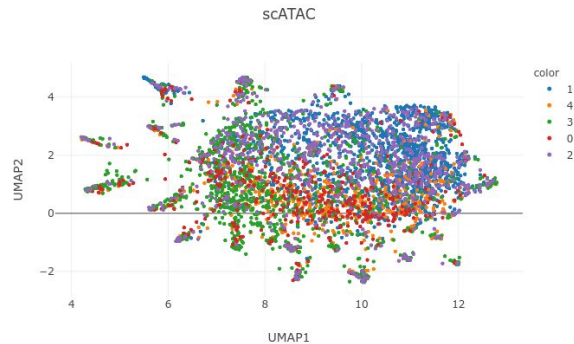
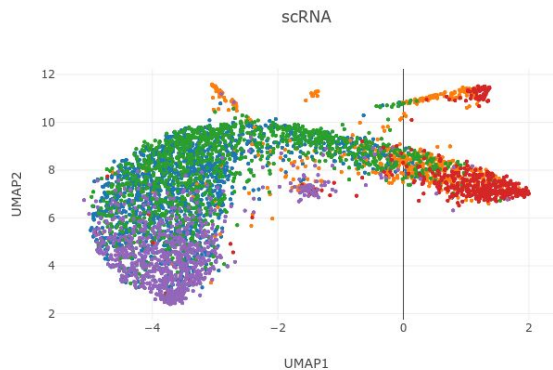
$$p(\mathbf{X}_R, \mathbf{X}_A, \mathbf{Y} \mid \Theta_R, \Theta_A, G) = \frac{1}{Z} \prod_{i,j} (1 - \mathbb{I}(y_i = y_j)(1 - \delta_{i,j})) \prod_i \alpha_{y_i} \\ \prod_i p(\mathbf{X}_R^i \mid y_i, \Theta_R) p(\mathbf{X}_A^i \mid y_i, \Theta_A).$$

- Solve via EM
- Under some assumptions $Z = \left(1 - \sum_{j=1}^k \alpha_j^2\right)^c$

Proof of Concept



Baseline: Ensemble method



Joint clustering

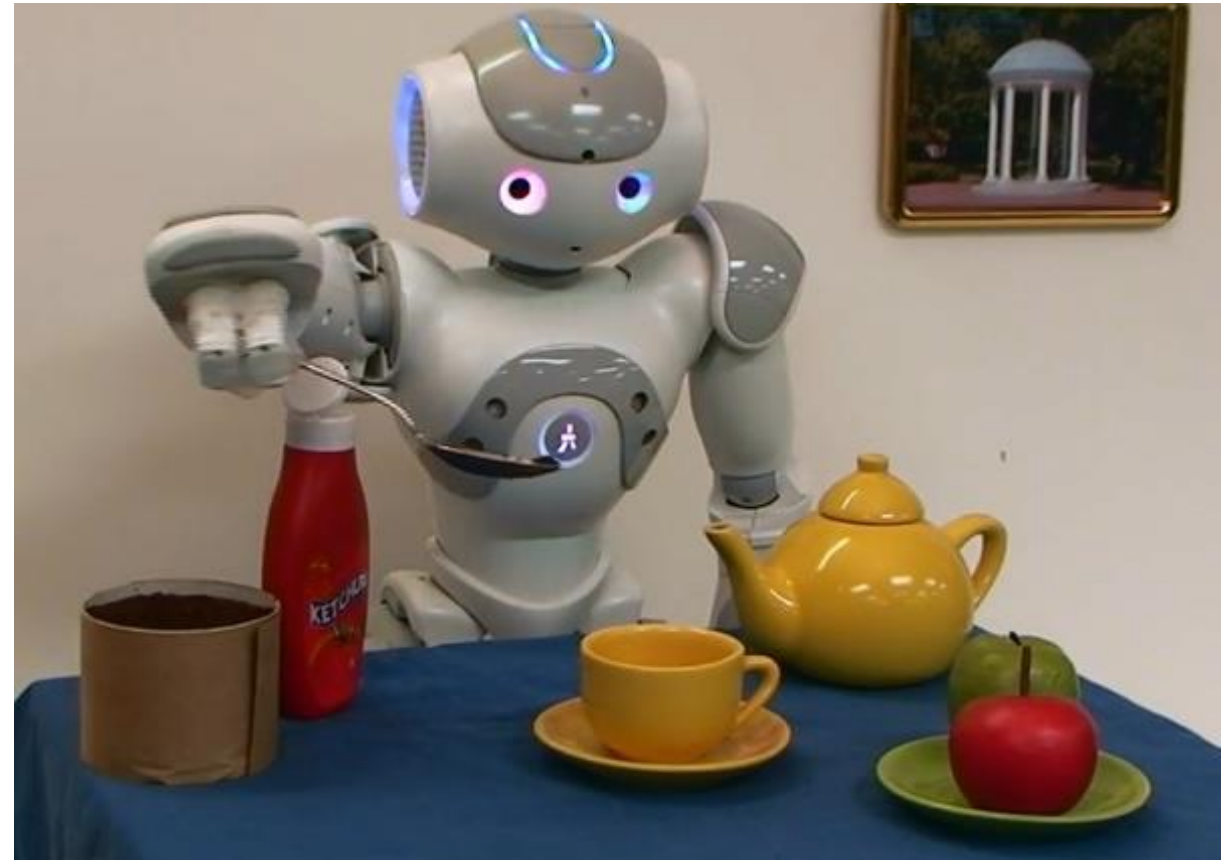
Translating Robot Skills using Cycle Consistency

Anand Bollu, Tanmay Shankar

10th December 2020

10-708 Course Project

Motivation



Motivation

Abstract away low-level details,
and reason over abstractions.

1. Reach for the kettle.
2. Grasp the kettle.
3. Move the
kettle towards the cup.
- ...
- N. Drink tea!



Motivation

Abstractions enable understanding commonalities across tasks.

Good abstractions are invariant to:

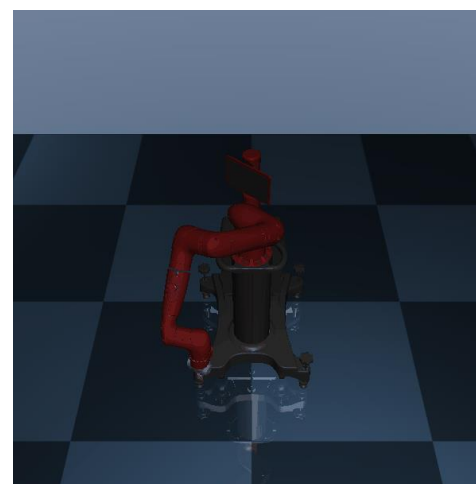
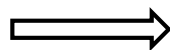
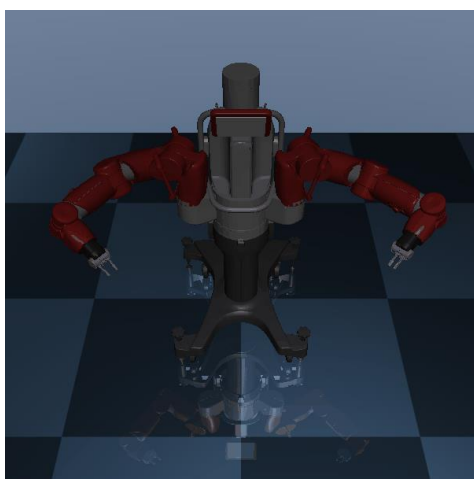
- Robot morphology



High-level Idea

How do we learn representations of these abstractions that are invariant to robot morphology?

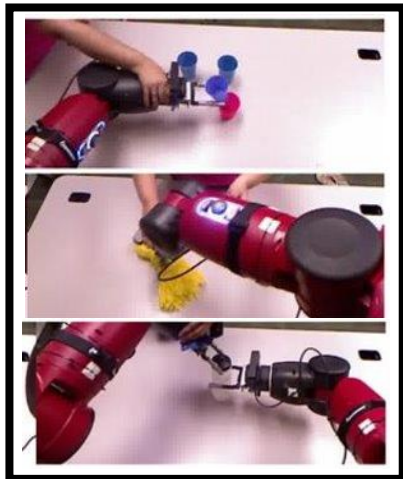
Problem Setting



Source
Morphology

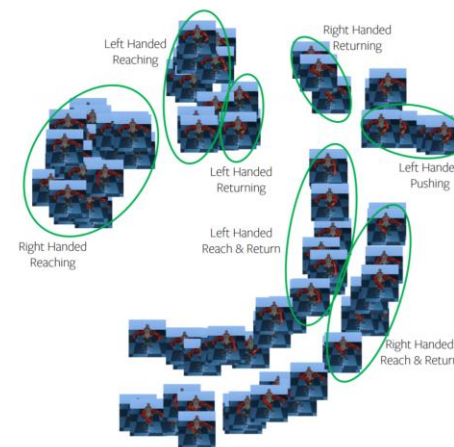
Target
Morphology

Source
Morphology



Demonstration data

Skill
Representation
Learning



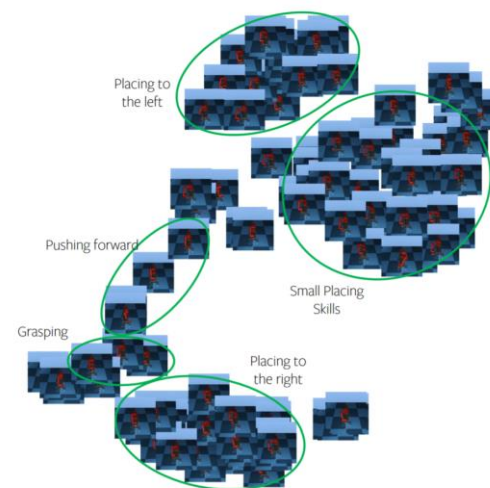
Learnt Skill Representation

Target
Morphology



Demonstration data

Skill
Representation
Learning



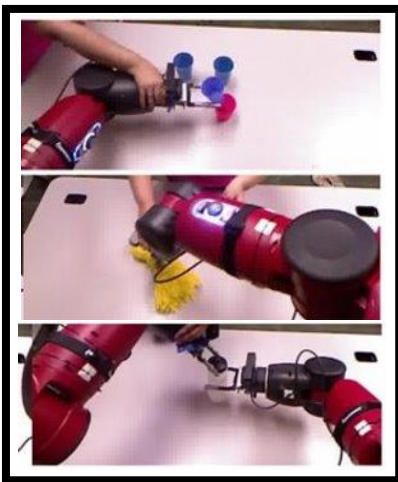
Learnt Skill Representation

Objective

Input Reconstruction:

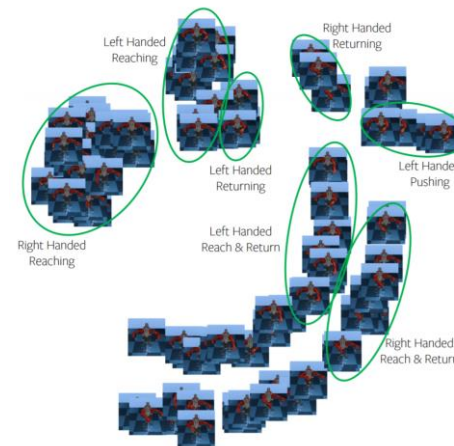
The latent representations should preserve identifying information about the original trajectory.

Source
Morphology



Demonstration data

Skill
Representation
Learning



Learnt Skill Representation

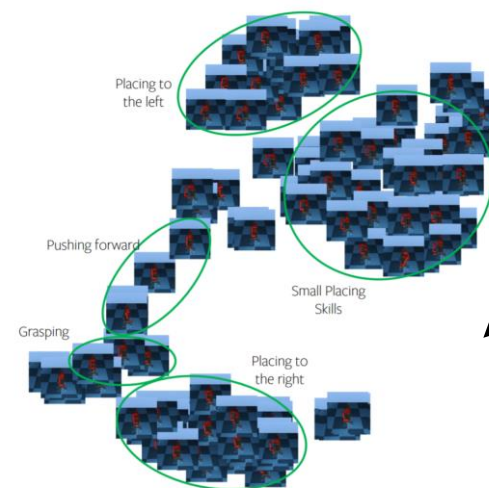
Translation

Target
Morphology



Demonstration data

Skill
Representation
Learning



Learnt Skill Representation

Objective

Input Reconstruction:

The latent representations should preserve identifying information about the original trajectory.

Latent Space Indiscriminability:

The distributions of latent representations should be virtually indistinguishable across different domains.

Objective

Input Reconstruction:

The latent representations should preserve identifying information about the original trajectory.

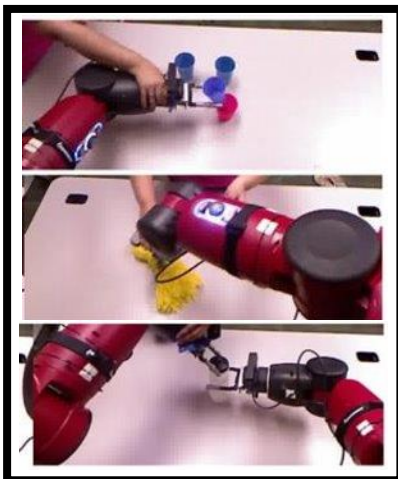
Latent Space Indiscriminability:

The distributions of latent representations should be virtually indistinguishable across different domains.

Cycle Consistency:

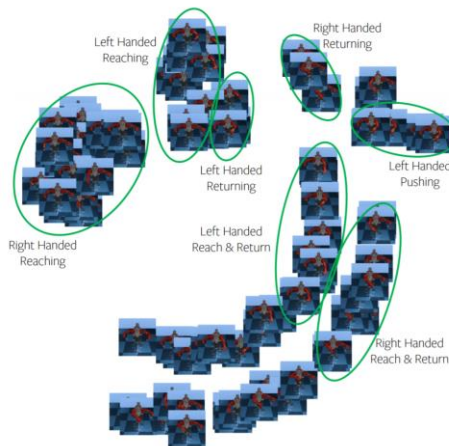
Enforce consistency of original and trajectory translated from source to target and back.

Source
Morphology



Demonstration data

Skill
Representation
Learning



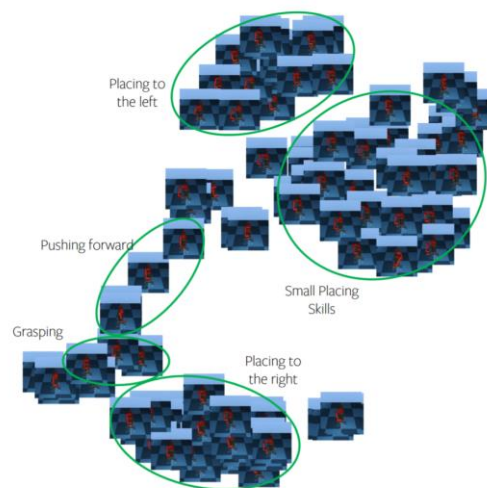
Learnt Skill Representation

Target
Morphology

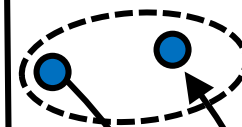


Demonstration data

Skill
Representation
Learning



Learnt Skill Representation



Cycle-
Consistency
Loss

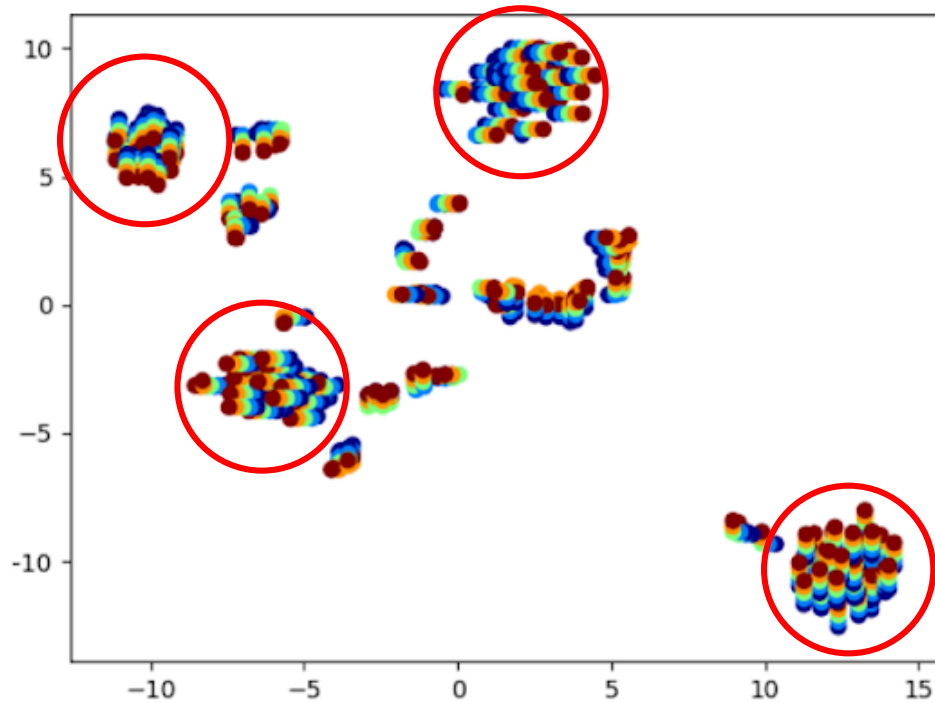
Forward
Translation

Backward
Translation

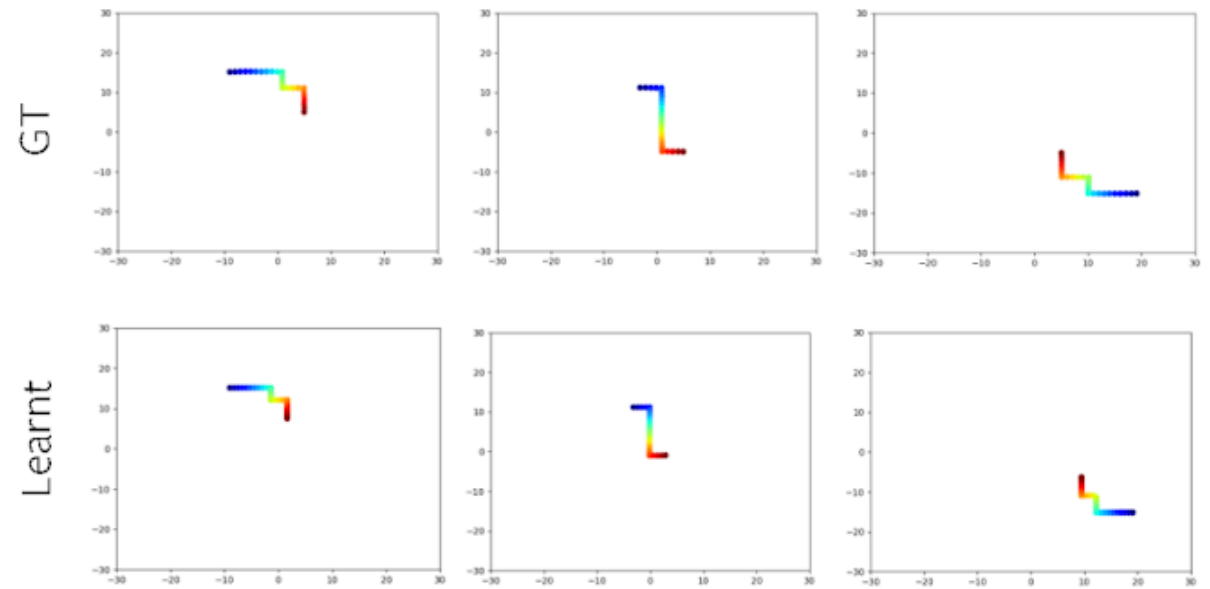
Experiments

Experiments – Learnt Skill Representations

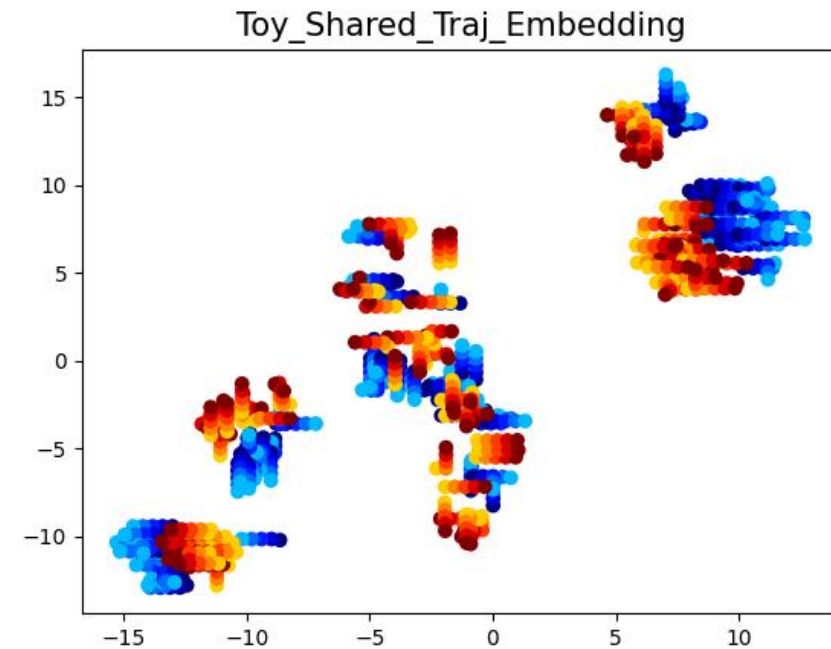
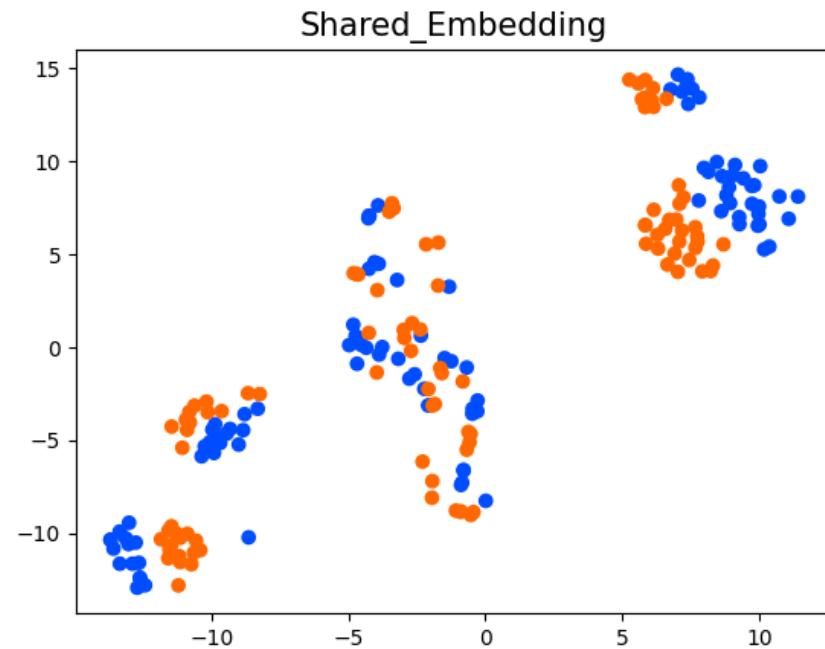
Learnt Embedding Space of ‘Skills’



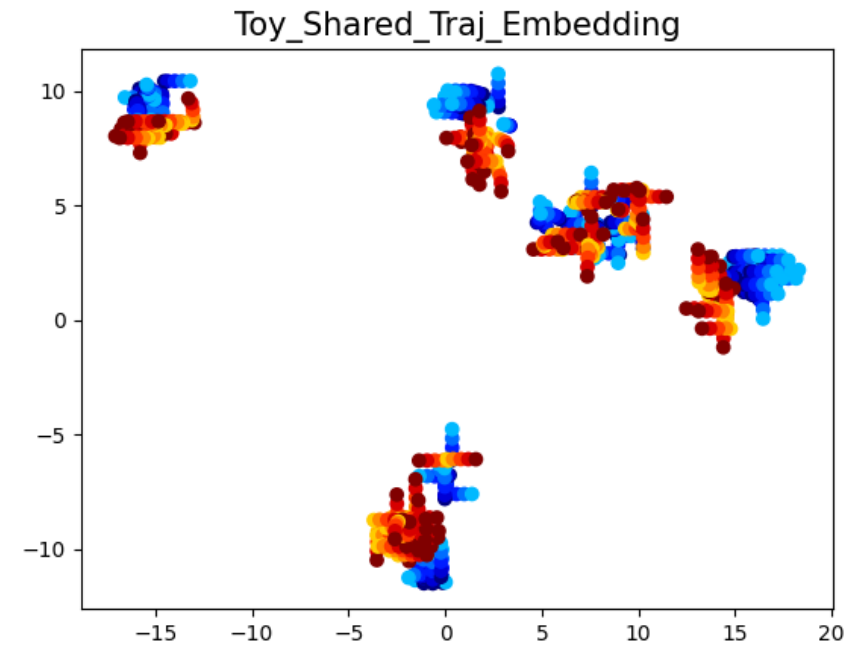
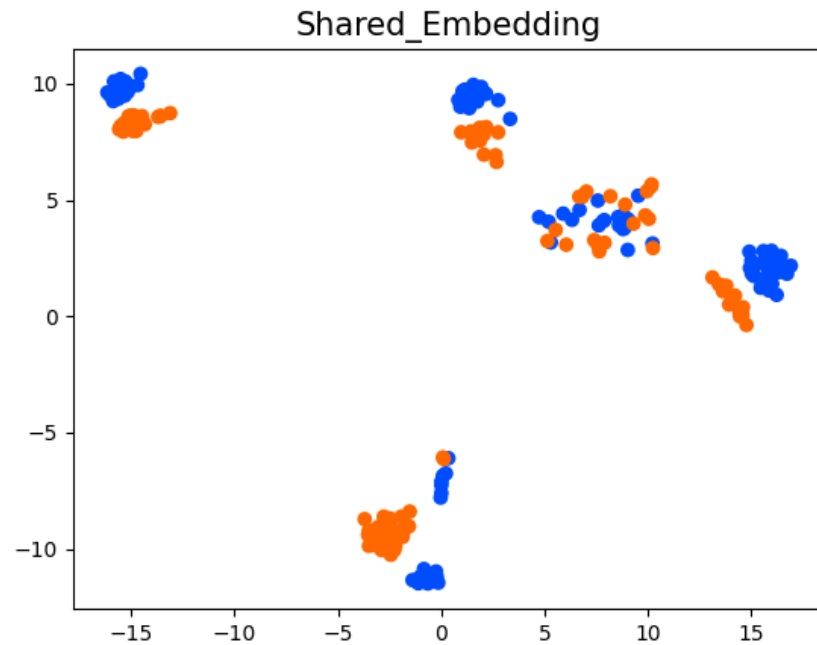
Ground Truth and Reconstructed Trajectories



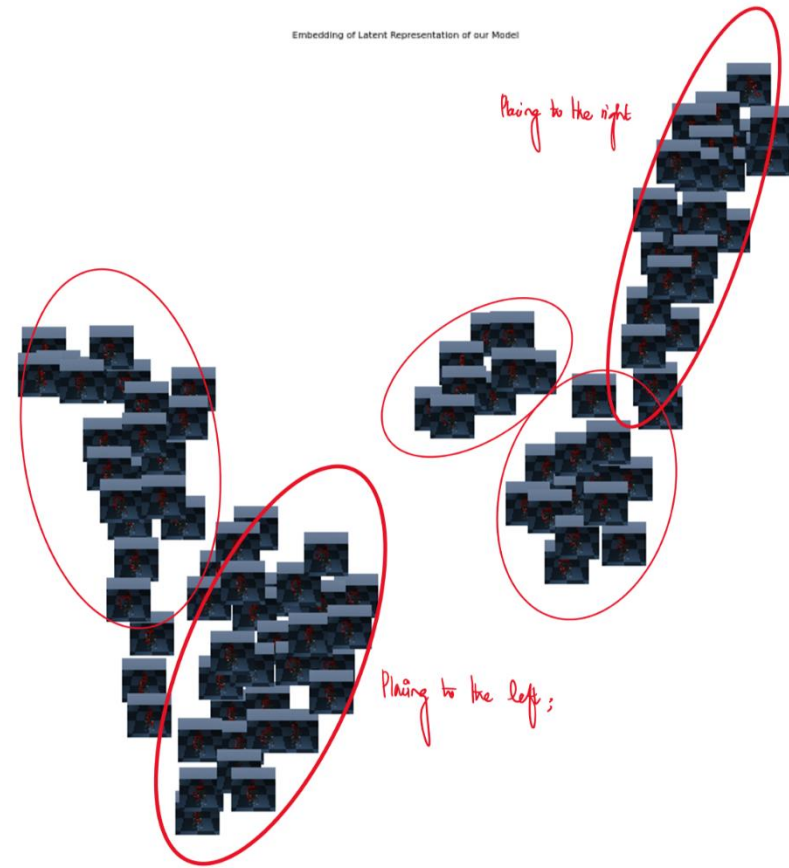
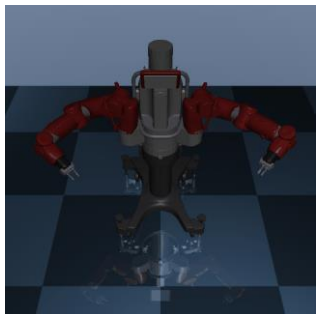
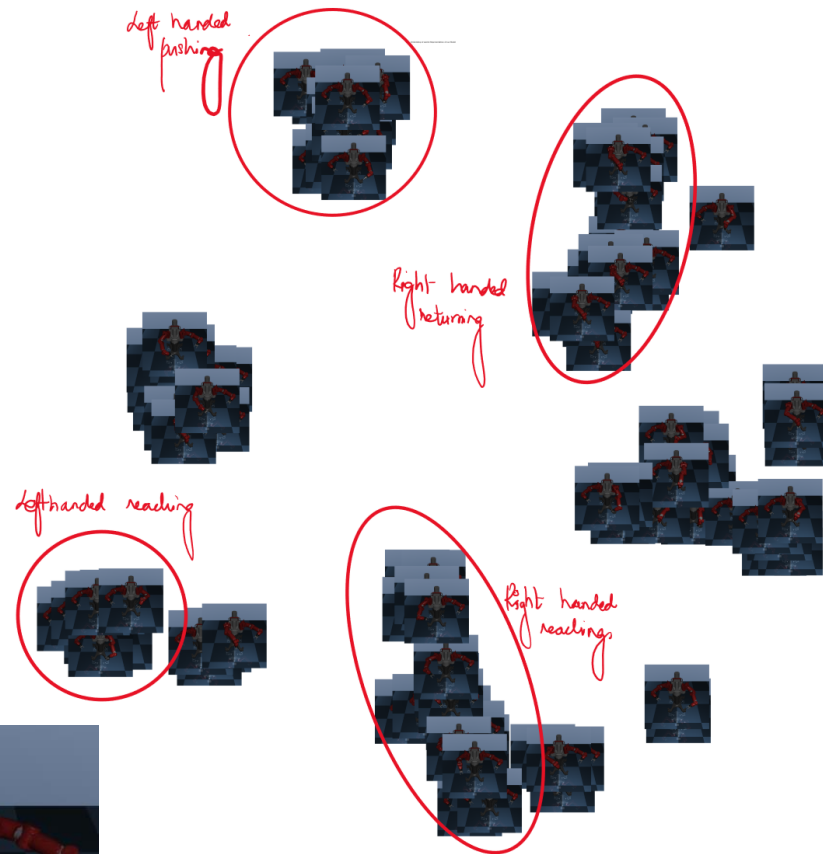
Experiments – Domain Adversarial Training



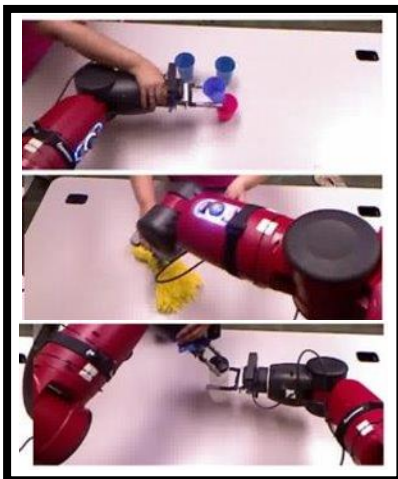
Experiments – Cycle Consistency Training



Experiments – Learnt Skill Representations

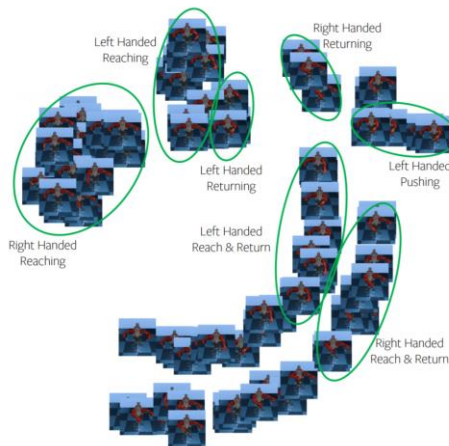


Source
Morphology



Demonstration data

Skill
Representation
Learning



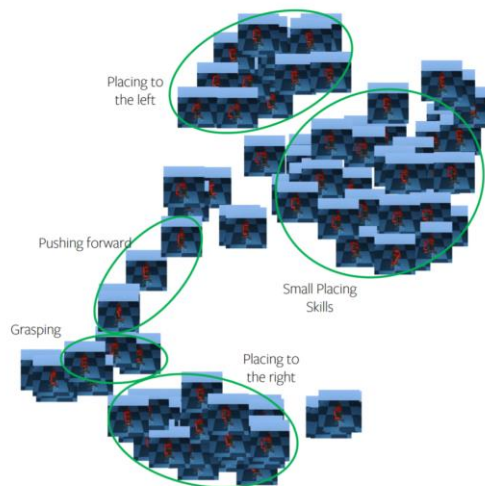
Learnt Skill Representation

Target
Morphology

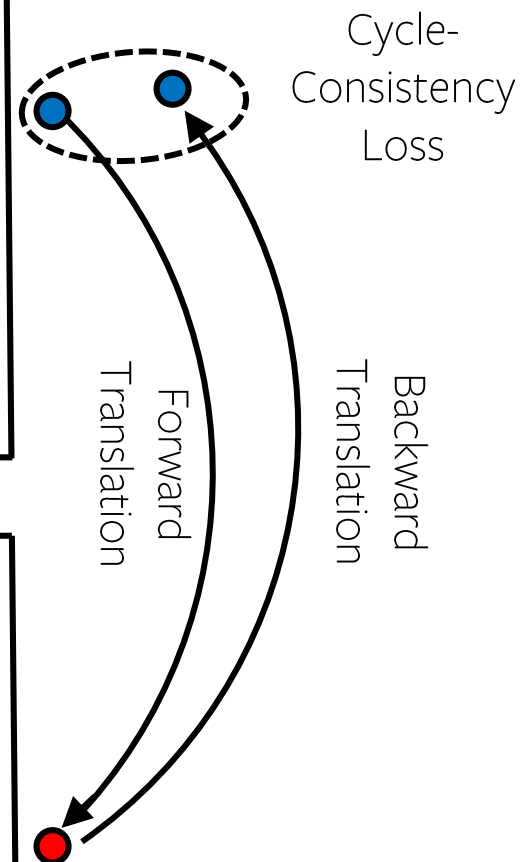


Demonstration data

Skill
Representation
Learning



Learnt Skill Representation



mol-D – Molecular Graph Directionalization to Address Message Passing Neural Network Tottering

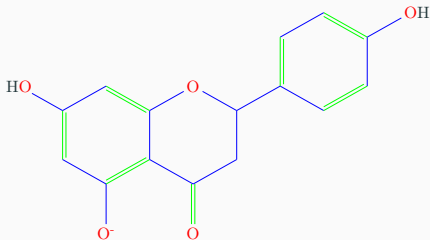
Abhinav Adduri, Monica Dayao, Mustafa Guler

December 9, 2020

Carnegie Mellon University

Message Passing Neural Networks (MPNNs)

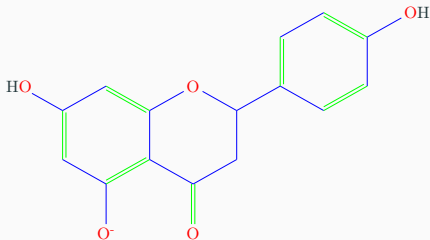
Molecule



Molecular Graph

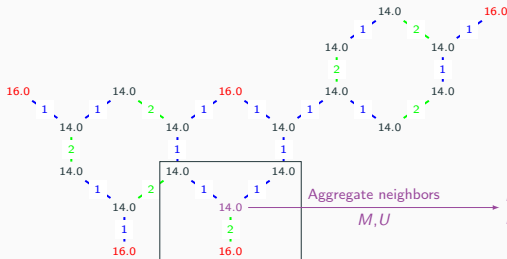
Message Passing Neural Networks (MPNNs)

Molecule



Create Atom &
Bond Features

Molecular Graph



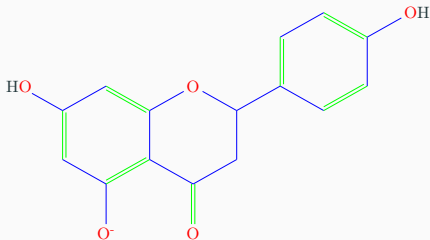
Aggregate neighbors

M, U

$$\begin{aligned} m_i^{t+1} &= \sum_{j \in ne(i)} M_t(h_i^t, h_j^t, \ell_{i,j}) \\ h_i^{t+1} &= U_t(h_i^t, m_i^{t+1}) \end{aligned}$$

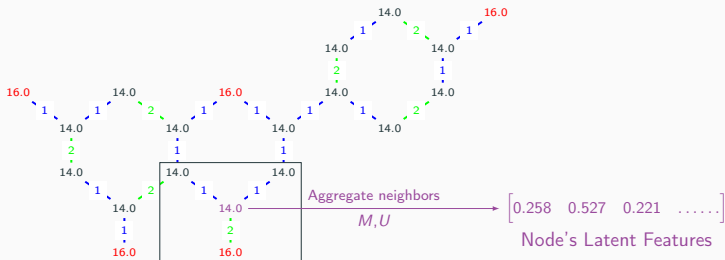
Node's Latent Features

Message Passing Neural Networks (MPNNs)

[illegible]

Create Atom & Bond Features

Molecular Graph



A Potential Problem with MPNNs

- Messages between neighbors bounce back and forth

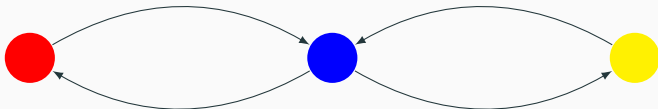


$$t = 0$$

¹Yang et al., "Analyzing learned molecular representations for property prediction".

A Potential Problem with MPNNs

- Messages between neighbors bounce back and forth

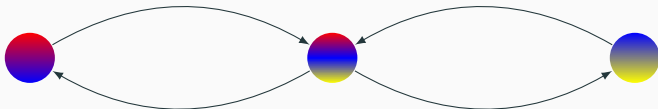


$$t = 0$$

¹Yang et al., "Analyzing learned molecular representations for property prediction".

A Potential Problem with MPNNs

- Messages between neighbors bounce back and forth

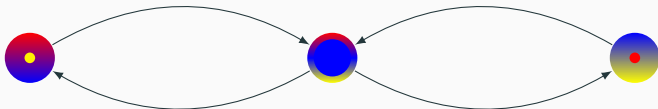


$$t = 1$$

¹Yang et al., "Analyzing learned molecular representations for property prediction".

A Potential Problem with MPNNs

- Messages between neighbors bounce back and forth

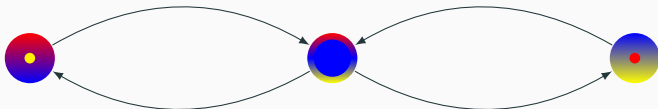


$$t = 2$$

¹Yang et al., "Analyzing learned molecular representations for property prediction".

A Potential Problem with MPNNs

- Messages between neighbors bounce back and forth

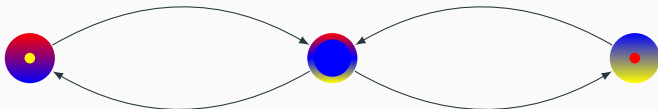


- Tottering

¹Yang et al., "Analyzing learned molecular representations for property prediction".

A Potential Problem with MPNNs

- Messages between neighbors bounce back and forth

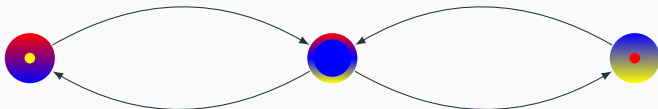


- Tottering
- Possible solution – pass messages along directed edges (D-MPNN)¹

¹Yang et al., “Analyzing learned molecular representations for property prediction”.

A Potential Problem with MPNNs

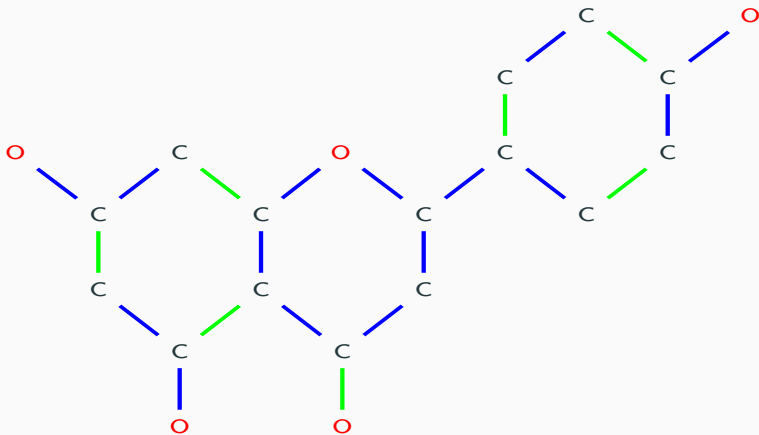
- Messages between neighbors bounce back and forth



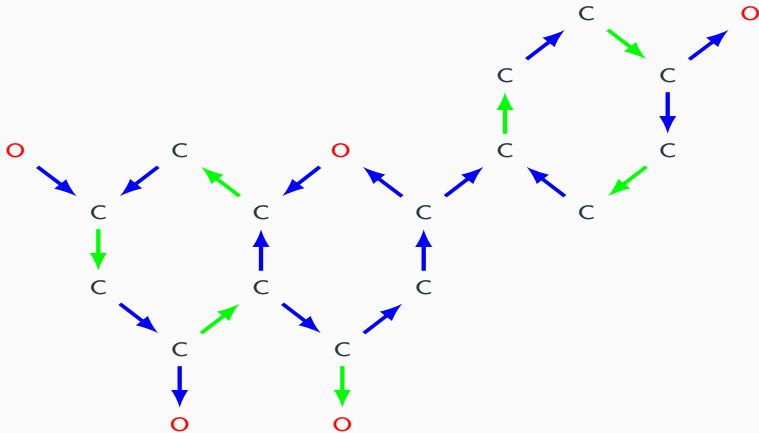
- Tottering
- Possible solution – pass messages along directed edges (D-MPNN)¹
- Our alternative – convert undirected molecule to a directed analogue

¹Yang et al., “Analyzing learned molecular representations for property prediction”.

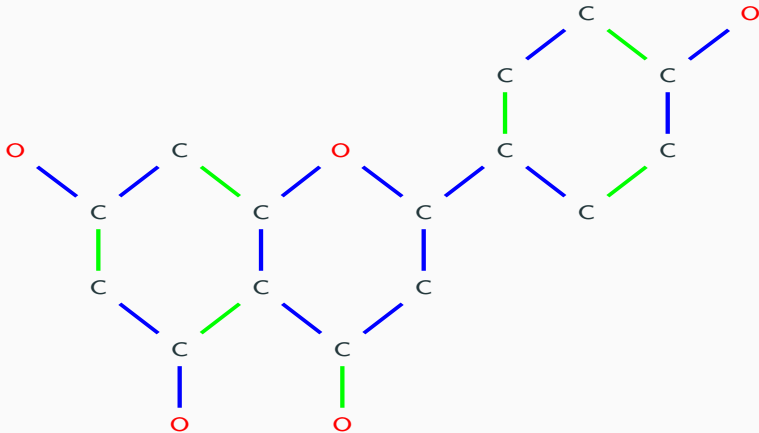
Undirected \rightarrow Directed – Directionality Assignment



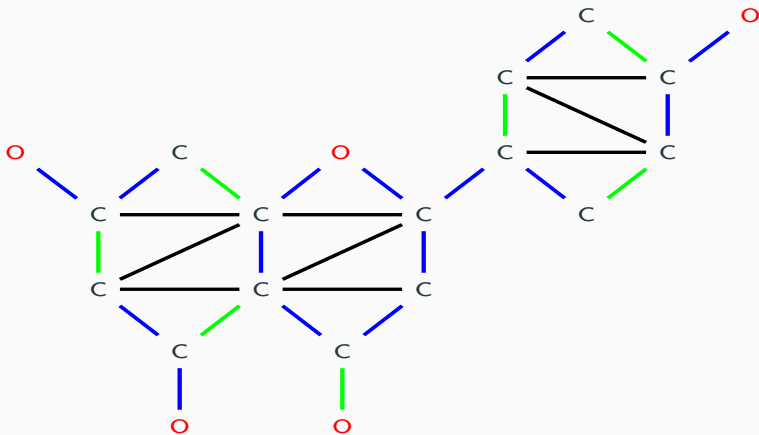
Undirected \rightarrow Directed – Directionality Assignment



Undirected \rightarrow Directed – Triangulation

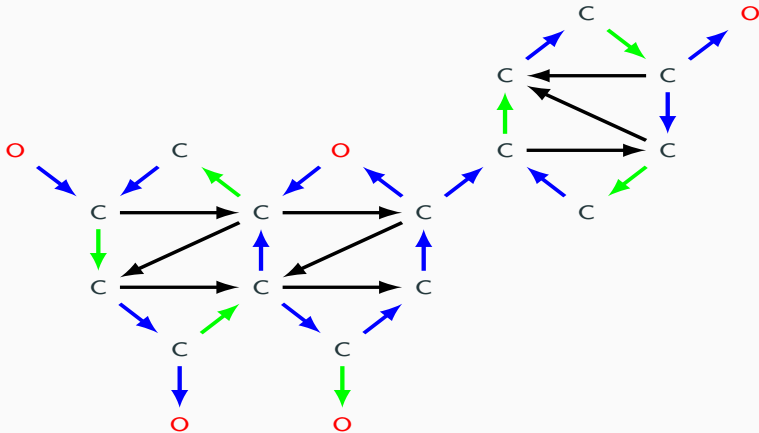


Undirected \rightarrow Directed – Triangulation²

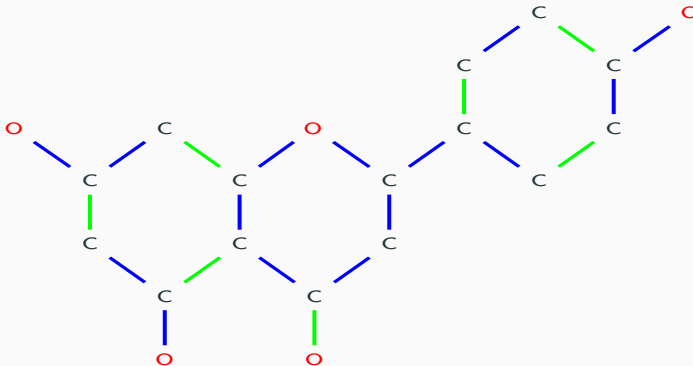


²Berry et al., "Maximum cardinality search for computing minimal triangulations of graphs".

Undirected \rightarrow Directed – Triangulation

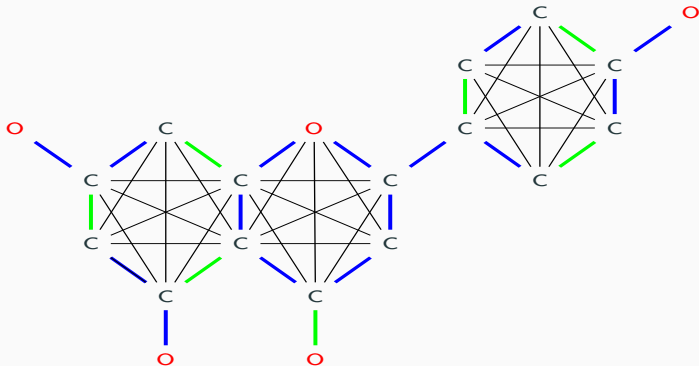


Undirected \rightarrow Directed – Junction Trees²

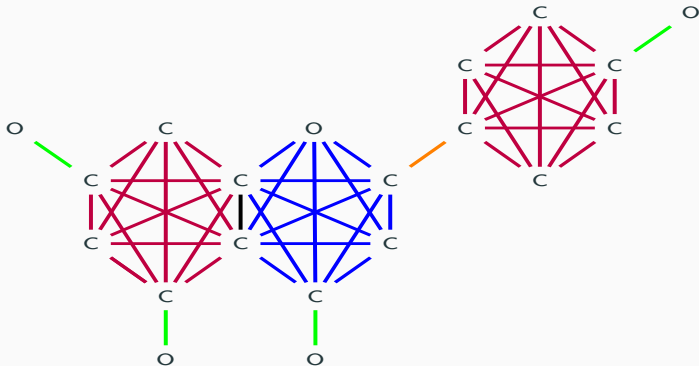


²Jin, Barzilay, and Jaakkola, "Junction tree variational autoencoder for molecular graph generation".

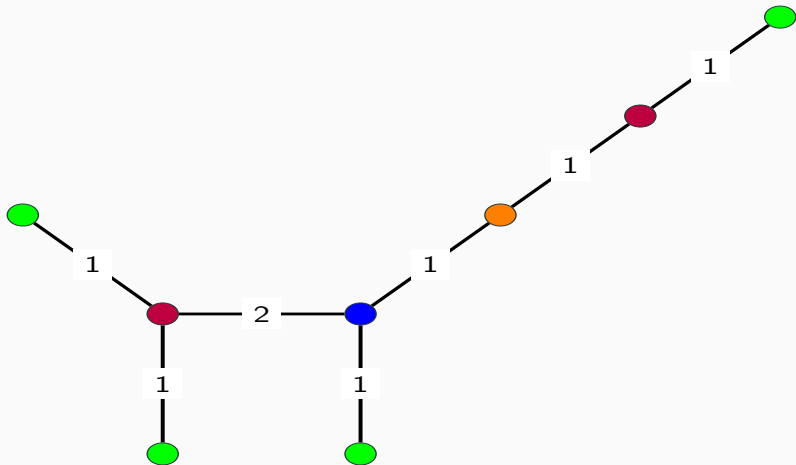
Undirected \rightarrow Directed – Junction Trees



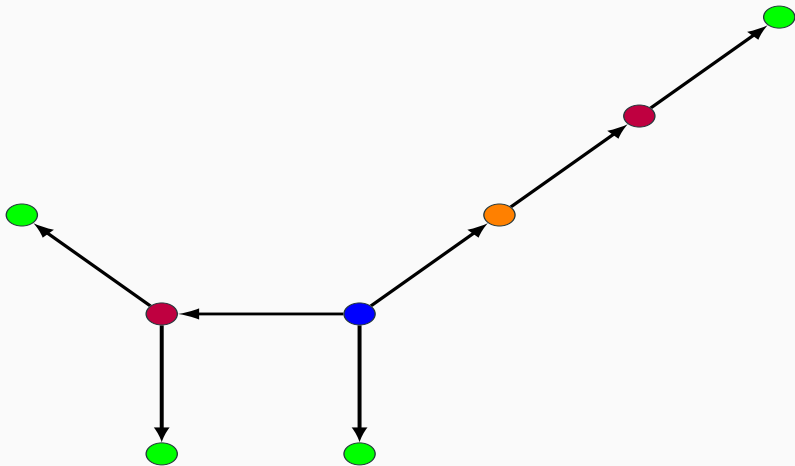
Undirected \rightarrow Directed – Junction Trees



Undirected \rightarrow Directed – Junction Trees



Undirected \rightarrow Directed – Junction Trees



Validation Accuracy – BBBP

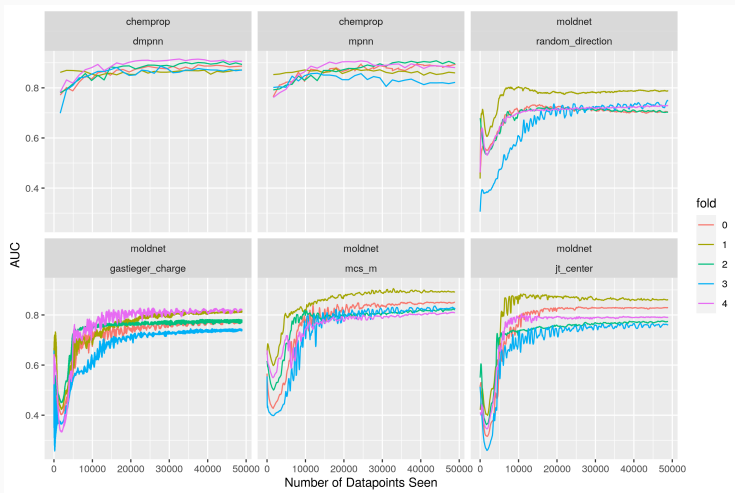


Figure 1: Validation AUC for BBBP Dataset

Validation Accuracy – Tox21

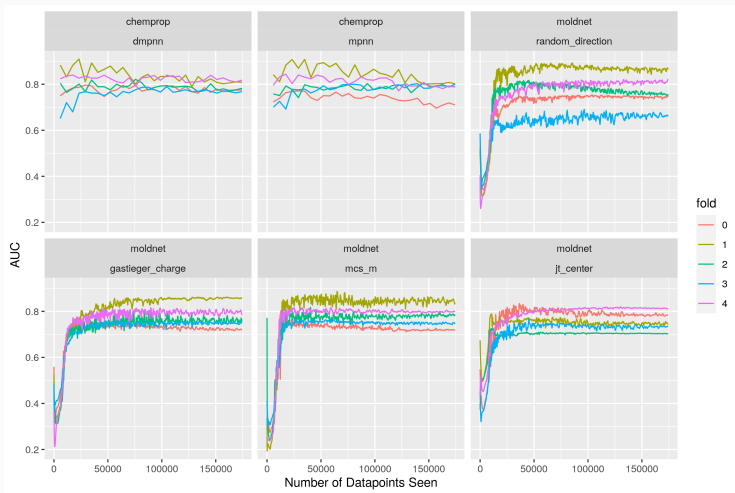


Figure 2: Validation AUC for Tox21 Dataset

Test Accuracy

Package	Method	Dataset	Mean Test AUC
chemprop	D-MPNN	bbbp	0.914338
chemprop	MPNN	bbbp	0.913947
moldnet	Random Direction	bbbp	0.756267
moldnet	Gastieger Charge	bbbp	0.777256
moldnet	Triangulation	bbbp	0.858727
moldnet	Junction Tree	bbbp	0.799936
chemprop	D-MPNN	tox21	0.790662
chemprop	MPNN	tox21	0.775361
moldnet	Random Direction	tox21	0.779763
moldnet	Gastieger Charge	tox21	0.756799
moldnet	Triangulation	tox21	0.739608
moldnet	Junction Tree	tox21	0.788076

- [1] Kevin Yang et al. “Analyzing learned molecular representations for property prediction”. In: *Journal of chemical information and modeling* 59.8 (2019), pp. 3370–3388.
- [2] Anne Berry et al. “Maximum cardinality search for computing minimal triangulations of graphs”. In: *Algorithmica* 39.4 (2004), pp. 287–298.
- [3] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. “Junction tree variational autoencoder for molecular graph generation”. In: *arXiv preprint arXiv:1802.04364* (2018).

RL4RL: Neural architecture search for deepQ learning

Haotian Teng, Yang Ping Kuo, Xiaoyue Cui

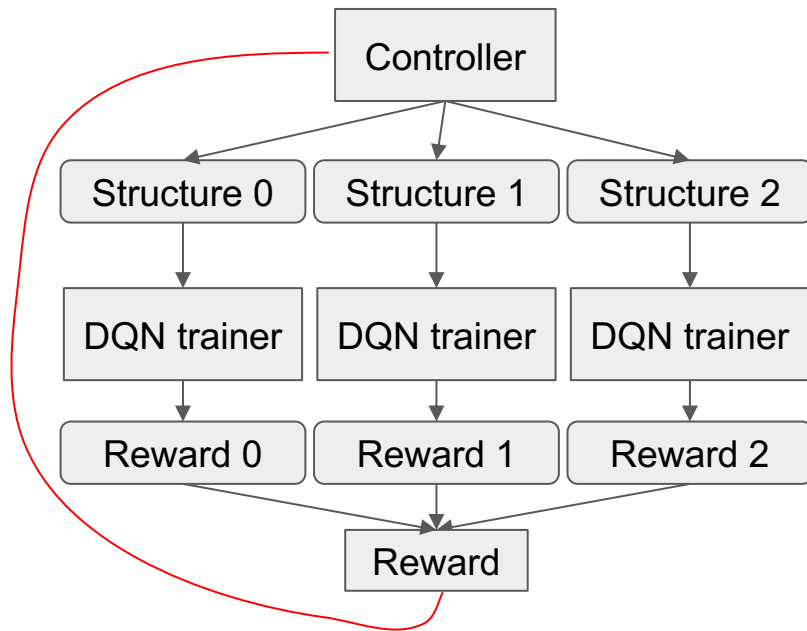
Neural architecture search with RL, for RL

Use reinforcement learning for some tasks (play the Pong game)

Agent: a CNN

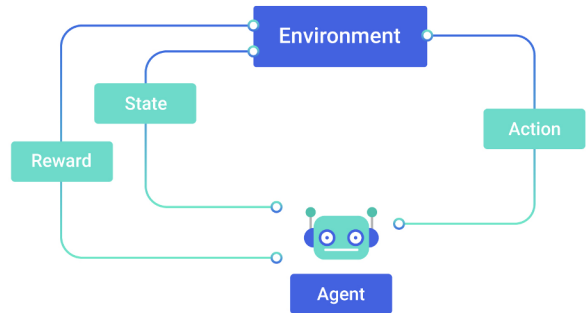
How do we design a good architecture for it?

Search architecture space with RL!



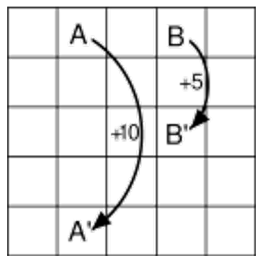
Deep Q learning

- Determine a set of actions that leads to maximum reward
- The q function tells the agent which action is optimal at every given state



$$Q(s, a) = r(s, a) + \gamma \max_a Q(s', a)$$

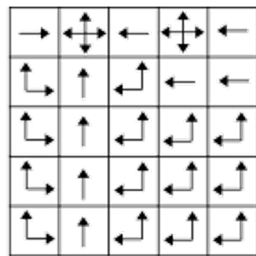
- Impossible to keep track of when state space is huge
- Neural network!!!



a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

b) V^*



c) π^*

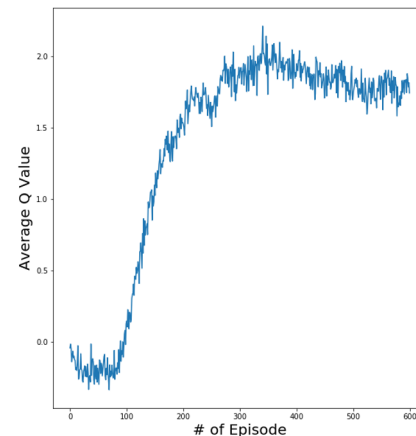
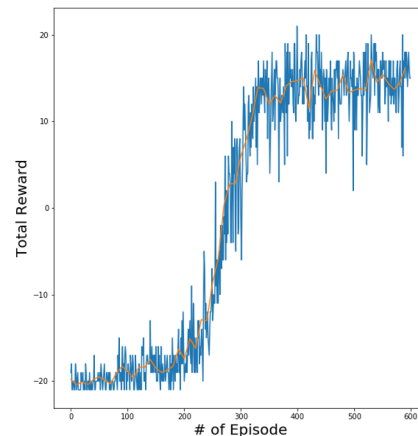
Atari: Pong

$$Q(s, a) = r(s, a) + \gamma \max_a Q(s', a)$$

- States: 210 × 160 pixel RGB images

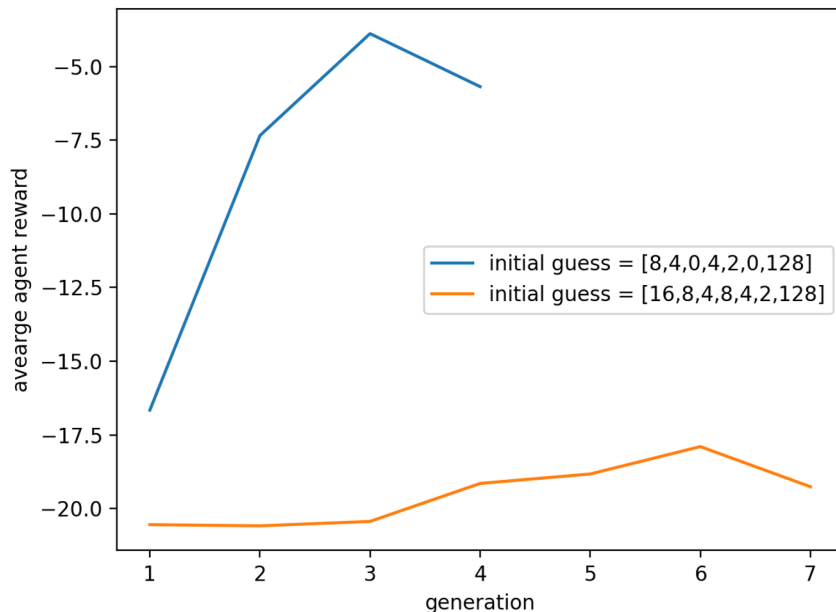
$$|\text{state}| = 256^{(3 \times 210 \times 160)}$$

- Train neural network to minimize predicted q value of every state
- Hard: training with output of the neural net instead of fixed objective
- No guaranteed convergence
- Heavily depended on hyperparameters and architecture

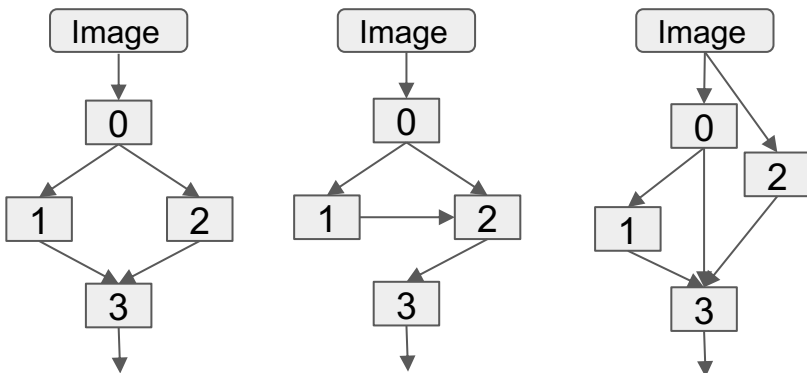
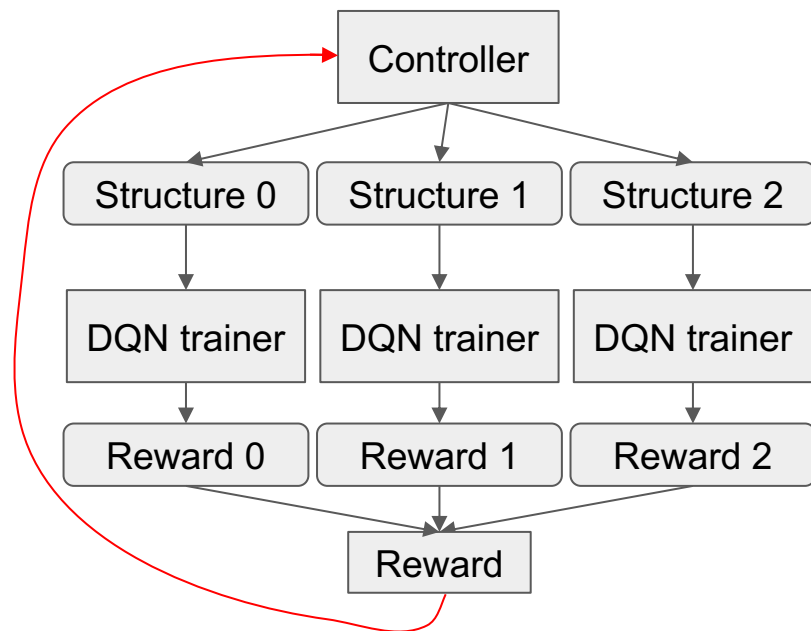
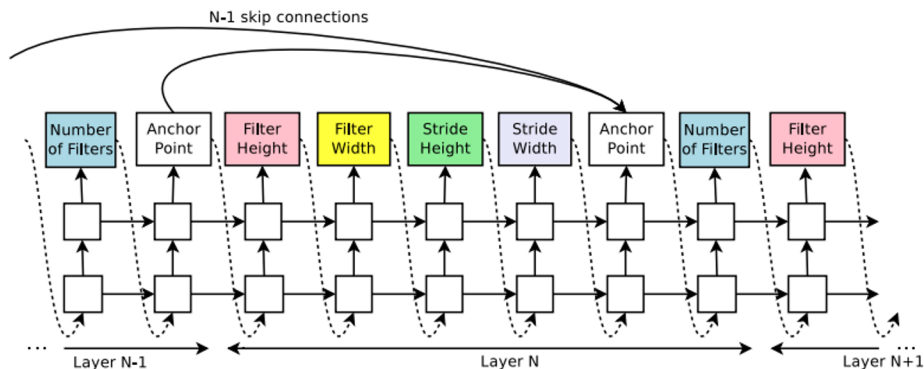


Search structure parameters using Evolutionary algorithm

- Convolutional layer 1: kernel size, stride, padding
- Convolutional layer 2: kernel size, stride, padding
- Fully connected layer



Search Neural network architecture using RNN



3 Convolutional neural network structures with positive reward.

$$\frac{1}{m} \sum_{k=1}^m \sum_{t=1}^T \nabla_{\theta_c} \log P(a_t | a_{(t-1):1}; \theta_c) R_k$$

Discussion

Current reward function only consider the final agent performance. The time to convergence can be added to the reward function to search for structure with faster convergence rate.

The neural network architecture search require huge amount of time as every agent has to be trained separately to converge. The weight of the neural network can be reused to achieve faster training convergence.

Deep Generative Models for FPGA Placement with Disentangled Representation

Shan Xue, Haiguang Liao

2020.12.10

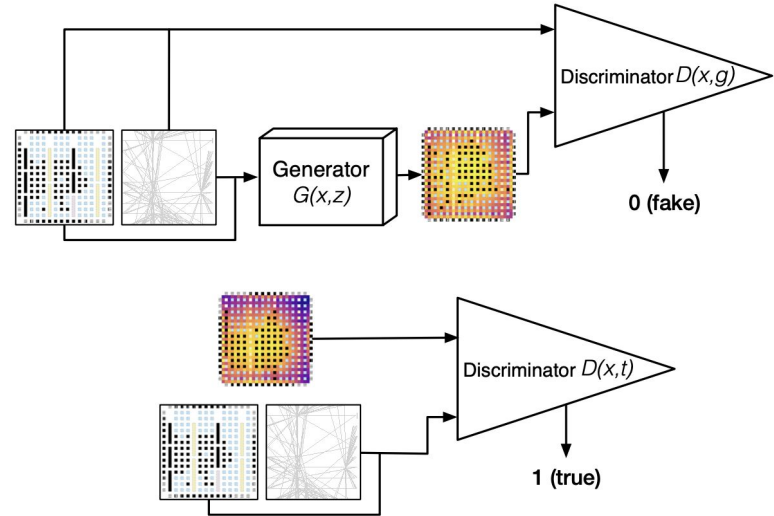
10708 Project Spotlight

Background

FPGA Placement and Route

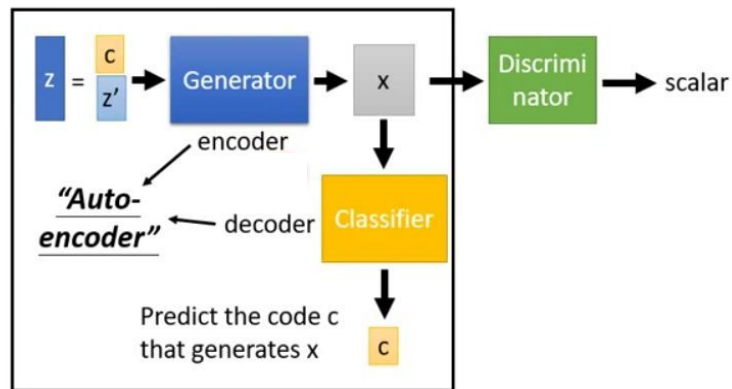
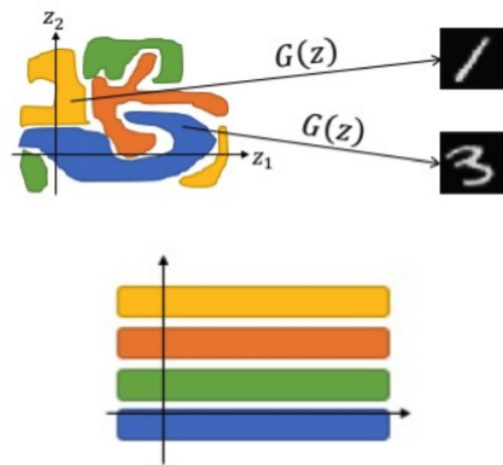


PnR Predictive Model (Yu 2019)



Background

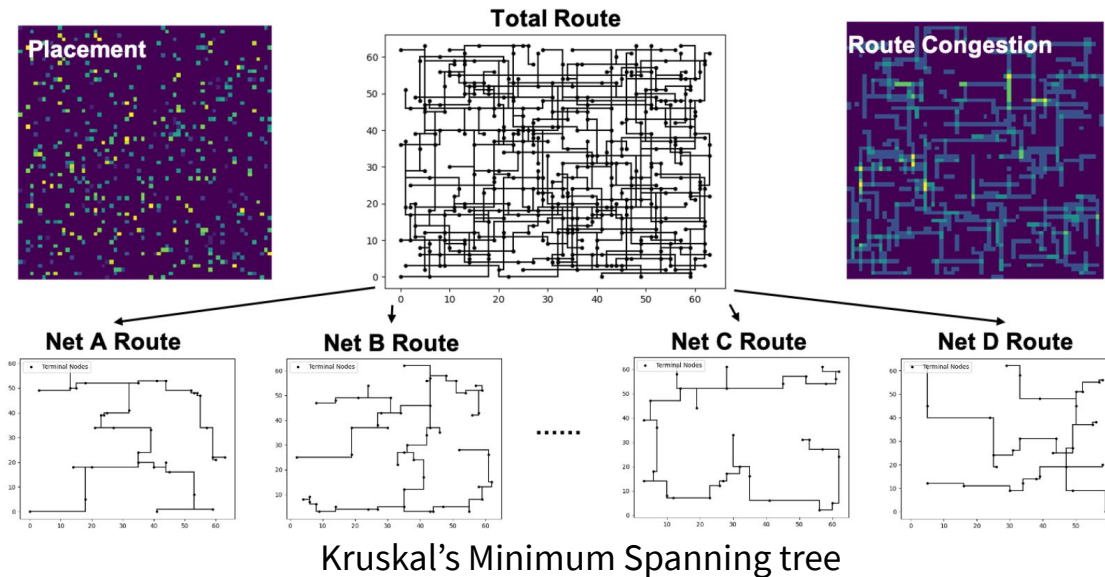
GAN for Disentangled representation



$$\min_G \max_D L_I(D, G) = \mathbb{E}_{x \sim P_{real}} [\log(D(x))] + \mathbb{E}_{P_G} [\log(1 - D(x))]$$

$$\min_G \max_D L_I(D, G) = L(D, G) - \lambda I(c; G(z, c))$$

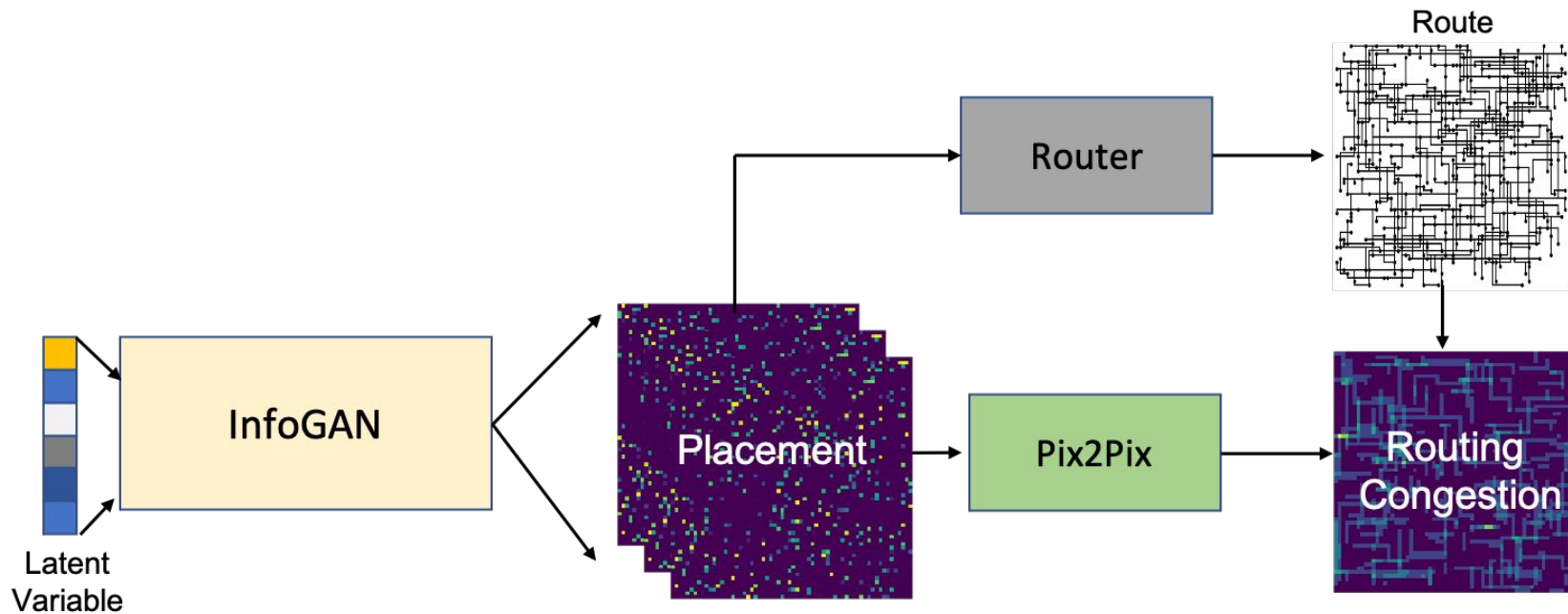
Method - Data Generation



Placement Image:
$$G_p(i, j) = \sum_{l=1}^p \mathbb{1}[(i, j) \in V_{N_l}] l$$

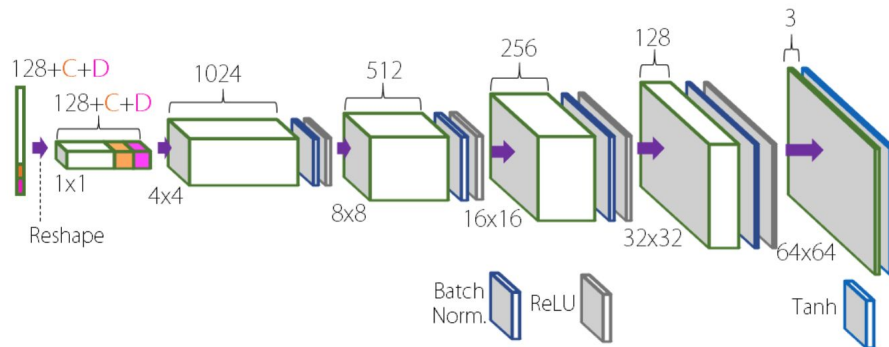
Routing Image:
$$G_r(i, j) = \sum_{i=1}^p \mathbb{1}[(i, j) \in R_{N_i}]$$

Method - Flow

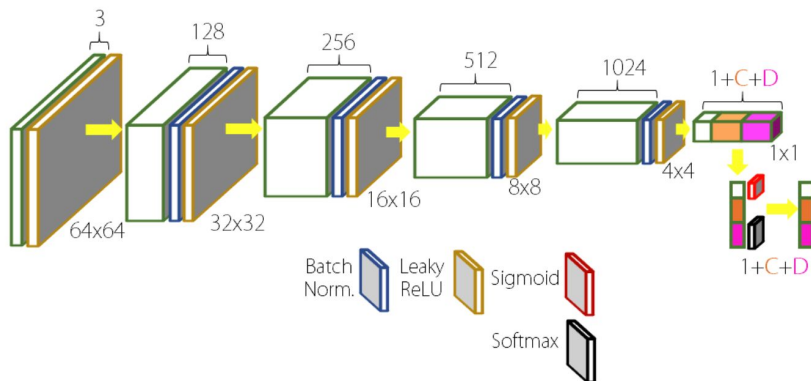


Method - infoGAN

Generator Networks

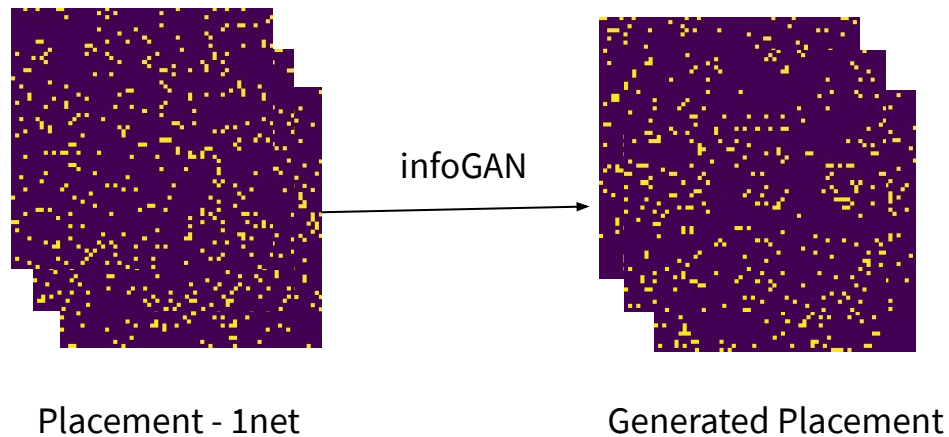
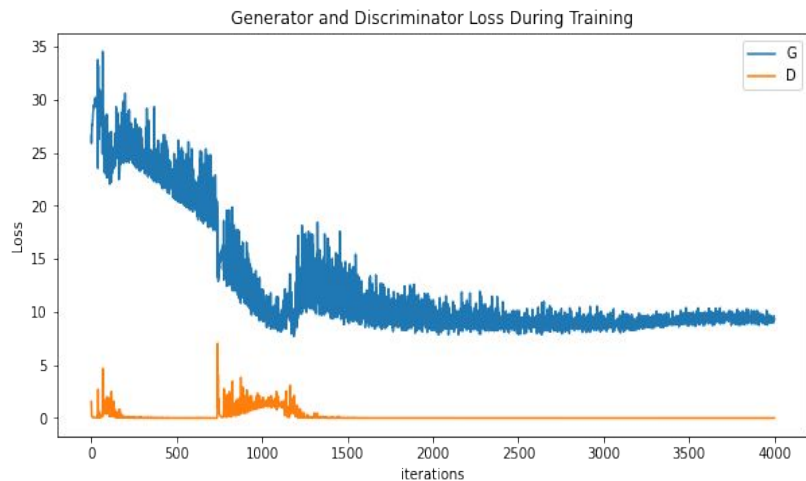


Discriminator Networks



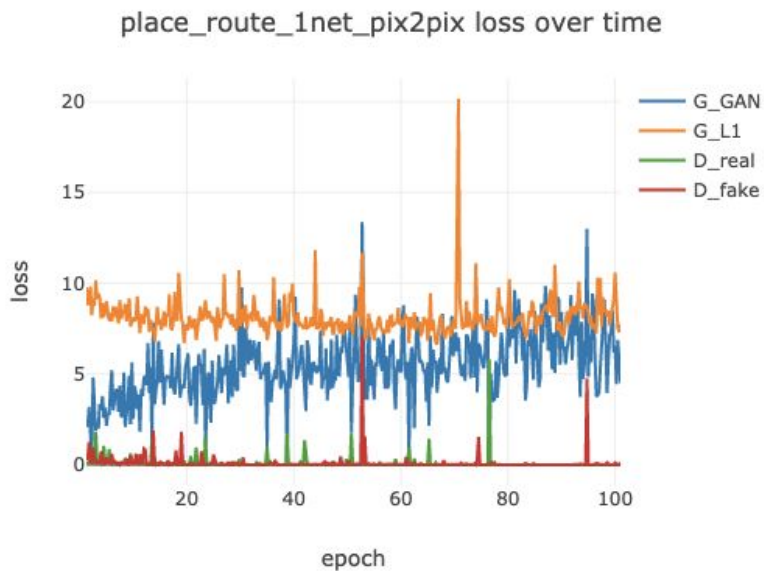
Results

Info GAN results

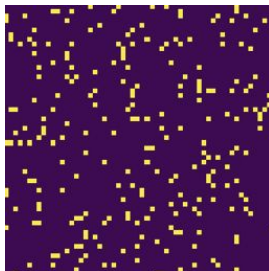


Results

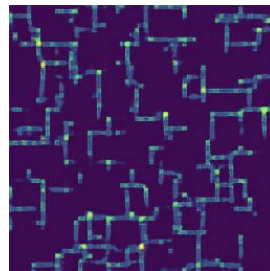
Pix2Pix Model for 1 net design



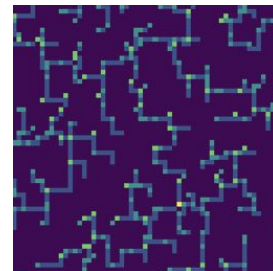
Epoch 5



Real Place

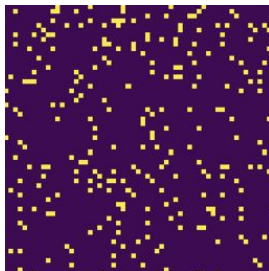


Fake Congestion

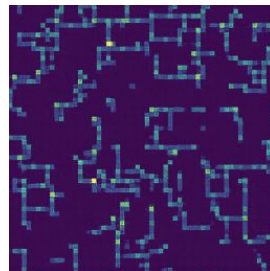


Real Congestion

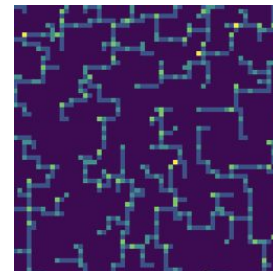
Epoch 100



Real Place



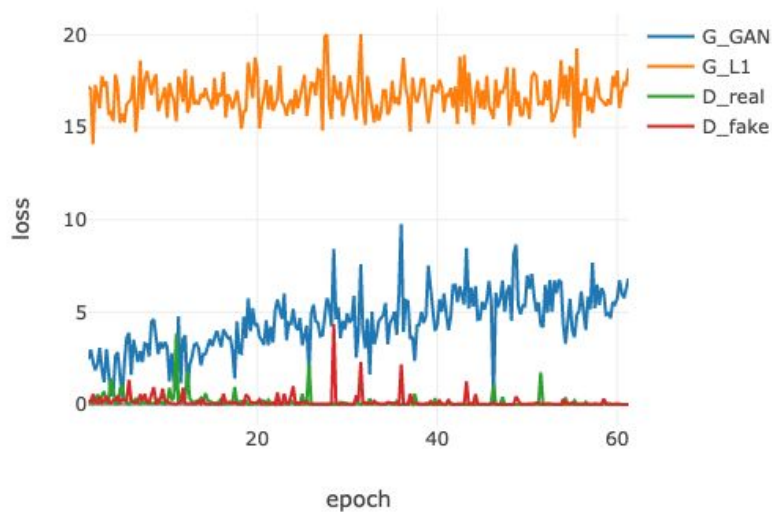
Fake Congestion



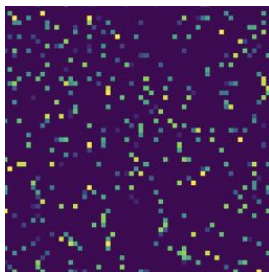
Real Congestion

Results

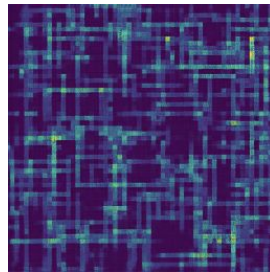
Pix2Pix Model for for 10 net design



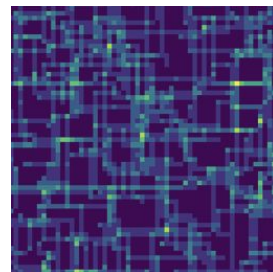
Epoch 5



Real Place

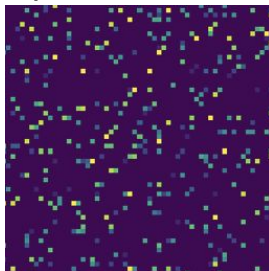


Fake Congestion

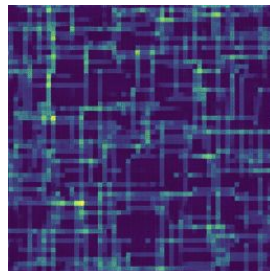


Real Congestion

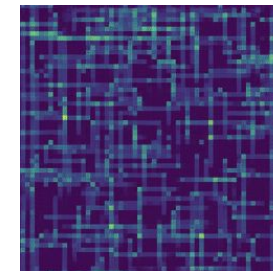
Epoch 50



Real Place



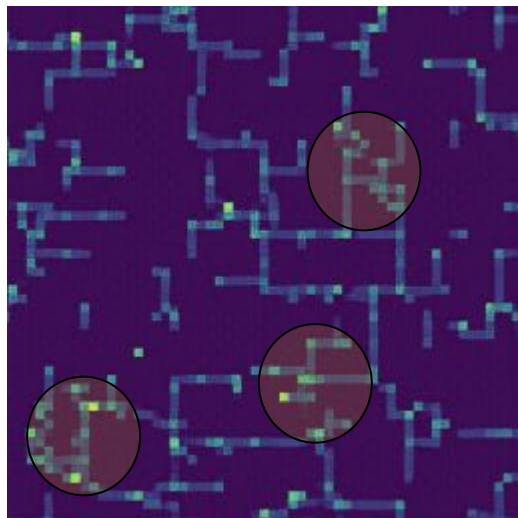
Fake Congestion



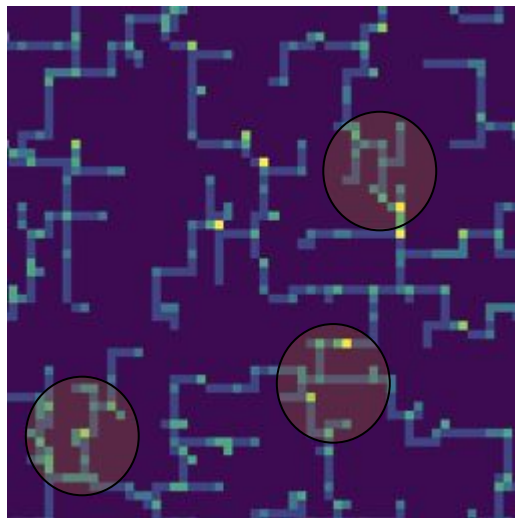
Real Congestion

Results

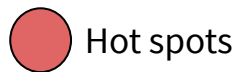
Pix2Pix Model for 1 net design with infoGAN data



Fake Congestion

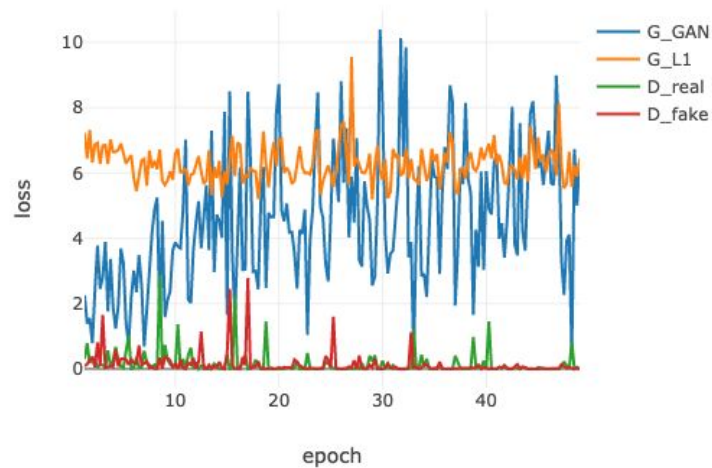


Real Congestion



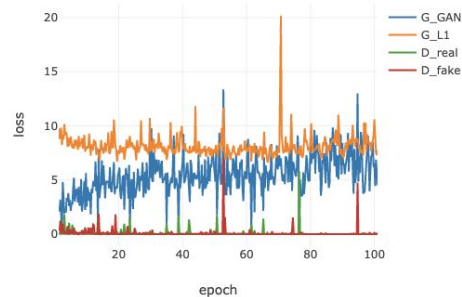
With InfoGAN Data

place_route_retrieve_pix2pix loss over time



With Random Data

place_route_1net_pix2pix loss over time



Conclusions

- A data generator for placement and route is developed imitating the real FPGA placement and routing problem
- The pix2pix routing congestion model is able to predict congested with high accuracy on a grid size of 64 with one net design, but still not able to generalize to multiple nets design
- InfoGAN is able to generate data for 1 net design, but not able to get trained on multiple nets design well.
- Based on InfoGAN generated placement data, the pix2pix model has around 20% lower l_1 loss

Thanks!

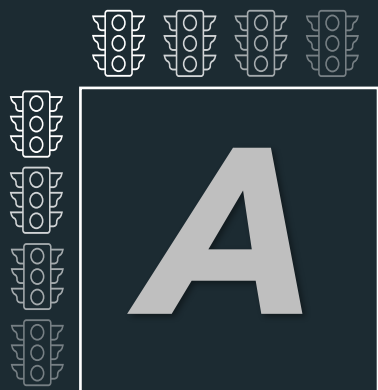
Spatial-Temporal Network with Adaptive Graph Structure

Yuchao Wu

yuchaow@andrew.cmu.edu



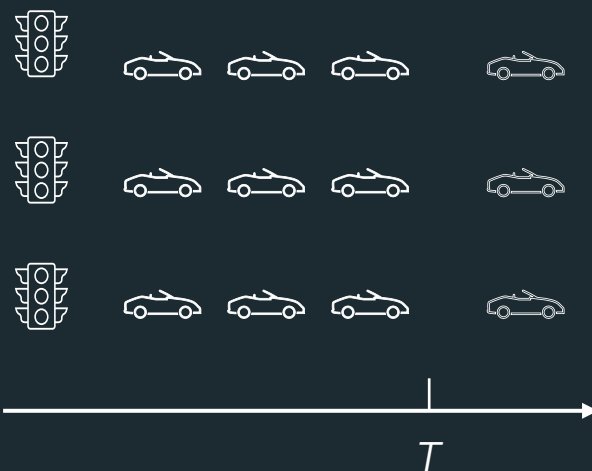
Spatial



Spatial Dependency

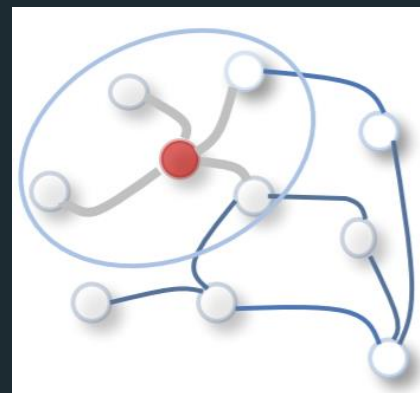


Temporal

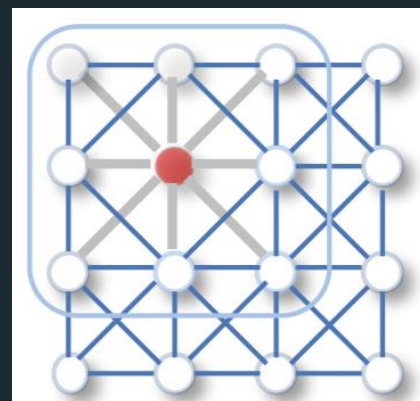


Temporal Dependency

Graph

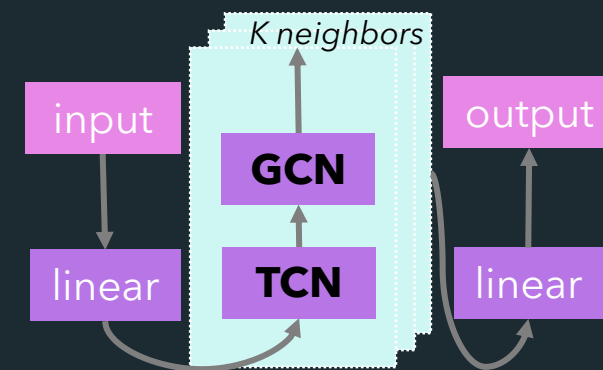


Graph convolution



2D convolution

Network



Temporal Convolution Block (TCN)
+
Graph Convolution Block (GCN)

Previous Methods

Proposed Method

Predefined Graph



Learned Graph

Static Graph



Dynamic Graph



Adaptive Adjacency Matrix

$$\mathbf{f}_{out} = \sum_{k=0}^K \mathbf{A}_{adp}^k \mathbf{f}_{in} \Theta_k$$

Graph convolution with K-step diffusion

$$\mathbf{A}_{adp}^{(t)} = \beta \tilde{\mathbf{A}}_{\Omega} + (1 - \beta) \mathbf{A}_{\mathcal{N}(K)}^{(t)}$$

Adaptive Graph = Global Graph + Local Graph

$$\tilde{\mathbf{A}}_{\Omega} = \text{SoftMax}(\text{ReLU}(\mathbf{A}_{\Omega}))$$

Global Graph captures long-term spatial relations

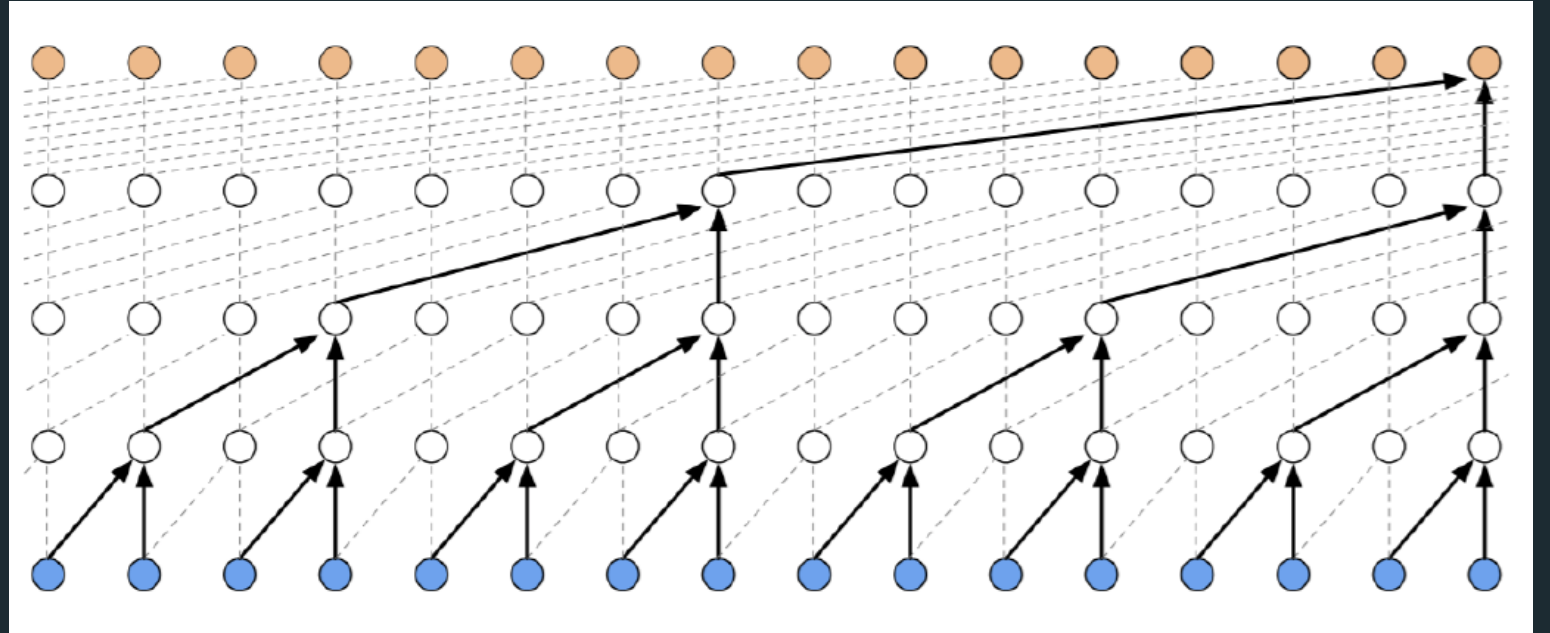
$$\mathbf{A}_{\mathcal{N}(K)}^{(t)} = \text{SoftMax}(\mathbf{X}^{(t-K:t)T} \mathbf{W}_{\theta}^T \mathbf{W}_{\phi} \mathbf{X}^{(t-K:t)})$$

Local Graph measures local node similarity in an embedding space

Gated Dilated Convolution

Stacked Dilated Convolution

+



Gated Linear Unit

$$\mathbf{h}_{k+1} = \tanh(\mathbf{W}_{f,k} * \mathbf{h}_k) \odot \sigma(\mathbf{W}_{g,k} * \mathbf{h}_k)$$

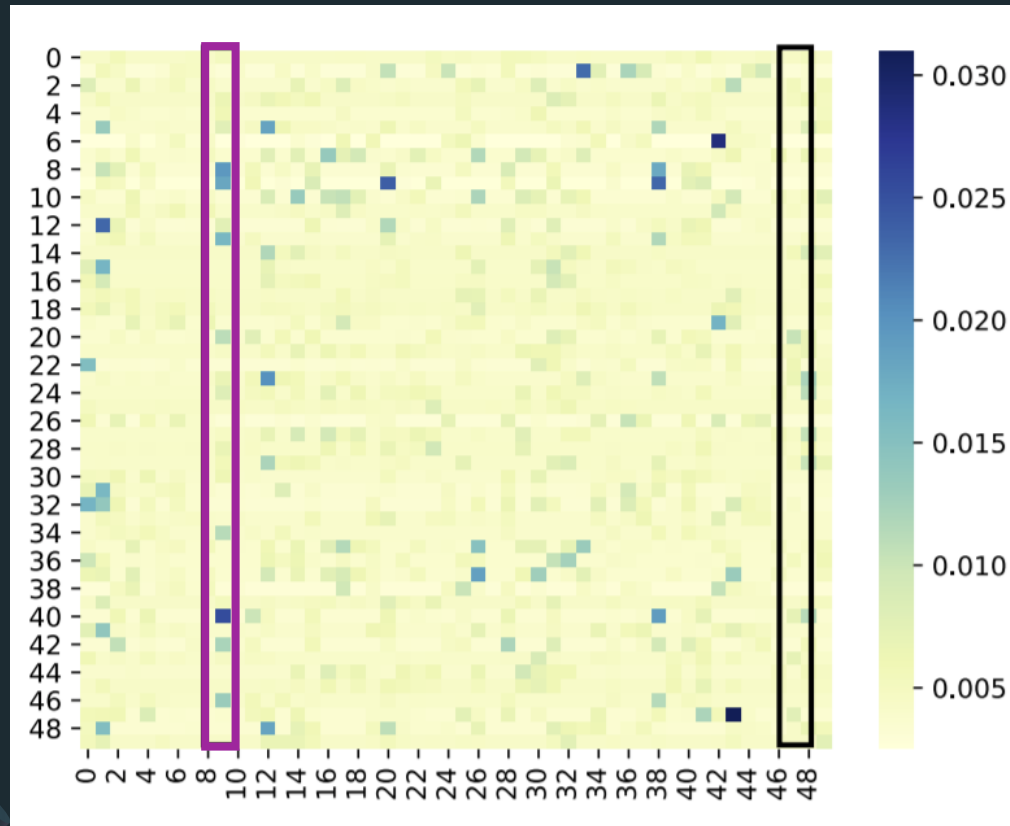
Experimental Results

METR-LR Dataset: Traffic Speed in Los Angeles

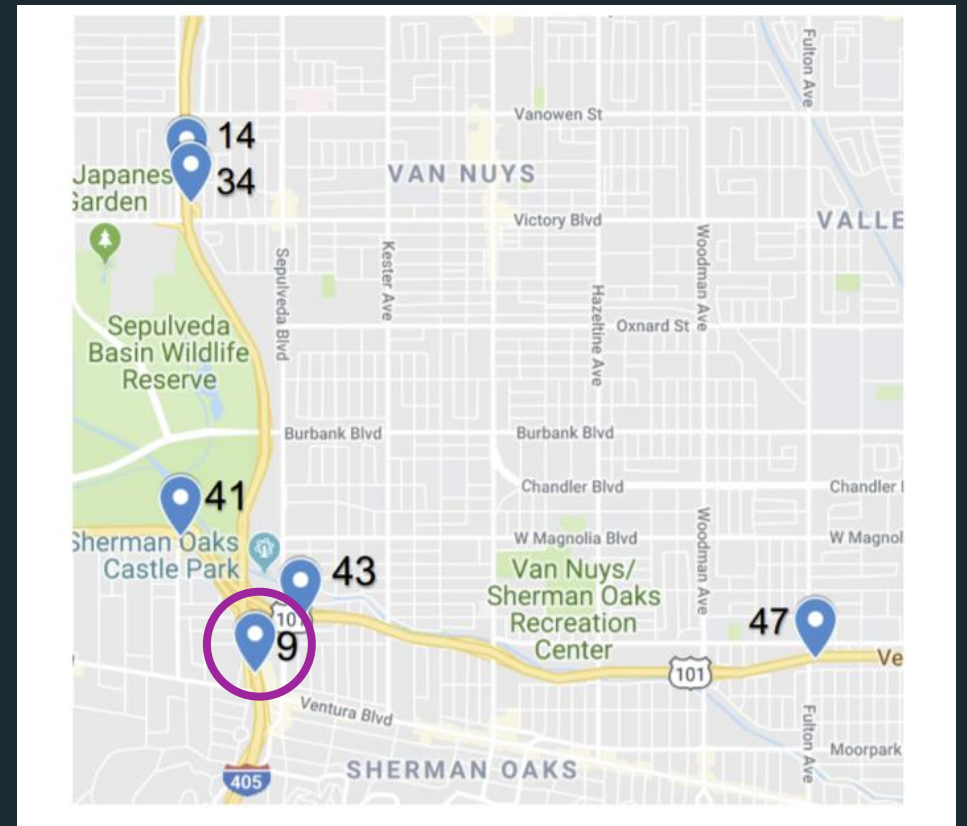
METHODS	ADJACENCY MATRIX	15 MIN RMSE
DCRNN	PRE-DEFINED	5.38
STGCN	IDENTITY	6.08
STGCN	PRE-DEFINED	5.24
STGCN	ADAPTIVE (RANDOM)	5.16
STGCN	ADAPTIVE (PREDEFINED)	5.15

Note: Adaptive (.) denotes Adaptive Adjacency Matrix with (.) initialization

Experimental Results



Learned Adaptive Matrix



Geographical Location



Unsupervised Conditional Generative Models

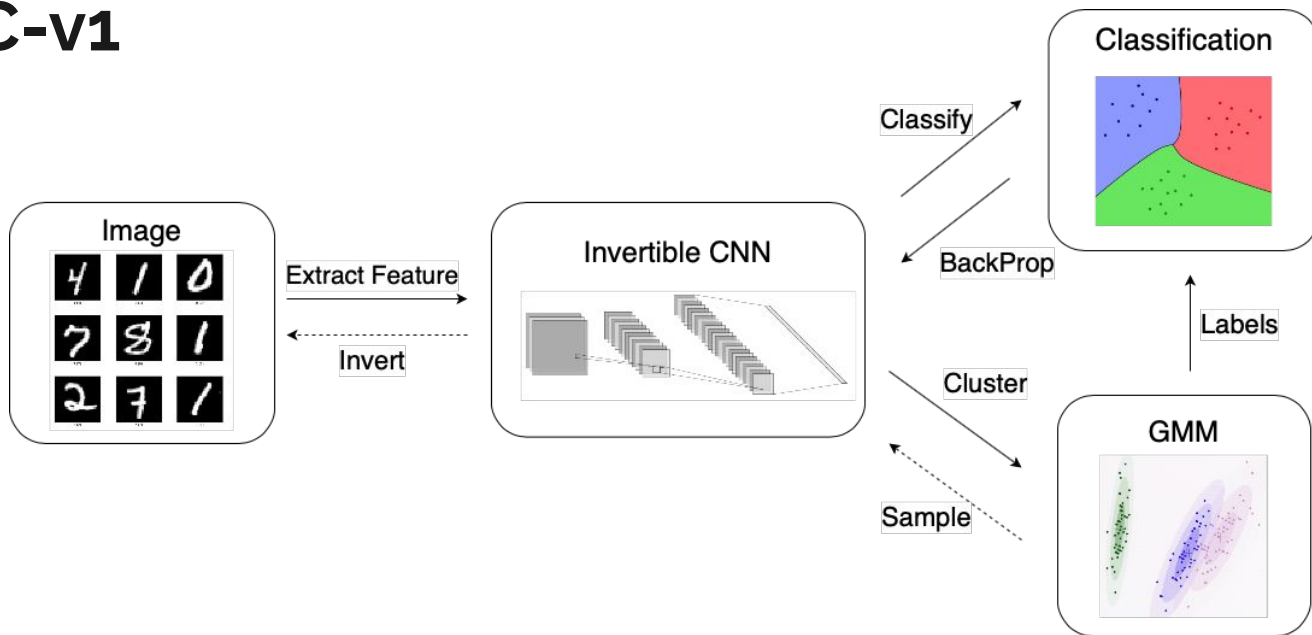
By Gregory Howe and Justin Jia



Introduction

1. Generative Models
 - a. Generative Adversarial Networks (GAN's)
 - b. Variational Auto Enconders (VAE's)
 - c. Flow Based
2. Deep Cluster
 - a. Unsupervised feature learning technique
 - b. Alternates between feature clustering and using the cluster assignments as labels to train a feature extractor
 - c. Consists of a feature extraction and a small classifier (typically a single dense layer)
3. i-RevNet
 - a. Fully Invertible CNN up to projection onto classes

i-DC-v1



Results v1

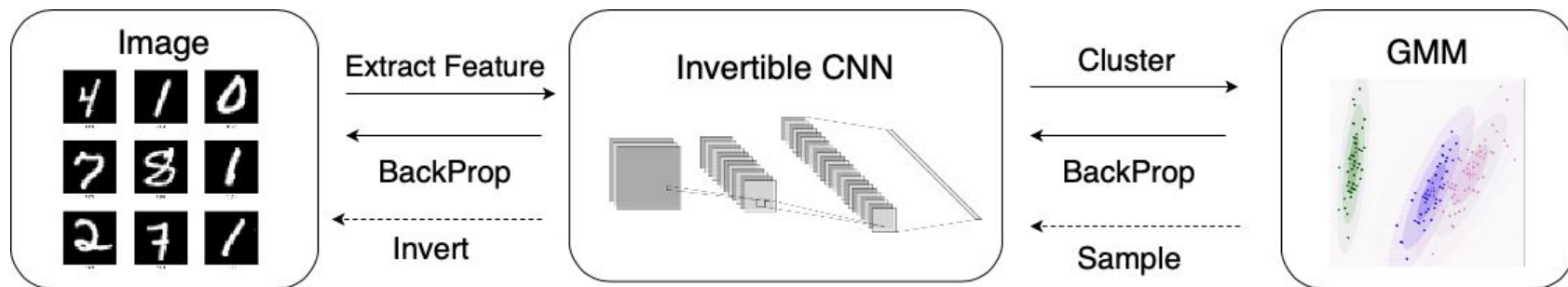


Cluster Means

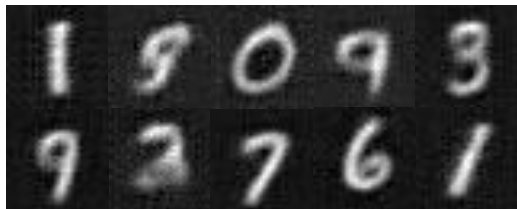


Samples

i-DC-v2



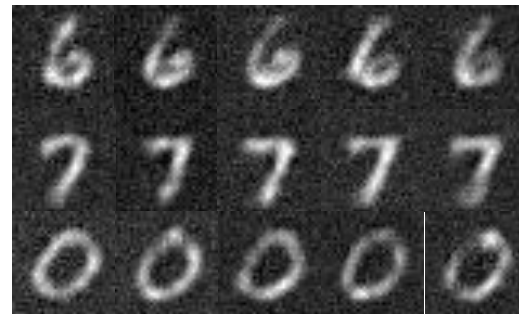
Results v2



Cluster Means



Sampled from GMM
(Variance Reduced)



Sampled from Individual Clusters
(Variance Reduced)



Summary and Future Work

- Introduced a generative model that can also sample from specific class distributions
- Sampling from specific class distributions provides model interpretability
- Would have liked to apply on image datasets with higher resolution



Thank You

Causal Direction and Unsupervised Transfer Learning

Minshi Peng

Department of Statistics and Data Science

F20-10708 project

Transfer learning (domain adaptation)

- Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.

T : target domain

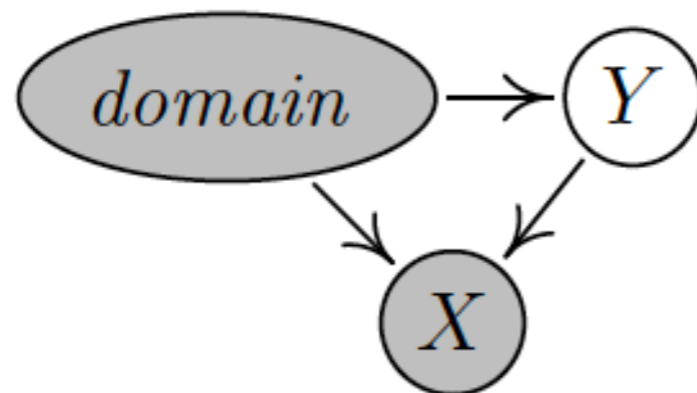
$(\mathcal{X}, \mathcal{Y}, P)$

S : source domain

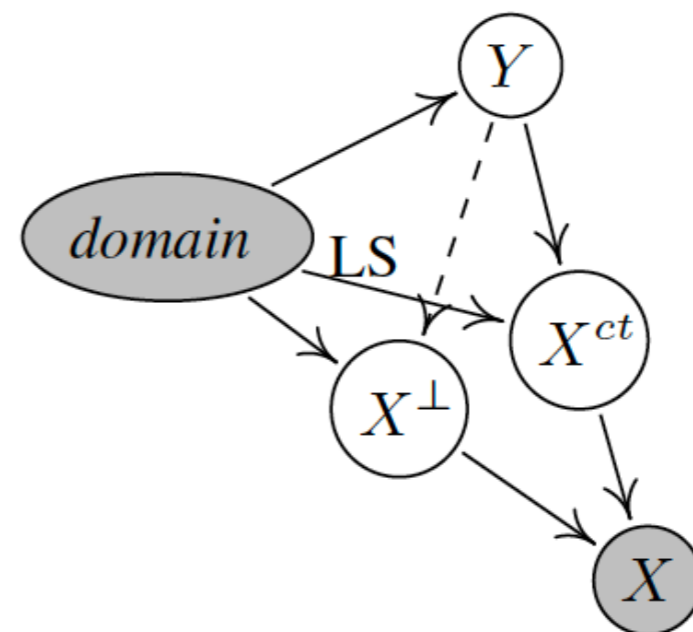
$(\mathcal{X}, \mathcal{Y}, Q)$

- Many existing works studying the theoretical properties of transfer learning under different model assumptions such as label shift, covariate shift, posterior shift etc (cite).

- My goal in this project is to study the transfer learning in unsupervised setting: how and when information can be passed from source to target to improve unsupervised learning
- Only feasible in anti-causal direction $Y \rightarrow X$ (Schölkopf et al., 2012)



(Zhang et al. 2013)



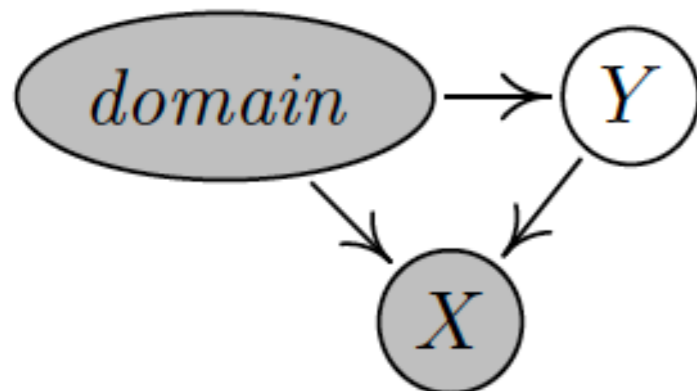
(Gong et al. 2016)

Goal:

- Examine the minimax bound of transfer learning on the unsupervised learning of target data under the assumed generative model.

S: source data
T: target data

$$P_{X,Y} = P_{X|Y}P_Y$$



Label shift

$$P_Y^S \neq P_Y^T$$

Conditional shift

$$P_{X|Y}^S \stackrel{LS}{\approx} P_{X|Y}^T$$

- Model: Gaussian mixture model with location-scale shift

Results

- Start with the simplest location-shift isotropic GMM model

$$\mathbf{X}_i^t = \eta_i^t \boldsymbol{\mu} + \varepsilon_i^t; \quad \mathbf{X}_i^s = \boldsymbol{\beta} + \eta_i^s \boldsymbol{\mu} + \varepsilon_i^s$$

- Study the minimax risk

$$\Psi_{\Delta} := \inf_{\tilde{\eta}^t, \tilde{\eta}^s, \hat{\boldsymbol{\beta}}} \sup_{(\mu, \eta) \in \Omega_{\Delta}} \frac{1}{n_t} \mathbb{E} \left[r(\tilde{\eta}^t, \eta^t) \right]$$

with $\Omega_{\Delta} = \pi_{\mu} \times \pi_{\eta^s} \times \pi_{\eta^t}$, where π_{η^s} and π_{η^t} are independent iid Rademacher random vectors. For the weak signal case,

suppose $\pi_{\mu} = \mathcal{N}(\mathbf{0}, \kappa_n^2 \mathbf{I}_p)$ for the strong signal case,

suppose π_{μ} is a point mass at $\mu_0 = (\Delta, 0, \dots, 0)$. (Ndaoud, 2018)

Results

- $n_s = \gamma n_t$, γ bound away from 0 and ∞ . Let

$$R_n = \frac{\Delta^2}{\sqrt{\Delta^2 + \frac{p}{n(1+\gamma)}}}$$

Theorem (lower bound). We have, with some $\epsilon_n = o(1)$, if $\Delta \geq \log^2 n / \sqrt{n}$

$$\Psi_\Delta \geq c\Phi^c \left(R_n (1 + \epsilon_n) \right)$$

$$\bar{\Delta}^2 = \sigma^2 \left(1 + \sqrt{1 + \frac{2p}{n \log n}} \right) \log n \quad \rightarrow \quad \bar{\Delta}^2 = \sigma^2 \left(1 + \sqrt{1 + \frac{2p}{(n_s + n_t) \log n_t}} \right) \log n_t$$

Results

- Examined the potential algorithm for transfer learning under the location shift model in order to derive upper bound
 - EM Algorithm
 - SDP: convex relaxation for learning GMM (Chen & Yang, 2020)
- Due to the non-convex nature, haven't figured out the upper bound yet.

Future extensions:

- Derive upper bounds through for alternative algorithms, potential through further convex relaxation .
- Study more general model, (i) with large label shift (distinction class composition, (ii) add the scale shift or consider other conditional shifts.

References:

1. Chen, X. and Yang, Y. Cutoff for exact recovery of gaussian mixture models. arXiv preprint arXiv:2001.01194, 2020.
2. Combes, R. T. d., Zhao, H., Wang, Y.-X., and Gordon, G. Domain adaptation with conditional distribution matching and generalized label shift. arXiv preprint arXiv:2003.04475, 2020.
3. Gong, M., Zhang, K., Liu, T., Tao, D., Glymour, C., and Schölkopf, B. Domain adaptation with conditional transferable components. In International conference on machine learning, pp. 2839–2848, 2016.
4. Ndaoud, M. Sharp optimal recovery in the two component gaussian mixture model. arXiv preprint arXiv:1812.08078, 2018.
5. Schölkopf, B., Janzing, D., Peters, J., Sgouritsa, E., Zhang, K., and Mooij, J. On causal and anticausal learning. arXiv preprint arXiv:1206.6471, 2012.
6. Zhang, K., Schölkopf, B., Muandet, K., and Wang, Z. Domain adaptation under target and conditional shift. In International Conference on Machine Learning, pp. 819–827, 2013.

A Causal (Re)-formulation of Controlled Text Synthesis

Amrith Setlur & Aman Madaan

Probabilistic Graphical Models Spotlight Presentation

Fall 2020

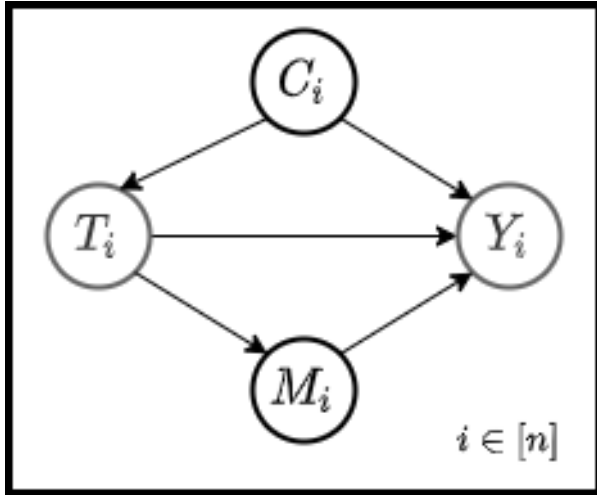
Controlled Text Synthesis is Counterfactual Inference

The pizza was terrible → The pizza was amazing

Send me the data → Please send me the data

- **Content words:** subject of the discussion
- **Style words:** alters the perception of the message (sentiment, level of politeness etc.)
- Controlled Text Synthesis is Counterfactual Inference!
 - **What** would the “The pizza was terrible” look like **if** it was written with a positive sentiment?

Causal Formulation



SCM

T: Presence/Absence of a Style word

C: Confounders (Topics, Content)

Y: Perceived Style (Sentiment)

M: Mediators (indirect effect)

$$\mathbf{x}_i^{s_i} \sim \mathbb{P}(X)$$

Observed Text

$$\mathbf{x} \sim \mathbb{P}_C^{\text{do}(S=s')}(X'|X = \mathbf{x}_i, S = s_i)$$

Sample from
Counterfactual
Distribution

$$\mathbb{P}_C^{\text{do}(S=s')}(X'|X = \mathbf{x}_i, S = s_i)$$

$$= \int_Z d\mathbb{P}_C^{\text{do}(S=s')}(X', Z|S = s_i, X = \mathbf{x}_i)$$

Intractable!

$$= \int_Z \mathbb{P}_C(X'|Z, S = s') d\mathbb{P}_C(Z|S = s_i, X = \mathbf{x}_i)$$

(1) Action (2) Abduction (3) Prediction

Tractable Approximation

- Step 1: **Identify** causally relevant words in sentence **and remove** them
- Step 2: **Generate** contextually relevant sentences by replacing the gaps with words/phrases associated with the new (**intervened**) style.


Good approximation **only when** the words identified to be removed
have a true causal effect on the inferred style!


Current frameworks identify words to be of a particular style merely through **estimators** that capture **correlation**, not **causation**!

Estimating Average Treatment Effect (ATE) (for Style Words)

$$\begin{aligned}\mathbf{ATE: } \psi_{ate} &= \mathbb{E}^{\text{do}(T=1)} [Y] - \mathbb{E}^{\text{do}(T=0)} [Y] \\ &= \mathbb{E}_C [\mathbb{E} [Y | T = 1, C] - \mathbb{E} [Y | T = 0, C]]\end{aligned}$$

$$\hat{\psi}_{pg} = \frac{1}{\sum_i t_i} \sum_i \boxed{g(c_i)} \left(\hat{Q}(1, c_i) - \boxed{\hat{Q}(0, c_i)} \right)$$

 PROPENSITY

 CONDITIONAL
EXPECTATION

Unbiased
ATE
Estimator

Causal Sufficiency

Veitch et. al. (AISTATS 2020)

Theorem 3.1. *Suppose $\lambda(\mathbf{w})$ is some function of the words such that at least one of the following is $\lambda(\mathbf{W})$ -measurable:*

1. $(Q(1, \mathbf{W}), Q(0, \mathbf{W}))$,
2. $g(\mathbf{W})$,
3. $g((Q(1, \mathbf{W}), Q(0, \mathbf{W})))$ *or*
 $(Q(1, g(\mathbf{W})), Q(0, g(\mathbf{W})))$.

If adjusting for \mathbf{W} suffices to render the ATT identifiable then adjusting for only $\lambda(\mathbf{W})$ also suffices. That is, $\psi = \mathbb{E}[\mathbb{E}[Y \mid \lambda(\mathbf{W}), T = 1] - \mathbb{E}[Y \mid \lambda(\mathbf{W}), T = 0]]$.

- For ATE, we need a **consistent estimator** for the **propensity score** and **conditional expectations** of the outcome (Y) given the treatment (T).
- But text data is extremely **high-dimensional** which makes the estimation hard.
- For this, we use the notion of **causal sufficiency** where we need only learn a function of the set of words comprising the confounder (C), that is sufficient to reliably predict $g(\cdot)$, $Q(1, \cdot)$, $Q(0, \cdot)$.

Learning Causally Sufficient representations

- In practice: add losses to predict the outcome Y and the treatment T , in addition to the language modeling loss

$$\begin{aligned} L(\mathbf{w}_i; \xi, \gamma) = & (y_i - \tilde{Q}(t_i, \lambda_i; \gamma))^2 \\ & + \text{CrossEnt}(t_i, \tilde{g}(\lambda_i; \gamma)) \\ & + L_U(\mathbf{w}_i; \xi, \gamma). \end{aligned}$$

- Treatment \mathbf{T} (collection of Binary RVs):
 - 1 if the sentence contains a word $w \in T_w$, 0 otherwise
 - We experiment with eight sets of T_w (food words, descriptive words)
- Outcome \mathbf{Y} :
 - The target style

Experiments

- 50,000 samples from the Yelp restaurant corpus for sentiment classification
- Eight sets of treatment words T_w : (top/barely/all) positive, (top/barely/all) negative, food words, and descriptive words.
- Calculate the ATE for each set T_w

Algorithm 1 ATE estimation for style words and phrases.

Given: A set of treatment words \mathbf{w}

Given: A corpus \mathcal{D} of n sentences, each labeled with an outcome Y (style)

Result: $\hat{\psi}_q(\mathbf{w})$, the ATE estimate of w on the target Y

```
/* Set treatment */
1 for sentence  $s_i \in \mathcal{D}$  do
2   if  $\mathbf{w}$  in  $s_i$  then
3     /* if any of the treatment phrases
4       is present in the sentence */
5      $T_i = 1$ 
6   else
7      $T_i = 0$ 
8   end
9 end
/* Train BERT with MLM, as well as
losses for  $\mathbf{T}$  and  $\mathbf{Y}$  using Equation 6
*/
8  $\gamma = \text{finetune}(\mathcal{D}, \mathcal{L}(\gamma))$ 
/* Get the estimator */
9  $\hat{\psi}_q(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n Q(t_i = 1, \lambda(s_i); \gamma) - Q(t_i = 0, \lambda(s_i); \gamma)$ 
10 return  $\hat{\psi}_q(\mathbf{w})$ 
```

Results

Treatment (\mathbf{w})	$\hat{\psi}_q(\mathbf{w})$
Positive	0.06
Negative	-0.07
Top Positive	0.40
Top Negative	-0.59
Barely Positive	0.25
Barely Negative	-0.32
Food Words	0.03
Descriptive Words	-0.02

Table 1. $\hat{\psi}_q(\mathbf{w})$ for different settings of \mathbf{W}

- Our method correctly associates low ATE with **descriptive words** like "cold", "warm", "edible" and **food words** "pizza", "sushi", "coffee" are incorrectly labeled as style words by existing systems.
- **Causally sufficient** embeddings can **tease out** the causal impact of each word, allowing us to infer the style-independent endogenous variables (namely, the content words) more effectively.
- Additional benefit: can **quantify** the causal contribution of each word to the style!

Thank you!

Human Trajectory Prediction with Social Interaction Transformer

Yangfan Liang & Xinyu Yao

Carnegie Mellon University

December 9, 2020

Background and Motivation

- ① Booming location-based services
- ② Accurate trajectory prediction will lead to
 - better navigation service
 - better autonomous driving
- ③ **Big Challenges:** how to correlate the spatial, temporal, semantic, and content-based information (Wu et al., 2018).
- ④ **Our Solution: Social Interaction Transformer for Trajectory Prediction**
- ⑤ Reasons:
 - **Transformer models:** excellent representation of the spatiotemporal trajectory data to mining heterogeneous human behavior patterns.
 - **Social mechanism:** consider the interaction between the agents in the system.

Baseline Model: Vanilla LSTM (Alahi et al., 2016)

- ① LSTM proven successful for sequential learning
- ② Vanilla LSTM for trajectory prediction learns **the state of the person** and predicts their future positions
- ③ Big Issue: a vanilla LSTM directly modeling human trajectory would **fail to take human interaction into account.**

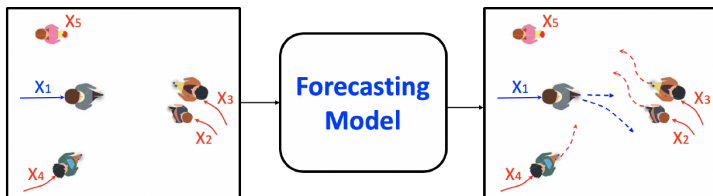


Figure: Trajectory Prediction Task (Kothari et al., 2020)

Baseline Model: Social LSTM (Alahi et al., 2016)

- ① Each individual time varying motion-properties captured by hidden states of LSTM
- ② Vanilla LSTM does not model interactions between LSTMs
- ③ Modification: Add a **social pooling layer** on those hidden states of LSTMs to incorporate interactions between different trajectories that are proximal in space; add pooled social hidden-state tensor as input for trajectory encoder LSTM

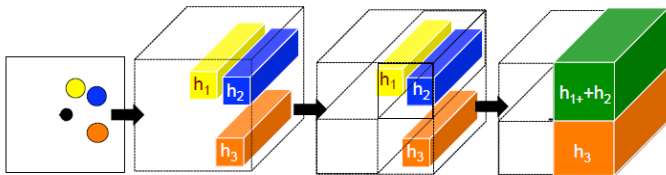


Figure: Social LSTM (Alahi et al., 2016)

Proposed method: Social Transformer

- ① Motivation:
 - The similarity between trajectory prediction and text generation
 - LSTM models has been considered less potent than the Transformer model for NLP problems (Vaswani et al., 2017).
 - Recent work about Transformer for trajectory prediction did not consider social interactions (Giuliani et al., 2020)
- ② Framework: the trajectory encoder module, the interaction module, the decoder module.
- ③ Fill the gap in those previous works by applying **Transformer for encoder while still considering social interaction.**
- ④ Main changes: Compared with Social-LSTM, replace LSTM with transformer for the trajectory encoder module

Proposed method: Social Transformer Formulation

- 1 Input: all observed motion trajectories: $M^o = [M_1^o, M_2^o, \dots, M_n^o]$
- 2 Output: the prediction of the remaining trajectories for each individual agent: $M^r = [M_1^r, M_2^r, \dots, M_n^r]$.
- 3 The observed position of agent i at time frame t : $m_i^t = (x_i^t, y_i^t)$, where $t = 1, \dots, T_{obs}$ and T_{obs} is the last received time frame point.
- 4 v_i^t and a_i^t : the velocity and acceleration at frame t of the agent i .
- 5 State of each agent at frame t : $s_i^t = [m_i^t, v_i^t, a_i^t]$
- 6 Directional pooling tensor D_i^t

$$D_i^t(m, n, :) = \sum_{j \in N_i} \mathbb{I}_{mn}[x_j^t - x_i^t, y_j^t - y_i^t](v_i^t - v_j^t) \quad (1)$$

- 7 The interaction vector $p_i^t = \phi_p(D_i^t; W_p)$ where ϕ_p is an MLP and W_p is the weight to learn.

Proposed method: Social Transformer Formulation

- ① The state embedding e_i^t :

$$e_i^t = \phi_{emb}(s_i^t; W_{emb}) \quad (2)$$

- ② Encoder output:

$$U_i = \text{TF}_{enc}([e_i, p_i]; W_{enc}) \quad (3)$$

- ③ Decoder output at each time stamp:

$$[\mu_i^t, \sigma_i^t, \rho_i^t] = \phi_{dec}(U_i; W_{dec}) \quad (4)$$

- ④ All the weights W_{emb} , W_{enc} , W_{dec} are to be learned.

- ⑤ Loss function: the negative log-likelihood function of the positions, velocity and acceleration on the training set for the i -th trajectory):

$$\mathcal{L}_i(W) = - \sum_{t=T_{obs}+1}^{T_{pred}} \log(\mathbb{P}(m_i^t, v_i^t, a_i^t | \mu_i^t, \sigma_i^t, \rho_i^t)) \quad (5)$$

We use TrajNet dataset.

- Goal: Predict the trajectory of the following 12 frames, given the observation of 8 consecutive frames
- Consists four different datasets:
 - BIWI Hotel (contains 389 pedestrians, bird's eye view, outdoor at building entrance)
 - Crowd UCY (contains three datasets of total 204 pedestrians, bird's eye view in various outdoor situations, from shopping streets to student pedestrians)
 - MOT PETS 2009 (19 pedestrians, multi-sensor sequences containing different crowd activities, outdoors on a path)
 - Stanford Drone Dataset (contains eight scenes of total 3297 pedestrians, bird's eye's view).

Table: TrajNet Results

Model name	Avg	ADE	FDE
Vanilla LSTM	2.107	3.114	1.100
Social LSTM	1.3865	2.098	0.675
Transformer	0.776	1.197	0.356

- Average Displacement Error (ADE) represents the mean squared loss between model prediction and ground truth on predicted points.
- Final Displacement Error (FDE) represents the distance between the final ground-truth destination and model predicted the final destination at the end of the prediction period.

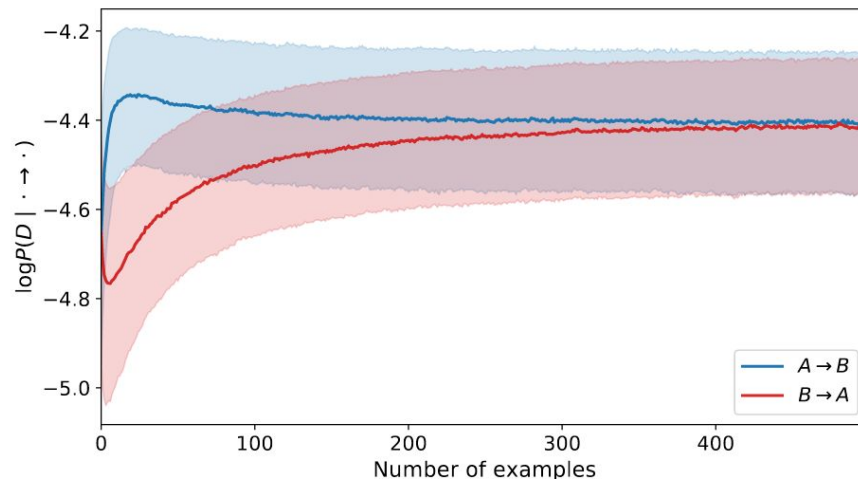
- Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., and Savarese, S. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 961–971, 2016.
- De Brébisson, A., Simon, É., Auvolet, A., Vincent, P., and Bengio, Y. Artificial neural networks applied to taxi destination prediction. *arXiv preprint arXiv:1508.00021*, 2015.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- Giuliani, F., Hasan, I., Cristani, M., and Galasso, F. Transformer networks for trajectory forecasting. *arXiv preprint arXiv:2003.08111*, 2020.
- Kothari, P., Kreiss, S., and Alahi, A. Human trajectory forecasting in crowds: A deep learning perspective. *arXiv preprint arXiv:2007.03639*, 2020.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.
- Walters, R., Li, J., and Yu, R. Trajectory prediction using equivariant continuous convolution. *arXiv preprint arXiv:2010.11344*, 2020.
- Wu, R., Luo, G., Shao, J., Tian, L., and Peng, C. Location prediction on trajectory data: A review. *Big Data Mining and Analytics*, 1(2):108–127, 2018.
- Ye, N., Zhang, Y., Wang, R., and Malekian, R. Vehicle trajectory prediction based on hidden markov model. *KSII Trans. Internet Inf. Syst.*, 10:3150–3170, 2016.

Identifying Causal Structure

10708 Project Presentation
Wael Al Saeed, Advait Gadhikar

Meta Transfer Objective

- The true causal form adapts faster to interventional changes in data
- Correct causal form needs to change fewer parameters to learn the transfer (intervened distribution)
- A Parameter counting proof for 2 nodes discrete case



- Meta Transfer Loss:
- $\mathcal{R} = -\log [\text{sigmoid}(\gamma)\mathcal{L}_{A \rightarrow B} + (1 - \text{sigmoid}(\gamma))\mathcal{L}_{B \rightarrow A}]$

where $\mathcal{L}_{A \rightarrow B} = \prod_{t=1}^T P_{A \rightarrow B}(a_t, b_t; \theta_t)$ and $\mathcal{L}_{B \rightarrow A} = \prod_{t=1}^T P_{B \rightarrow A}(a_t, b_t; \theta_t)$.

Meta Transfer Objective for more than two nodes

- Graphs with more than 2 nodes can be parametrized as a soft adjacency matrix
- Ground truth graph is a causal DAG, but the parametrization does not assume so
- Can be easily included in the meta transfer learning objective framework

Each edge in the graph is B_{ij} such that

$$B_{ij} \sim \text{Bernoulli}(p_{ij}) \text{ where } p_{ij} = (\gamma_{ij})$$

V_j is a parent of V_i if $B_{ij} = 1$

The set of parents of a node is given by

$$pa(i, V, B_i) = \{V_j | B_{ij} = 1, j \neq 1\}$$

$$\mathcal{L}_{B_i} = \prod_t P_{B_i}(V_i = v_{ti} | pa(i, v_t, B_i))$$

Now the total regret can be given as

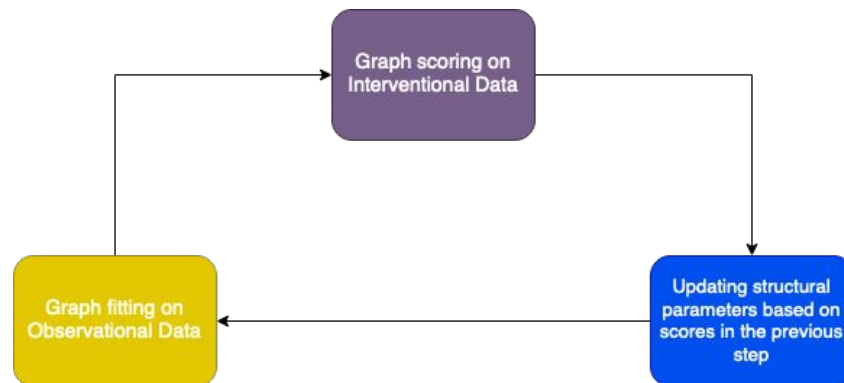
$$\mathcal{R} = -\log E_B \{\mathcal{L}_B\} \text{ where } \mathcal{L}_B = \prod_i \mathcal{L}_{B_i}$$

And its gradient can be estimated as

$$g_{ij} = \frac{\sum_k (\sigma(\gamma_{ij}) - B_{ij}^{(k)}) \mathcal{L}_{B_i}^{(k)}}{\sum_k \mathcal{L}_{B_i}^{(k)}}$$

Neural Causal Learning with Unknown Interventions

- Score based method to learn the causal structure in data
- Alternately learns structural parameters of the underlying graph and the functional parameters of the SCM
 - Learn parameters of a neural network, that model the functional parameters, on observational data
 - Calculate the log likelihood of this model on interventional data
 - Predict intervened variable as one with least log likelihood
 - Update structural (graph) parameters based on intervened log likelihood
 - Enforce Sparsity and Smooth Acyclicity constraint

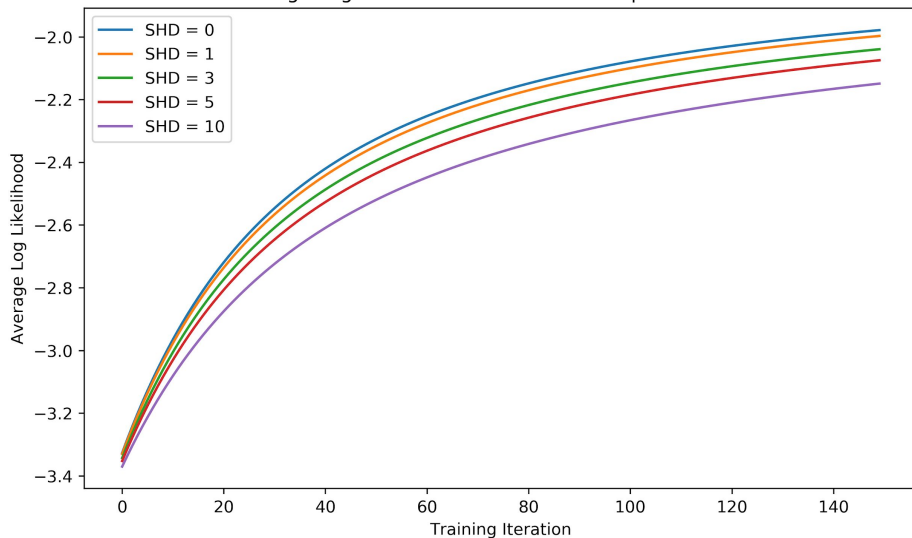


Gradient for Structural Parameters

$$g_{ij} = \frac{\sum_k (\sigma(\gamma_{ij}) - c_{ij}^{(k)}) \mathcal{L}_{C,i}^{(k)}(X)}{\sum_k \mathcal{L}_{C,i}^{(k)}(X)}, \quad \forall i, j \in \{0, \dots, M-1\}$$

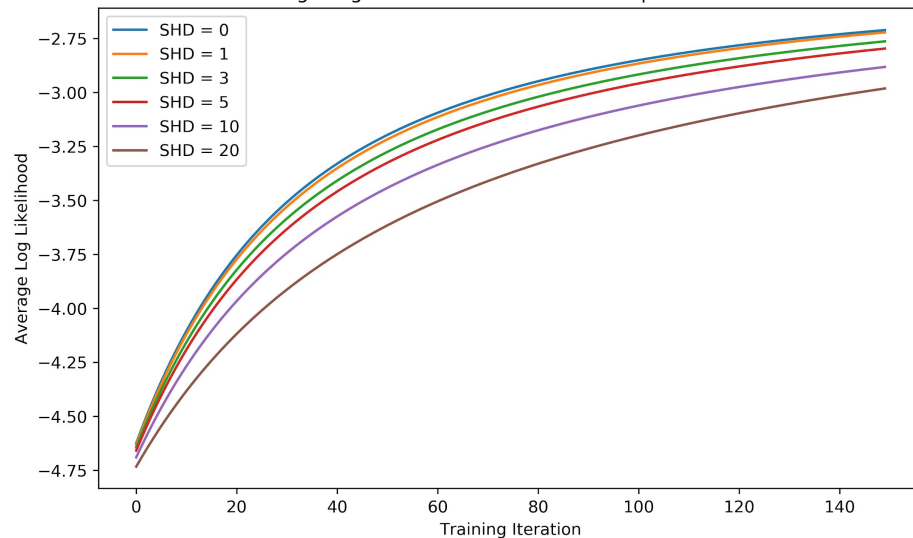
Analysis of the Meta-Transfer Loss

Average Log Likelihood of Transfer Data per SHD value



Ground Truth: Chain5

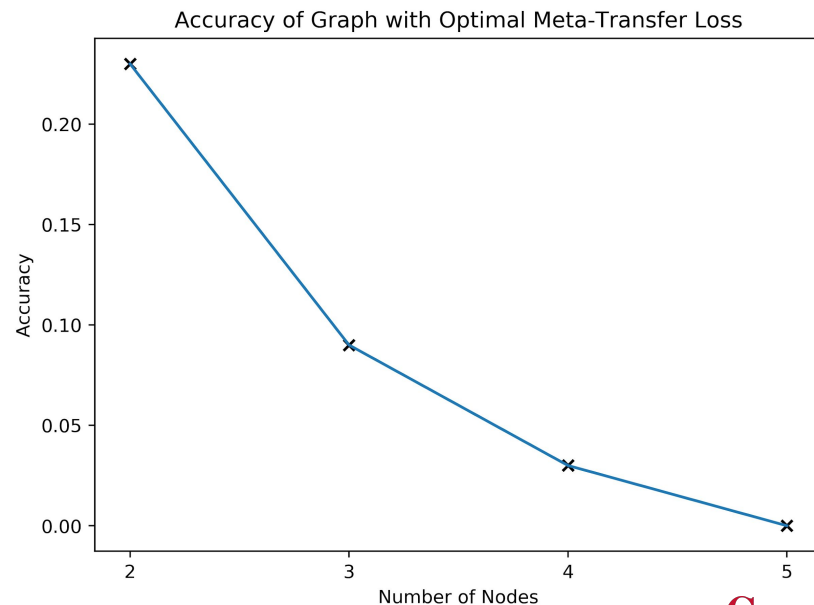
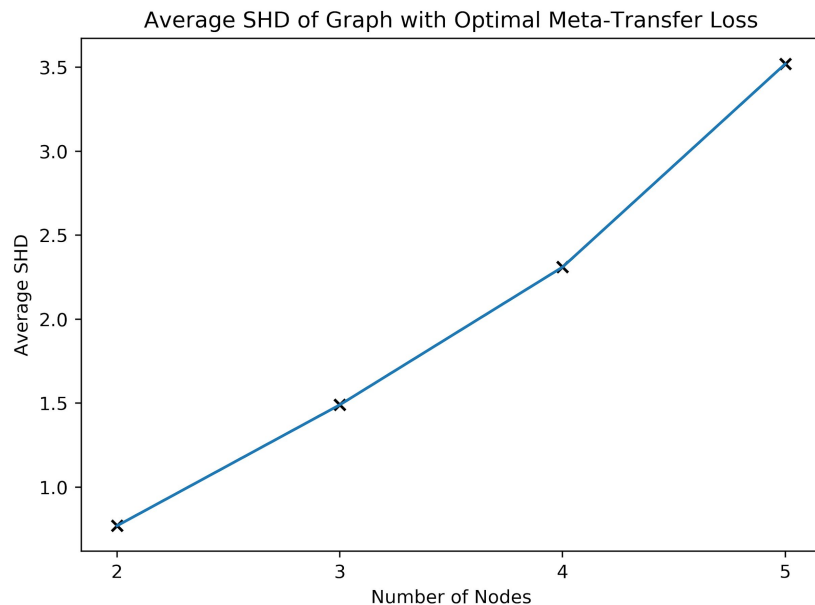
Average Log Likelihood of Transfer Data per SHD value



Ground Truth: 3 Layers Full Binary Tree

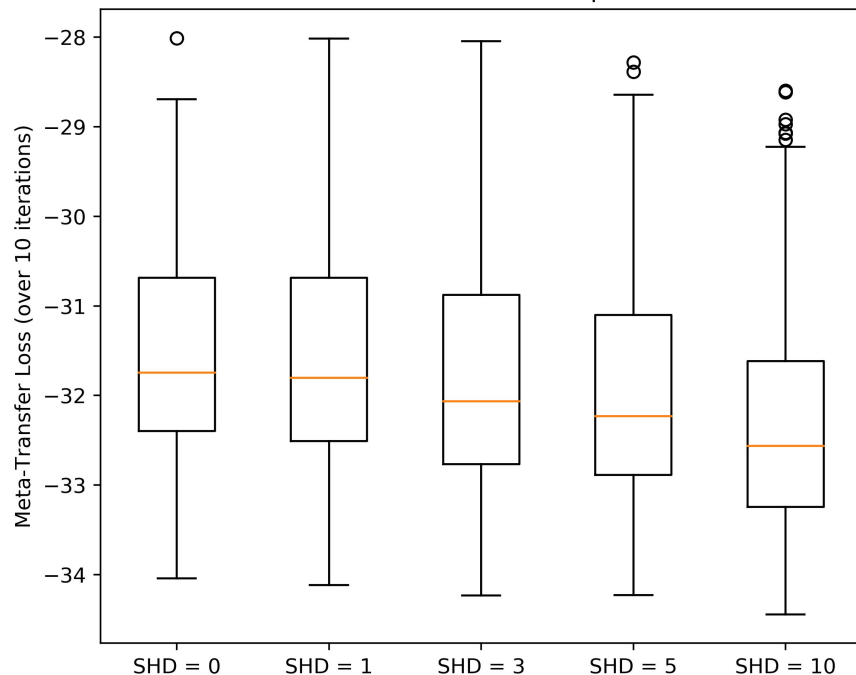
Analysis of the Meta-Transfer Loss

Performance of the Graph with Optimal Meta-Transfer Loss for Ground Truth Chain2-5



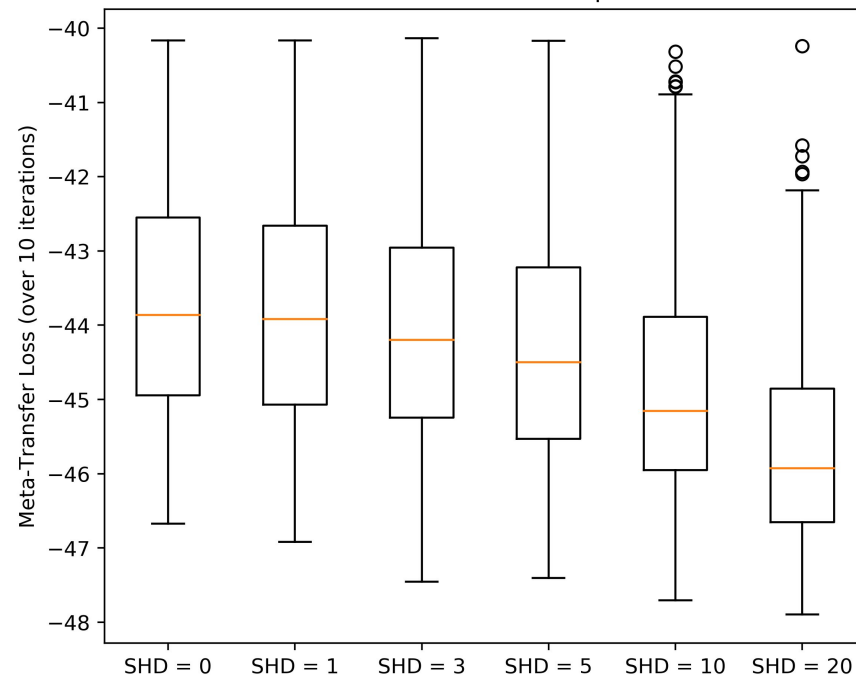
Analysis of the Meta-Transfer Loss

Meta-Transfer Loss Distribution per SHD value



Ground Truth: Chain5

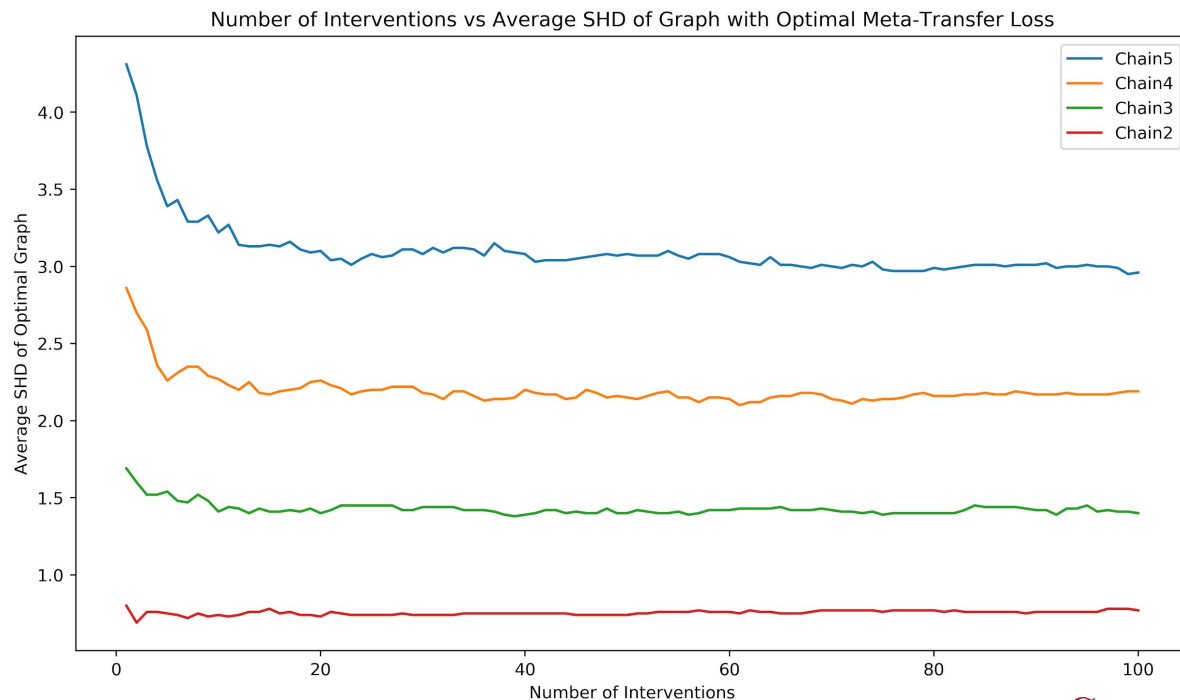
Meta-Transfer Loss Distribution per SHD value



Ground Truth: 3 Layers Full Binary Tree

Analysis of the Meta-Transfer Loss

How Number of Interventions affects performance of Meta-Transfer Loss



References

- Bengio, Yoshua, et al. "A meta-transfer objective for learning to disentangle causal mechanisms." *arXiv preprint arXiv:1901.10912* (2019).
- Ke, Nan Rosemary, et al. "Learning neural causal models from unknown interventions." *arXiv preprint arXiv:1910.01075* (2019).

Unsupervised Causal Cluster Inference with Domain Transfer

Tyler Lovelace, Daniel Yuan

Human Diseases: Cancer

Many major types of cancer, categorized by starting cell type

→ each cancer has their own further subtypes (molecular, genetic, etc)

→ subtypes linked to different survival rates and treatment plans

Main Questions:

- How can we identify these subtypes from data? **Cluster Inference**
- How are these subtypes linked to gene interaction networks? **Causality**
- How do we apply these models on a wider scale? **Domain Transfer**

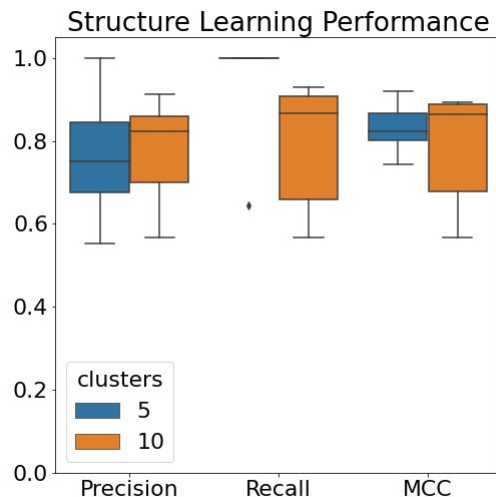
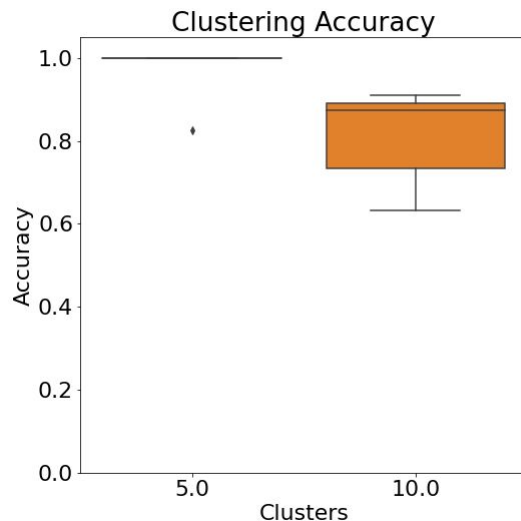
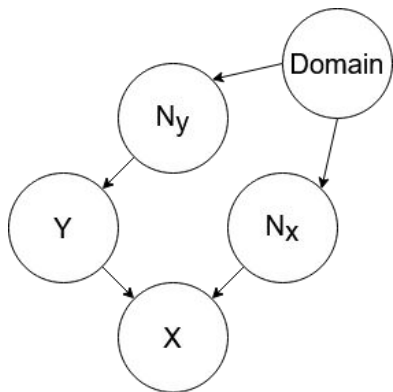
Our strategy:

Link clusters directly to causal networks and domain noise

Component 1: Causal Cluster Inference

Learn directed acyclic graph (DAG)
over the observed features \mathbf{X} to
identify cluster membership in \mathbf{Y} .

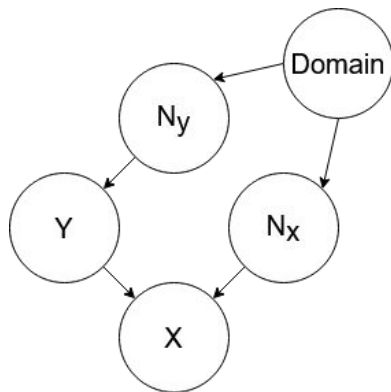
Each cluster in \mathbf{Y} is represented as
an indicator variable that is a
casual parent of features in \mathbf{X} .



Component 2: Domain Transfer

DAG is learned under assumptions of causality and Markov faithfulness

Utilize the assumption of **independent causal mechanisms** to adapt our clusters to the different exogenous noise in new domains.



Learn linear causal model over X

$$x_s = \alpha_s + \sum_{t \in PA_s} \beta_{st} x_t + \mathcal{N}(0, \sigma_s)$$



Learn causal effects of Y on X

$$x_s = \alpha_s + \sum_{k=1}^K \gamma_{ks} y_k + \sum_{t \in PA_s} \beta_{st} x_t + \mathcal{N}(0, \sigma_s)$$

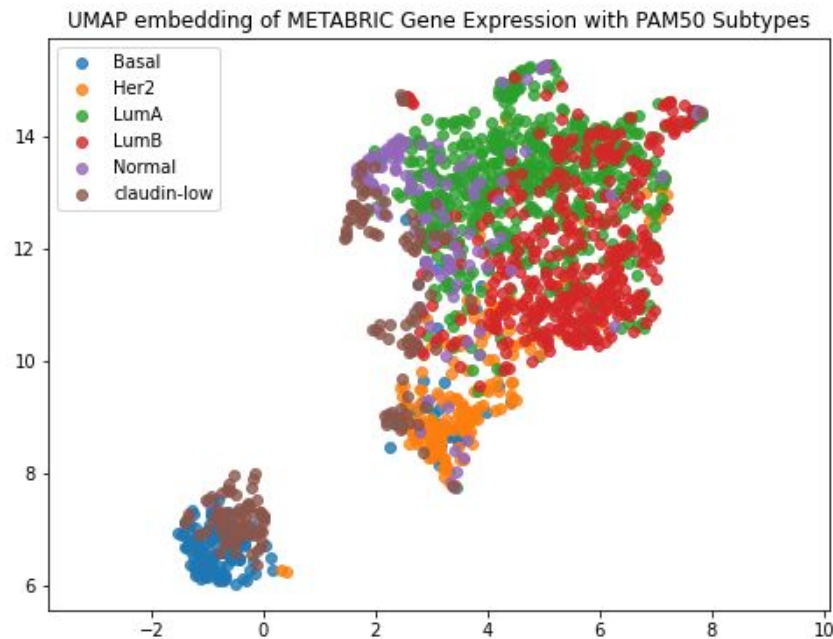
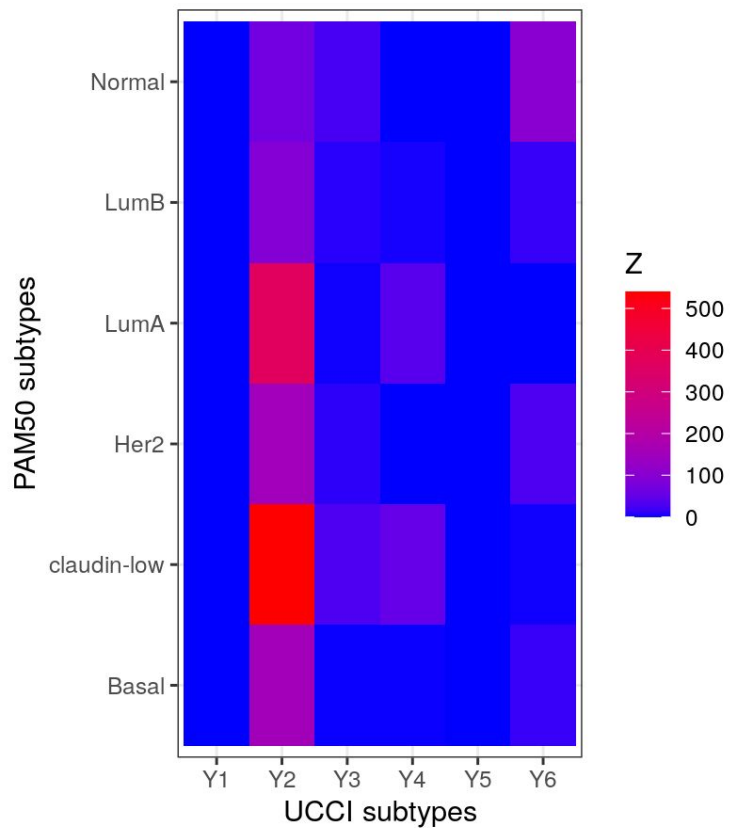


Estimate domain specific parameters, σ and α

$$r_s = \sum_{k=1}^K \gamma_{ks} y_k = x_s - \left(\alpha_s + \sum_{t \in PA_s} \beta_{st} x_t + \mathcal{N}(0, \sigma_s) \right)$$

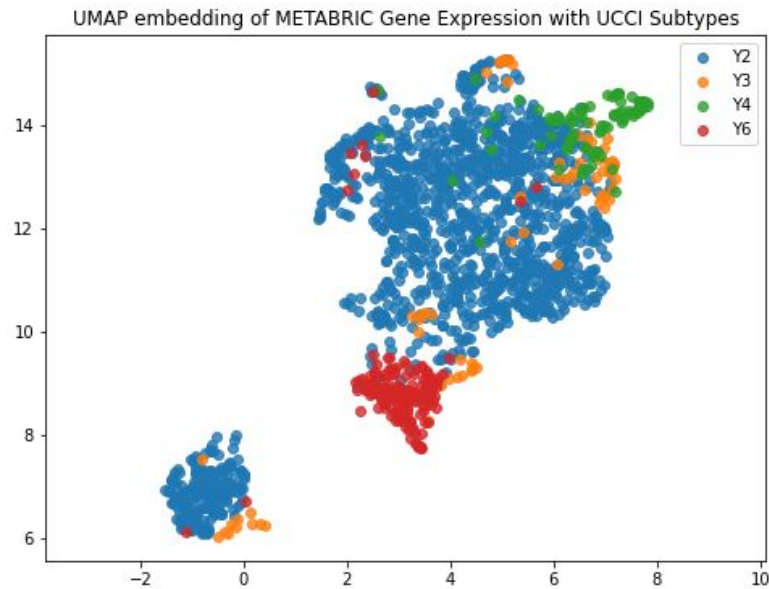
$$r_s = \alpha_s + \sum_{k=1}^K \gamma_{ks} y_k = x_s - \left(\sum_{t \in PA_s} \beta_{st} x_t + \mathcal{N}(0, \sigma_s) \right)$$

METABRIC Clustering Results



Interpreting UCCI METABRIC Clusters

- UCCI is identifying latent cluster features that cause tumors to deviate from the underlying causal model
 - Different from clustering directly on the feature space
- Y2 cluster is baseline
 - Tumors fit the learned causal model
- Other clusters cause distinct deviations from baseline, and are enriched for genes related to
 - Y3: metabolism, cytokine response, cell division
 - Y4: cellular regulation, cell communication, immune system
 - Y6: immune system, negative cellular regulation



Conclusion

- UCCI clusters mostly did not align well with known breast cancer subtypes
 - Likely due to difference in assumptions:
 - Traditional breast cancer subtypes are based on similarity in gene expression
 - UCCI breast cancer subtypes are based on deviation from underlying causal model of gene expression
- UCCI clusters are still biologically meaningful
 - Clusters are causally linked to genes involved in cellular processes linked to cancer
- Domain transfer of clusters to external datasets needs to be analyzed



Emotional Speech Synthesis with Combined Speaker-Emotion Embeddings

Jeffrey Chen



Deep Generative Text-to-Speech

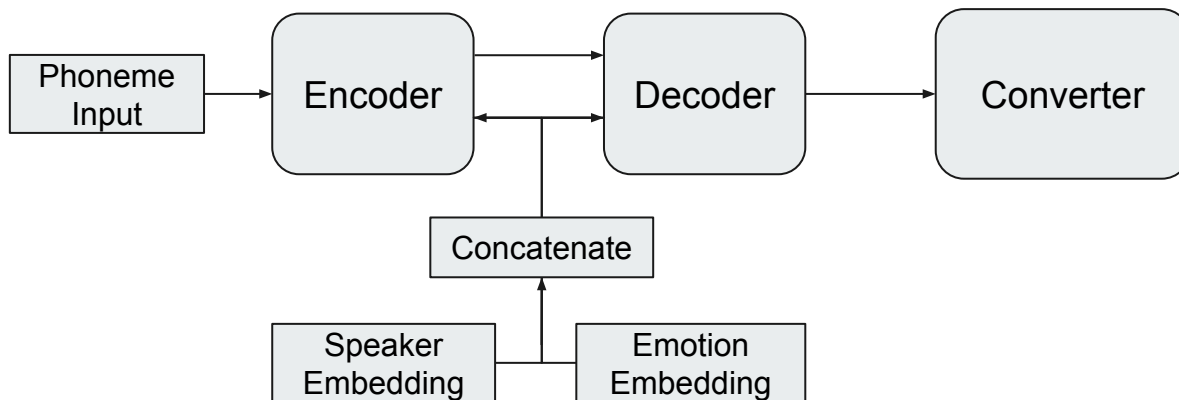
- Given text, speaker identity, and emotion, predict output waveform
- Learn joint probability

$$p(\mathbf{x}, s, m, \mathbf{y}) = \prod_{t=1}^T p(y_t \mid y_1, \dots, y_{t-1}, \mathbf{x}, s, m)$$

where \mathbf{x} is input text (preprocessed into phonemes), s is speaker, m is emotion, \mathbf{y} is output waveform

Deep Generative Text-to-Speech

- Use multi-speaker Deep Voice 3 (Ping et al. 2018) as baseline model
- Consists of encoder, decoder, and converter





Embedding Vectors

- To predict text for different speakers, we use low-dimensional speaker embedding vectors
- Augment with low-dimensional emotion embedding vectors to capture emotion features
- Concatenate speaker and emotion embedding vectors, then pass them into encoder and decoder
- Speaker and emotion embeddings learn different latent features

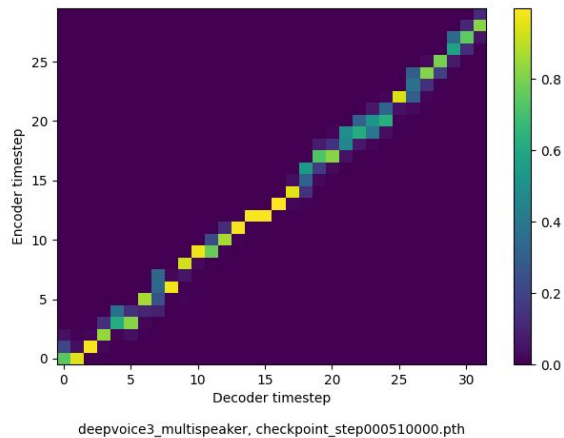


Methods

- Datasets:
 - VCTK: 108 speakers, ~44 hours of labeled speech, no labeled emotions (assume neutral)
 - RAVDESS: 24 speakers, 15 emotions, ~72 minutes of speech labeled with emotions
- Train on VCTK for achieving good speech synthesis, then finetune by training on both VCTK and RAVDESS to learn emotion embeddings

Results

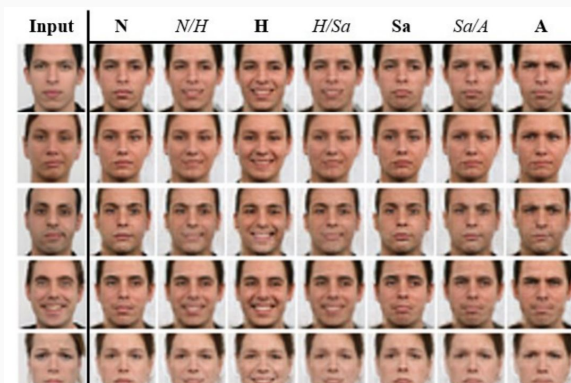
- Can generate speech for combinations of text, speaker, and emotion



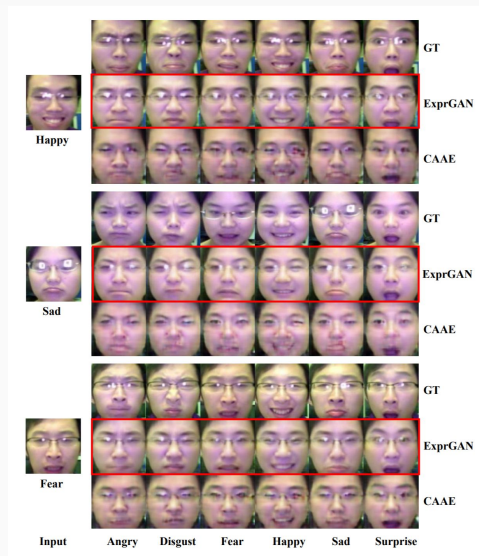
Wasserstein-GP CGAN Optimization For Facial Expression Synthesis

Tianlei Pan, Zheng Xu

Facial Expression Synthesis

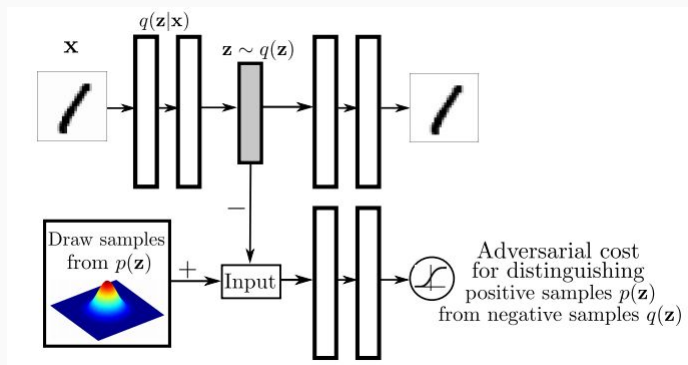


CDAAE

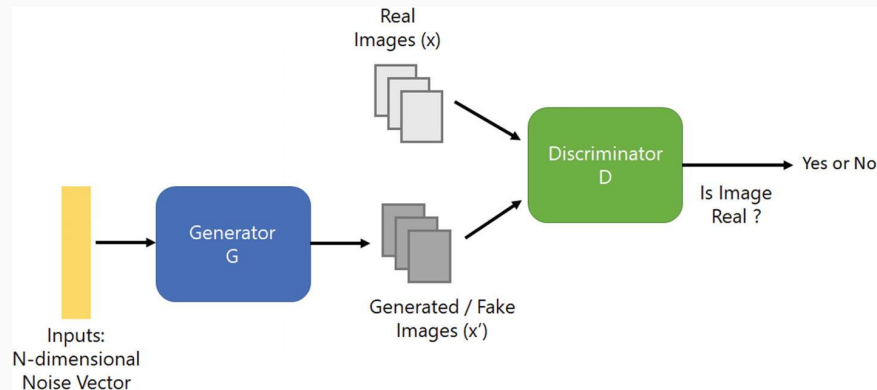


ExprGAN

Generative Models

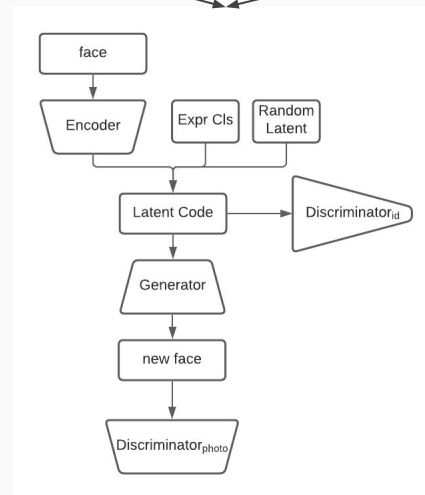
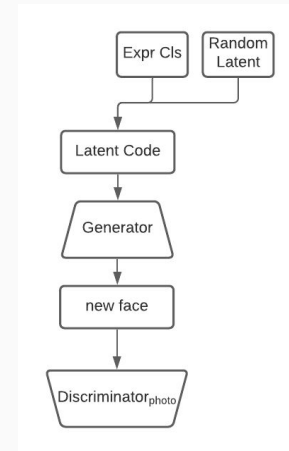
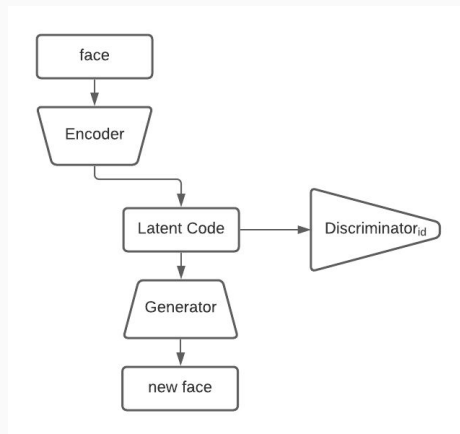


Adversarial Autoencoder

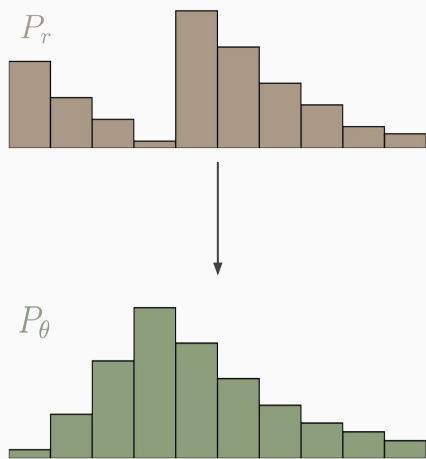


Generative Adversarial Network

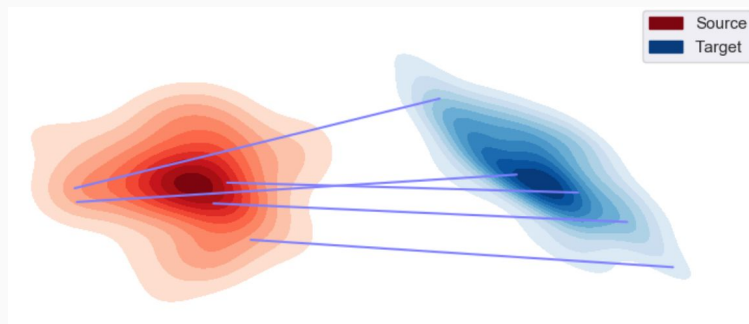
Network Structure



Wasserstein Distance



Wasserstein Distance



WGAN GP

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ , the number of critic iterations per generator iteration n_{critic} , the batch size m , Adam hyperparameters α, β_1, β_2 .

Require: initial critic parameters w_0 , initial generator parameters θ_0 .

```
1: while  $\theta$  has not converged do
2:   for  $t = 1, \dots, n_{\text{critic}}$  do
3:     for  $i = 1, \dots, m$  do
4:       Sample real data  $\mathbf{x} \sim \mathbb{P}_r$ , latent variable  $\mathbf{z} \sim p(\mathbf{z})$ , a random number  $\epsilon \sim U[0, 1]$ .
5:        $\tilde{\mathbf{x}} \leftarrow G_\theta(\mathbf{z})$ 
6:        $\hat{\mathbf{x}} \leftarrow \epsilon \mathbf{x} + (1 - \epsilon) \tilde{\mathbf{x}}$ 
7:        $L^{(i)} \leftarrow D_w(\hat{\mathbf{x}}) - D_w(\mathbf{x}) + \lambda(\|\nabla_{\hat{\mathbf{x}}} D_w(\hat{\mathbf{x}})\|_2 - 1)^2$ 
8:     end for
9:      $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2)$ 
10:   end for
11:   Sample a batch of latent variables  $\{\mathbf{z}^{(i)}\}_{i=1}^m \sim p(\mathbf{z})$ .
12:    $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^m -D_w(G_\theta(\mathbf{z})), \theta, \alpha, \beta_1, \beta_2)$ 
13: end while
```

Preliminary Results



Dataset



Generated

FER2013

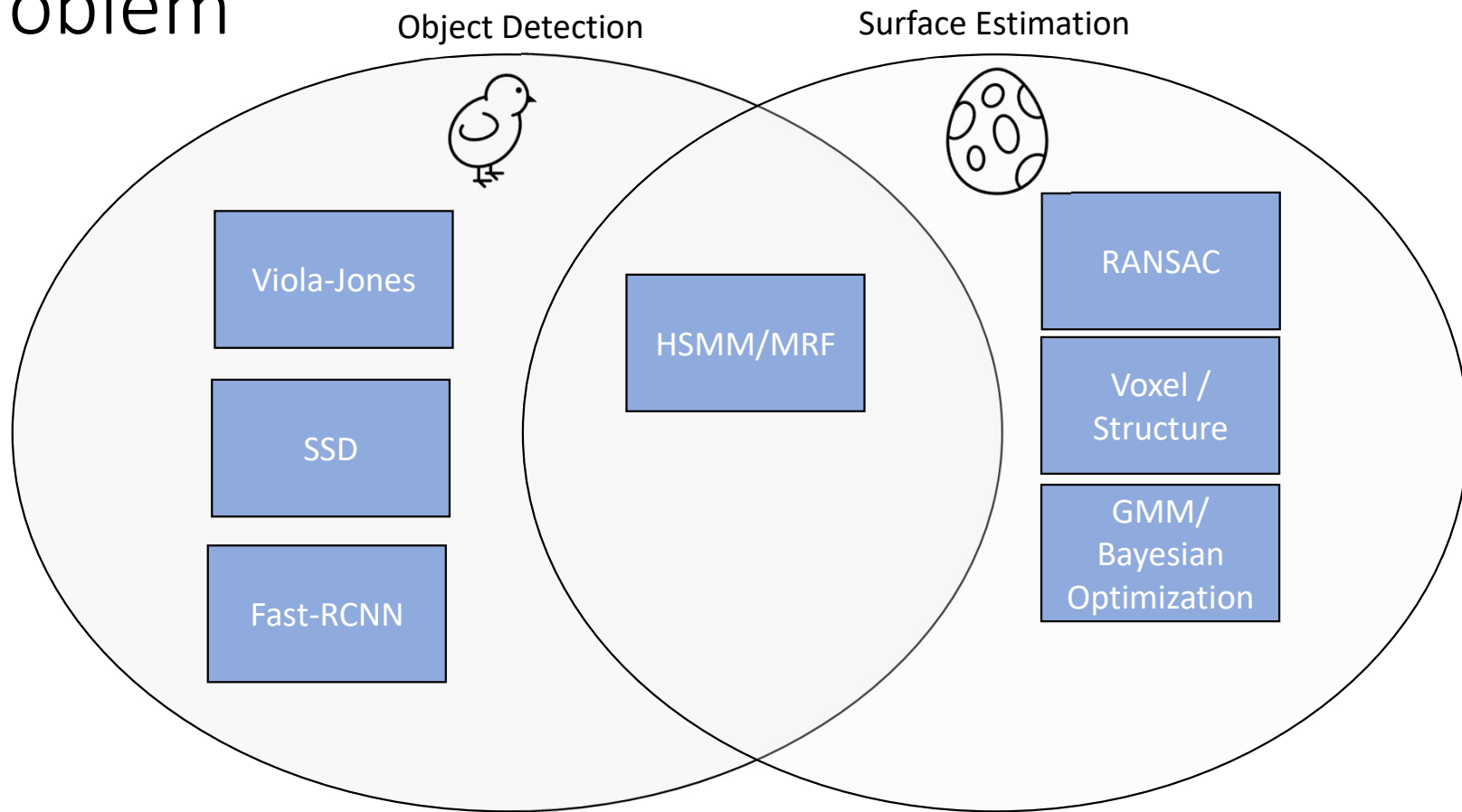
Probabilistic Graphical Models for Joint Object Detection and Surface Estimation

Matthew Levine (mjlevine@andrew) (12/12/2020)

Overview

- Object detection for autonomy off-road imposes additional challenges
- Goal: detect objects and surfaces
- Approach: integrate older probabilistic approaches with deep learned features for improved performance
- Benchmark old approaches against new approach with new dataset

Problem



Challenges

- Complicated correlations / conditioning
- Partial observability
- Ambiguous target state

Model

- **Appearance**
- Density
- Freespace
- Obstacles

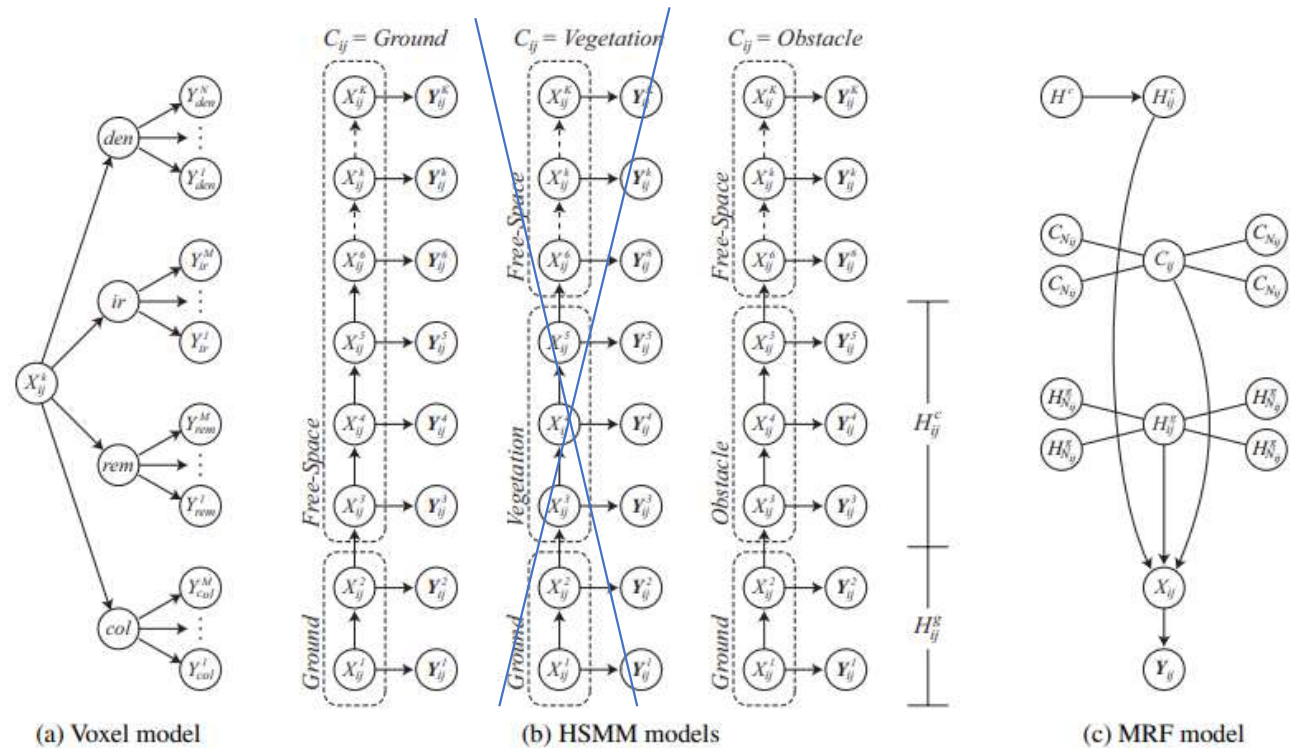


Fig. 5. A graphical description of the model showing (a) the voxel, (b) the voxel column, and (c) the connections between voxel columns. For each voxel column ij , the model contains voxel states X_{ij}^k , observations Y_{ij}^k , and a class C_{ij} , class height H_{ij}^c , and ground height H_{ij}^g that interact with neighbors N_{ij} and the common class height H^c .

Data

- Outdoor agricultural setting
- People hidden; many other obstacles
- Difficult terrain
 - Vegetation
 - Dips and runoffs
 - Relative flatness but some complicated inclination



***metrics are difficult here, all these come with caveats!**

Results

Model	Recall	Precision (Person)
RetinaNet	65.5%	92%
Base MRF	32.2%	61.9%
Deep MRF	61%	64%

Model	Mean l_2 Surface Error
RANSAC	0.32m
Base MRF	0.09
Deep MRF	0.13

Learning Nonparanormal DAGs from Data

Shantanu Gupta, Vishwak Srinivasan
{shantang,vishwaks}@andrew.cmu.edu

December 9, 2020

Methods for Learning DAGs

Constraint based

- ▶ Rely on conditional independence testing to learn the edges in the graph.
- ▶ Two classical algorithms are the PC algorithm [4] and the Fast Causal Inference (FCI) algorithm [1].

Score based

- ▶ Try to optimize a score $\mathcal{S} : \mathbb{D} \rightarrow \mathbb{R}$, where \mathbb{D} is the set of DAGs:

$$\min_G \mathcal{S}(G), \text{ subject to } G \in \mathbb{D}.$$

- ▶ The DAG constraint is challenging to enforce.
- ▶ Can be converted into a continuous constrained optimization problem in [5].

Nonparanormal distributions and Nonparanormal graphs

Nonparanormal distribution \equiv nonparametric extension to multivariate Gaussian distributions [3].

Why? Limiting nature of Normality.

$\overbrace{\text{NPN}(\Sigma, \{f_i\}_{i=1}^p)}$
Nonparanormal distribution : defined w.r.t. a covariance matrix $\Sigma \in \mathbb{S}_{++}^p$ and a collection of strictly increasing functions $\{f_i\}_{i=1}^p$.

$$(f_1(X_1), \dots, f_p(X_p)) \sim \text{NPN}(\Sigma) \Leftrightarrow (X_1, \dots, X_p) \sim \mathcal{N}(0, \Sigma)$$

Given a DAG G , a $\overbrace{\text{NPN}(G)}$ *nonparanormal graphical model*: the set of distributions $\text{NPN}(f, \Sigma)$ that are Markov w.r.t. G .

Rank-PC: a provable algorithm for learning nonparanormal DAGs

- ▶ PC algorithm [4] uses Gaussian conditional independence tests to infer a **CPDAG** from the given data.
 - ▶ Specifically, the test statistic is chosen to be the **Pearson correlation**.
- ▶ Key fact: PC algorithm provably finds the **true CPDAG** by only checking whether distinct pairs of nodes are d-separated by arbitrary sets of a certain size.
- ▶ Idea behind Rank-PC [2]
 1. *Simple* transformations of the **Spearman correlation** and **Kendall's τ** approximate the Pearson correlation well for multivariate normal data [3].
 2. Aforementioned measures are ranking-based and ranks are still intuitively preserved due to strictly increasing functions.

Preliminary Experiments

Our preliminary experiments focus on synthetically generated DAGs. We simulate random DAGs and sample from multivariate normal distributions that are faithful to them.

We choose the number of nodes in the random DAG to be 10 and 20 and sample 10 graphs each.

From each random DAG, we obtain $n = 1000$ samples to evaluate the methods. The transformation applied to these samples is $f(z) = \exp(z)$.

Method	Recall	Precision	Accuracy
Rank-PC	0.956 (0.059)	0.855 (0.153)	0.979 (0.015)
NOTEARS	0.352 (0.097)	0.778 (0.228)	0.917 (0.035)

Table: Results

Our Proposed Method

- ▶ Let $Y \sim NPN(f, \Sigma)$. Each node Y_i is associated with a hidden latent variable X_i such that $Y_i = f_i(X_i)$ where $X \sim \mathcal{N}(0, \Sigma)$.
- ▶ Thus we can write X as

$$X = WX + \epsilon,$$

where $\epsilon \sim \mathcal{N}(0, \Psi)$, and $\epsilon \perp\!\!\!\perp X$.

- ▶ NOTEARS requires two components: (i) a loss function L ; and (ii) a continuous constraint $H(W) = 0$ that becomes zero when W becomes a DAG.

Proposed Method

- ▶ Let \mathcal{G} and \mathcal{H} be an arbitrary *parametric* class of functions (e.g. neural networks). The loss function is

$$R = \frac{1}{n} \sum_{j=1}^n \|Y^{(j)} - h(g(Y^{(j)}))\|_2^2,$$

$$\begin{aligned} L(W, \Psi, g_1, \dots, g_p, h_1, \dots, h_p) \\ = -\log(\det(\Sigma)) + \frac{1}{n} \sum_{j=1}^n g(Y^{(j)})^\top \Sigma^{-1} g(Y^{(j)}) + \lambda |W|_1 + R. \end{aligned}$$

- ▶ The NOTEARS objective then is:

$$\min_{W, \Psi} \min_{g_i \in \mathcal{G}, h_i \in \mathcal{H}, i \in [p]} L(W, \Psi, g_{1:p}, h_{1:p}) \text{ subject to } H(W) = 0.$$

- ▶ Can be optimized using EM style methods.

References

- [1] C. Glymour, K. Zhang, and P. Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in genetics*, 2019.
- [2] N. Harris and M. Drton. Pc algorithm for nonparanormal graphical models. *The Journal of Machine Learning Research*, 2013.
- [3] H. Liu, J. Lafferty, and L. Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research*, 2009.
- [4] P. Spirtes, C. Glymour, R. Scheines, S. Kauffman, V. Aimale, and F. Wimberly. Constructing bayesian network models of gene expression networks from microarray data. 2000.
- [5] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing. Dags with no tears: Continuous optimization for structure learning. In *Advances in Neural Information Processing Systems*, 2018.

10-708 Course Project

Variational Inference for Federated Multi-Task Learning

Yichen Ruan

Dec 10, 2020

Background

- Federated Learning

A distributed learning framework without clients sharing raw data

Algorithm 1 FederatedAveraging. The K clients are indexed by k ; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

initialize w_0

for each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of m clients)

for each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$w_{t+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} w_{t+1}^k$

Synchronization

ClientUpdate(k, w): // Run on client k

$\mathcal{B} \leftarrow$ (split \mathcal{P}_k into batches of size B)

for each local epoch i from 1 to E **do**

for batch $b \in \mathcal{B}$ **do**

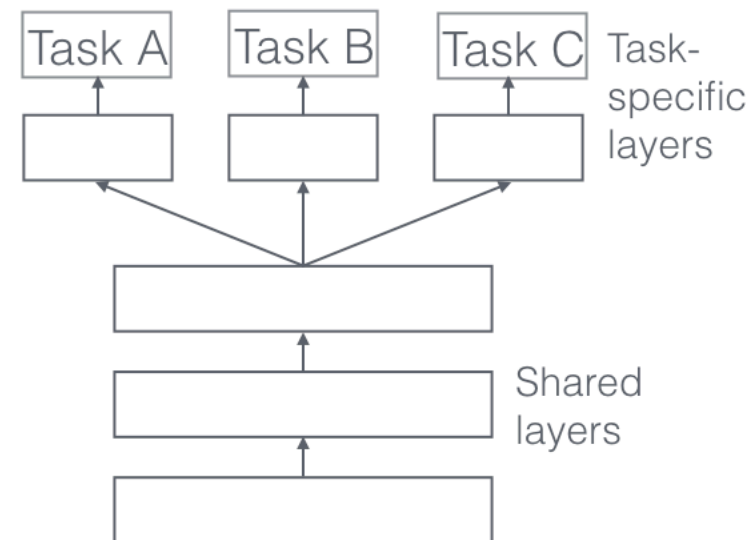
$w \leftarrow w - \eta \nabla \ell(w; b)$

Local update

 return w to server

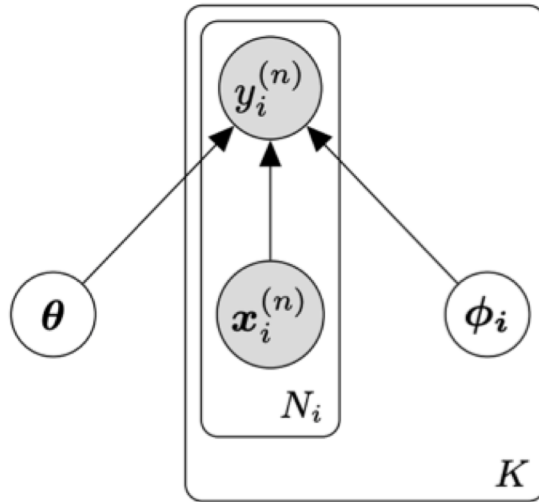
- Multi-task Learning

Learning of multiple local models with shared global information



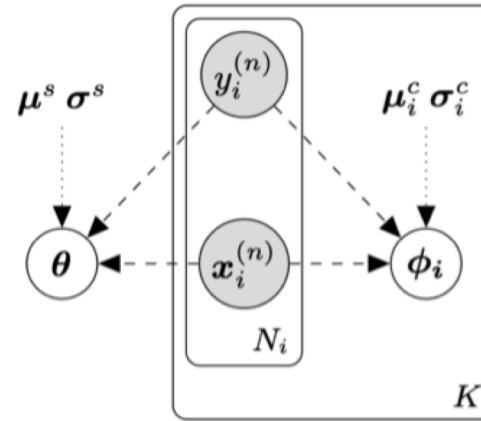
FMTL as Variational Inference

- PGM for FMTL



$$p(\theta, \phi_1, \dots, \phi_K | D_{1:K}) \propto \prod_{i=1}^K p(\theta, \phi_i | D_i)$$

- Variational PGM



Gaussian mean-field approximation:

$$s(\theta) = \prod_{d=1}^{\dim(\theta)} \mathcal{N}(\theta_d | \mu_d^s, \sigma_d^s)$$

$$c_i(\phi_i) = \prod_{d=1}^{\dim(\phi_i)} \mathcal{N}(\phi_{id} | \mu_{id}^c, \sigma_{id}^c)$$

$$q(\theta, \phi_1, \dots, \phi_K) = s(\theta) \prod_{i=1}^K c_i(\phi_i)$$

$$q_i(\theta, \phi_i) = s(\theta) c_i(\phi_i)$$

Variational FMTL

Algorithm 1: Variational Federated Multi-Task Learning

Input: Datasets $\{D_i\}_{i=1}^K$, $|D_i| = N_i$, priors $p(\theta)$, $\{p(\phi_i)\}_{i=1}^K$, number of global epochs T , number of local epochs E .

Initialize all variational posteriors;

for $t = 1, 2, \dots, T$ **do**

% Client Procedure;

for $i = 1, 2, \dots, K$ *in parallel* **do**

$s_i^{t,0}(\theta) \leftarrow s^t(\theta);$

$c_i^{t,0}(\phi_i) \leftarrow c_i^{t-1,E}(\phi_i);$

for $e = 1, 2, \dots, E$ **do**

$s_i^{t,e}(\theta)c_i^{t,e}(\phi_i) \leftarrow SVI(s_i^{t,e-1}(\theta)c_i^{t,e-1}(\phi_i), p_i(\theta, \phi_i|D_i));$

end

end

% Server Procedure;

$s^{t+1}(\theta) \leftarrow \sum_{i=1}^K \frac{N_i}{\sum_{j=1}^K N_j} s_i^{t,E}(\theta)$

end

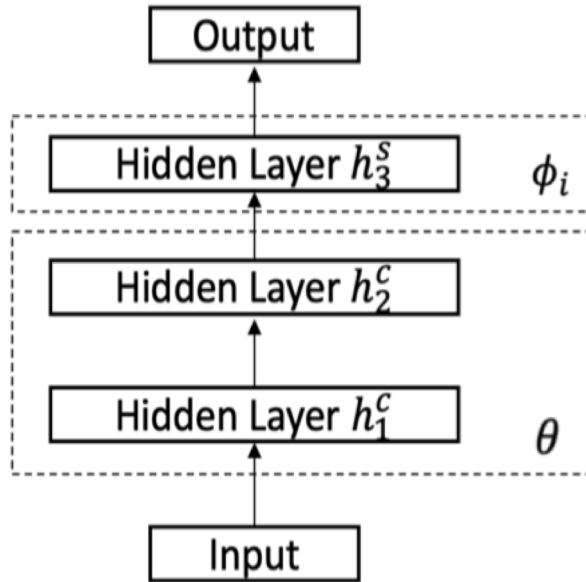
Local update with stochastic variational inference

Synchronization of global variational parameters

$$\begin{aligned} \mu_s &\leftarrow \sum_k \alpha_k \mu_s(k), \alpha_k = \frac{N_k}{\sum N_i} \\ \sigma_s^2 &\leftarrow \sum_k \alpha_k \sigma_s^2(k) + \sum_k \alpha_k (\mu_s - \mu_s(k))^2 \end{aligned}$$

Experiments

- MNIST



- $784 \times 1024 \times 512 \times 256 \times 10$ with ReLu
- All priors initialized to $N(0,1)$
- Non-IID Data partition: 10 clients, each got 1 label
- Test scheme: sampling 100 models and average

Table 1: Average test accuracy with different number of local epochs for MNIST.

	Proposed-NIID	FedAvg-NIID	Proposed-IID	FedAvg-IID
$E = 10$	0.82	0.80	0.91	0.90
$E = 30$	0.80	0.79	0.87	0.88
$E = 50$	0.72	0.75	0.81	0.83

Inflection-aware Data Augmentation for Low-resource Language Machine Translation

Ye Yuan, Xingyuan Zhao

Motivation

- NMT: Data-hungry \nrightarrow Low-Resource Languages
 - **Data augmentation** needed for low-resource MT
- Current Word replacement methods consider no morphology
 - Morphology is specially important for many low-resource languages which are morphologically rich
 - Hard to obtain grammatically correct sentences with proper inflection
- Language Models for low-resource languages are not good enough for inflection
- Propose an inflection-aware data augmentation method to rectify the grammatical mistakes in the sentence pairs generated by word replacement for low-resource MT

Method: Overview

These are cute puppies --> Das sind süße Welpen

These are cute **cat** --> Das sind süße **Katze**

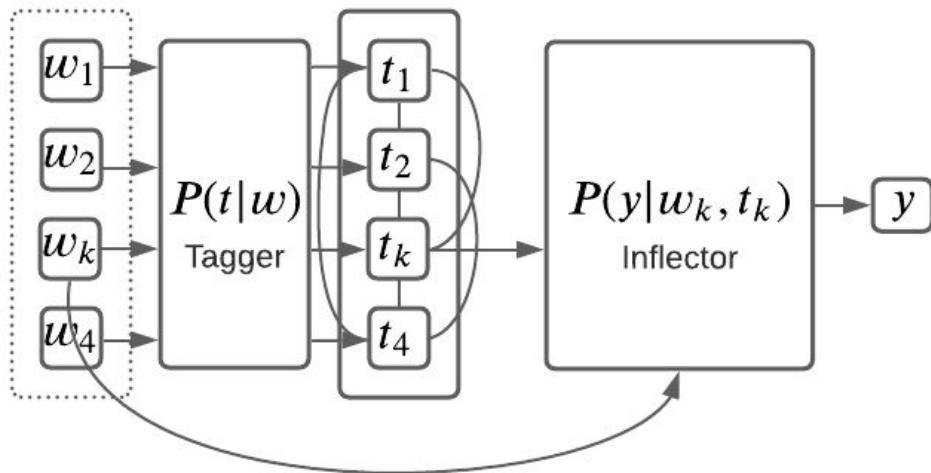
These are cut **cats** --> Das sind süße **Katzen**

- Word replacement & alignment
 - Language model prediction in high-resource language
 - Fast alignment

Method: Inflector

- Contextual inflection

- Das sind süße Katze
- (DET;DEF V;IND;3;PRS;FIN ADJ)



Preliminary Results

Model	Data	test2016
Baseline	192K	4.92
Naive-Replace	235K	4.52
LM-Replace		
Inflected-Replace		

Thank you

On Model-Enhanced A2C methods in Deep RL

Di Wang, Alvin Pan

Introduction and Problem Formulation

- Model-free RL, with an actor-critic (A2C) model, has been a goto model in a variety of simulation environments
- Model-based methods have been less explored, especially in the actor-critic framework and context
- We aim to introduce model-based methodology to the Actor-Critic model in order to improve the training efficiency of the A2C paradigm

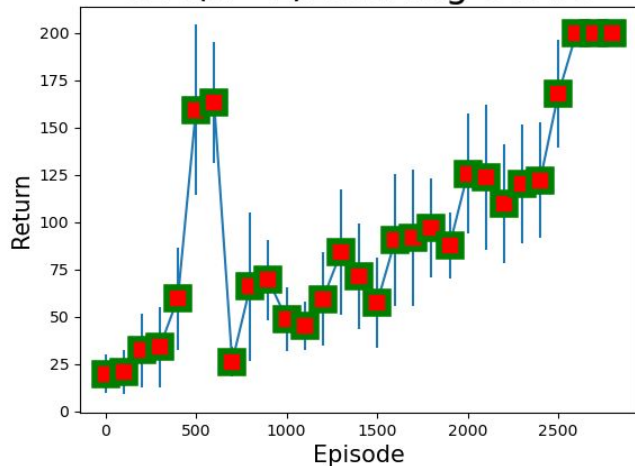
Methodology

- Main Algorithm (train until converged for a given interval of t episodes):
- Sample real trajectory steps w.r.t. a given environment and buffer them
- Use these trajectory steps to train an ensemble model based on PETS of the environment dynamics for a predetermined number of iterations
- Use the transition of the model to train an actor critic A2C instance for t episodes
- Rinse and repeat until converged

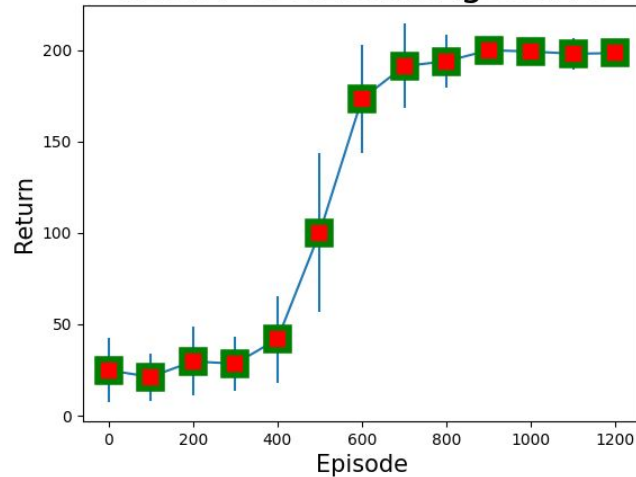
Important: Backpropagate using importance sampling involving real trajectory and its corresponding virtual trajectory

Results: Cartpole

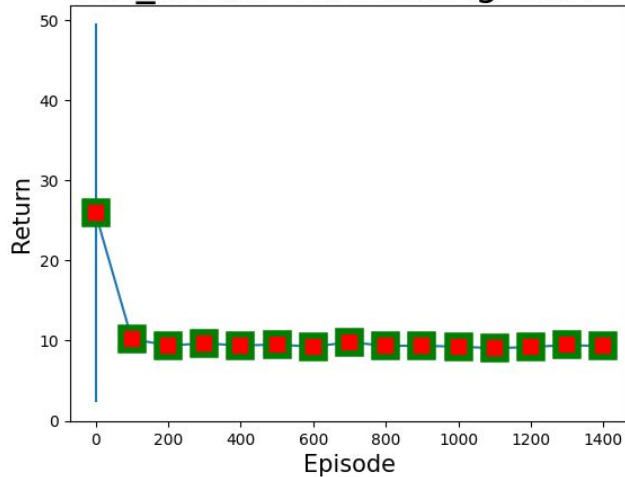
a2c(N=1) Learning Curve



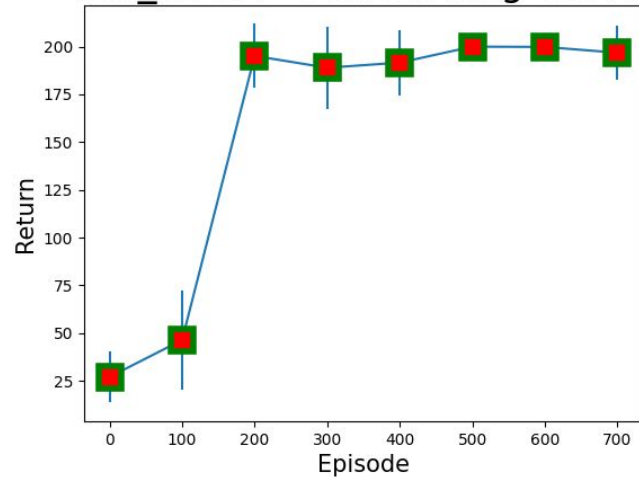
a2c(N=50) Learning Curve



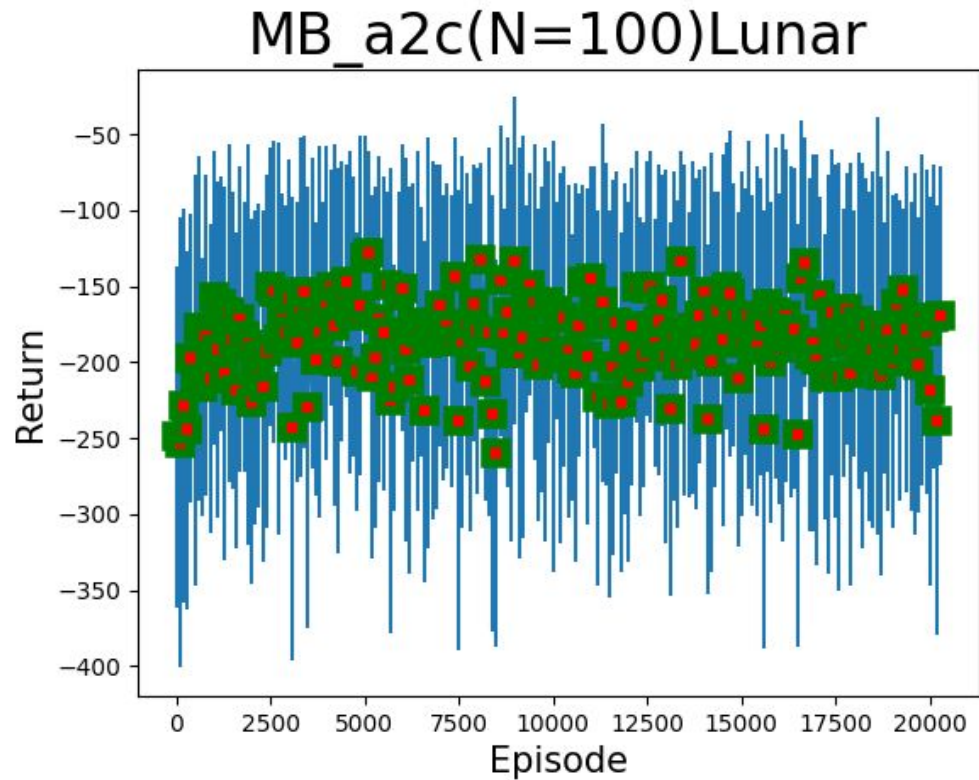
MB_a2c(N=1) Learning Curve



MB_a2c(N=50) Learning Curve



Results: Lunarlander



References

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

The Effect of Fact-checking on the Spread of Information

Ramon Villa-Cox

Melda Korkut

Introduction

- We seek to quantify the Causal effect of a tweet being called out as false on its diffusion over the network.
- The treatment is defined based on the presence of a fact-checking response.
 - These were identified by the usage of a URL from a prominent fact-checking domain (e.g. poltifact, snopes, etc.).
- The data was collected during the first months of the COVID-19 crisis. Rumors were identified using a classifier that achieved aprox. 90% f1-score in that task. We match treated and non-treated rumors by controlling for semantic and network characteristics.

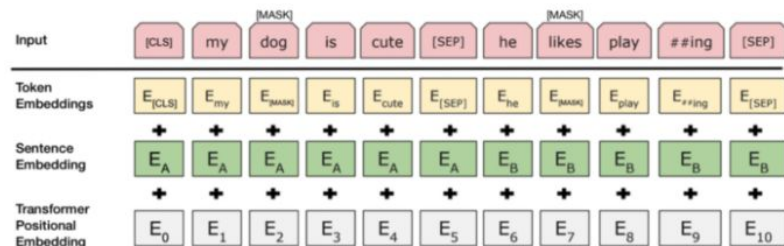
Rumor Diffusion

- We fine-tuned the rumor classification model from (Memon & Carley, 2020), which was applied to identify rumors in a COVID-19 Twitter dataset.
 - The model uses a BERT embedding layer combined with 2 MLP layers to predict whether a tweet contains a rumor or not.
- We identified approximately 100k rumors, 12k of which had a fact checking response.
- We measure the overall diffusion of a rumor based on the number of retweets that it obtained through its life-cycle.

Confounder Matching

Semantic Similarities:

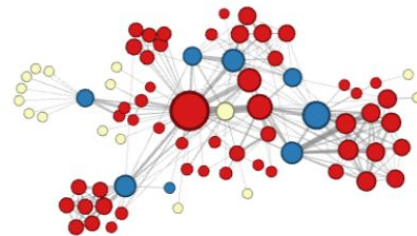
- We obtain the BERT embeddings of each identified rumor (fine-tuned for the rumor classification task).
- These 768 dimensional vector encode the semantic characteristics of the rumors. We hypothesize that similar rumors (written in similar ways) will be close in this space.



Source: <https://arxiv.org/pdf/1810.04805.pdf>

Network Influence:

- We obtain node2vec embeddings of the users creating the rumors in their retweet networks.
- These 128 dimensional vectors encode the structural equivalency of users in the network. Users that are close in this space have similar access to the overall retweet network.



Source: <https://cs.stanford.edu/~jure/pubs/node2vec-kdd16.pdf>

Average Treatment Effect

$$ATE = \frac{1}{N} \sum_{i=1}^N (Y_{i1} - Y_{i0})$$

We calculate the average treatment effect by using matching between the treated and untreated population.

Propensity score estimation

For confidence intervals around this point estimate, to make it more robust, we employ bootstrapping to get confidence intervals around this estimate.

Difference in differences