Deep Generative Models at Scale

-- Opportunities, Challenges and Applications

Chunyuan Li

March 19, 2020

http://chunyuan.li

Outline

At a small scale → Definition & Background

At a large scale

① OPTIMUS: Opportunities in Language Modeling

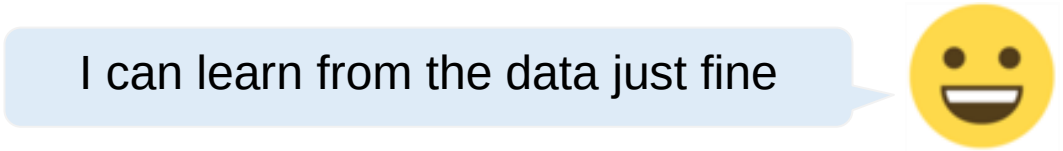
② FQ-GAN: Challenges in Image Generation

③ PREVALENT: Applications to Vision-and-Language Navigation

Background

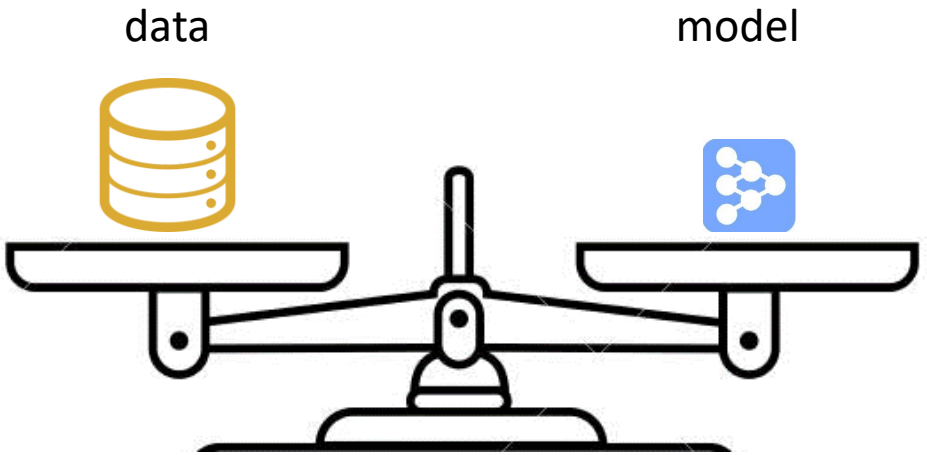
To train a generative model, we first collect a large amount of data in some domain (e.g., think of images, sentences, or sounds, etc.) and then **train a model to generate data like it**.

"What I cannot create, I do not understand."
—Richard Feynman



The trick of DGMs:

The trick is that the neural networks we use as generative models (i.e. DGM) have a number of parameters significantly smaller than the amount of data we train them on, so **the models are forced to discover and efficiently internalize the essence of the data** in order to generate it.

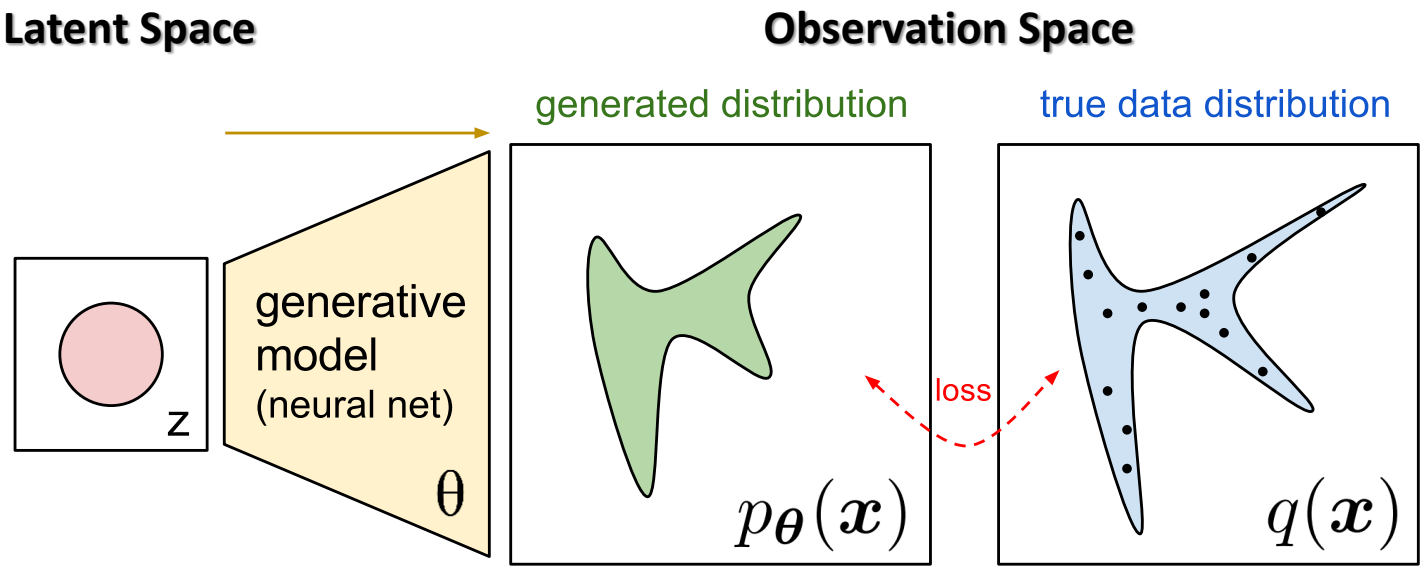


https://openai.com/blog/generative-models/

Definition

Generative Modeling: $\tilde{\boldsymbol{x}} = g_{\boldsymbol{\theta}}(\boldsymbol{z}, \boldsymbol{\epsilon}),\ \boldsymbol{z} \sim p(\boldsymbol{z})$; Goal $p_{\boldsymbol{\theta}^*}(\boldsymbol{x}) = q(\boldsymbol{x})$

Representation Learning: $\tilde{\boldsymbol{z}} = g_{\boldsymbol{\phi}}(\boldsymbol{x}, \boldsymbol{\zeta}),\ \boldsymbol{x} \sim q(\boldsymbol{x})$



https://openai.com/blog/generative-models/

ϵ and ζ are optional stochastic sources

Taxonomy

Most generative models have the basic setup of modeling data generation process but differ in the details. Here are three popular examples:

Models	Key Concept	Pros	Cons
Variational Autoencoders (VAEs)	An encoder-decoder framework via probabilistic graphical models, where we are maximizing a lower bound on the log likelihood of the data	Simultaneously perform both generation and inference with latent variables	Generated samples tend to be slightly blurry
Generative Adversarial Networks (GANs)	A generator-discriminator framework via an adversarial training game. where we are directly generating samples of the data	Generate the sharpest samples	More difficult to optimize due to unstable training dynamic
Autoregressive models (e.g. PixelRNN, Neural LM)	Factorize the joint distribution of data into the conditional distributions, modeling every individual dimension given previous dimensions	Simple and stable training, yielding the best log likelihood	Inefficient during sampling and don't easily provide low-dimensional features

https://openai.com/blog/generative-models/

Current Research Status

Most in Academia:
Theoretical principles/connections/advances of DGMs, and their applications to new domains, e.g., language, music etc.

arXiv.org

Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, Yoshua Bengio

(Submitted on 12 Mar 2020)

Variational Autoencoders and Nonlinear ICA: A Unifying Framework

Ilyes Khemakhem, Diederik P. Kingma, Ricardo Pio Monti, Aapo Hyvärinen

(Submitted on 10 Jul 2019 (v1), last revised 26 Feb 2020 (this version, v3))

...

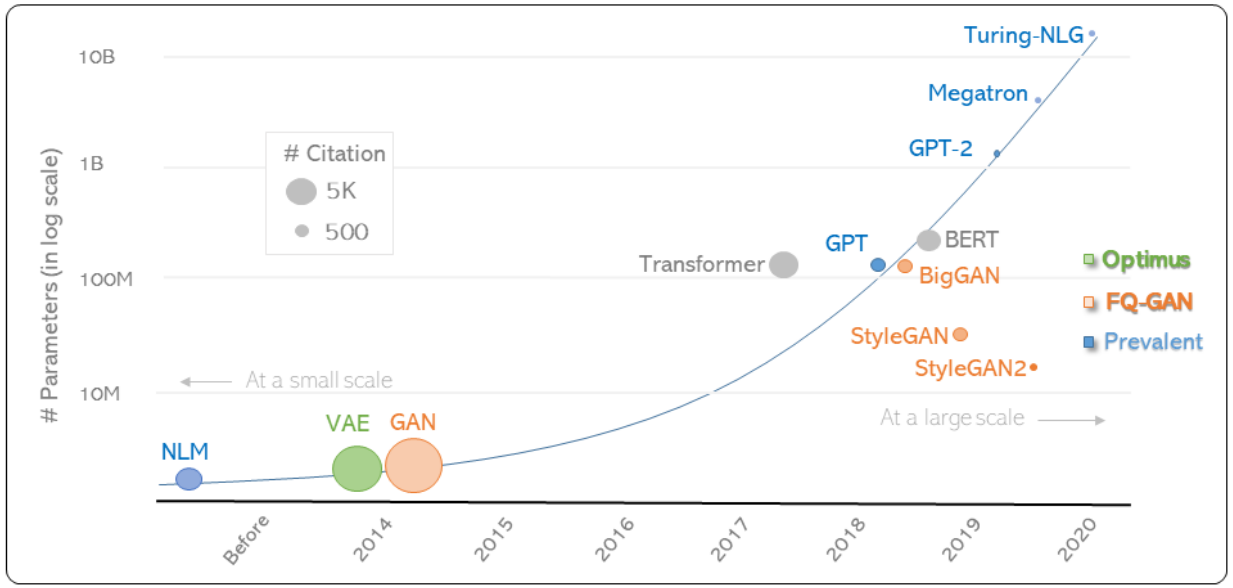
Less for Practioners:
How Good are the Deep Generative Models Really? especially when we face massive data in industrial practice?

"Two roads diverged in a wood and I took the one less traveled by, and that has made all the difference."
-- Robert Frost

At Scale: data & computing

Current Trends: Strong empirical results via pre-training on **massive data** with massive computing

The trick of DGMs is less studied at a large scale



I have never afforded this much data

① **Opportunity**: How good could it be with pre-training?

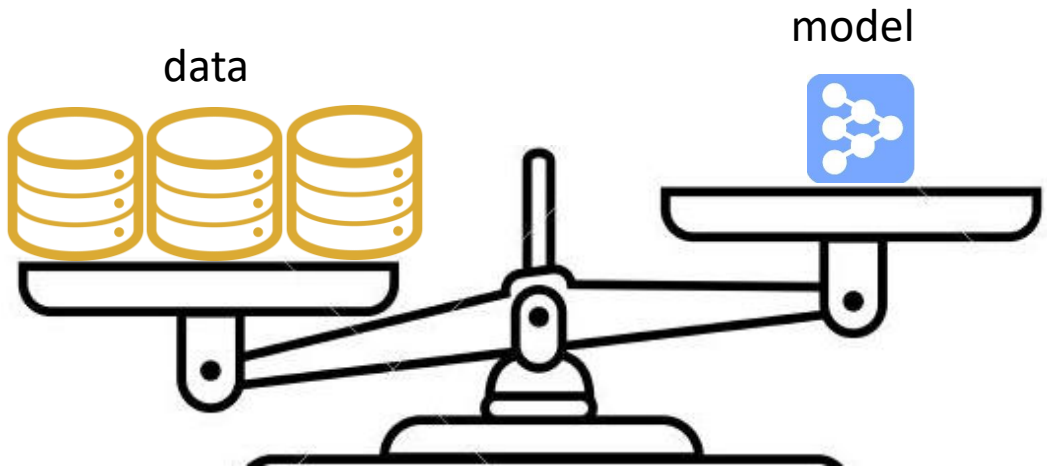
OPTIMUS: Opportunities in Language Modeling

② **Challenge**: The traditional methods do NOT work well

FQ-GAN: Challenges in Image Generation

③ **Application**: How could it benefit pre-training?

PREVALENT: Data augmentation for pre-training VLN



① Optimus: Organizing Sentences via Pre-trained Modeling of a Latent Space

C. Li, X. Gao, Y. Li, X. Li, B. Peng, Y. Zhang, J. Gao

Pre-trained Language Models (PLMs)

PLMs are great! Achieving state-of-the-art performance in various domains.

Existing PLMs

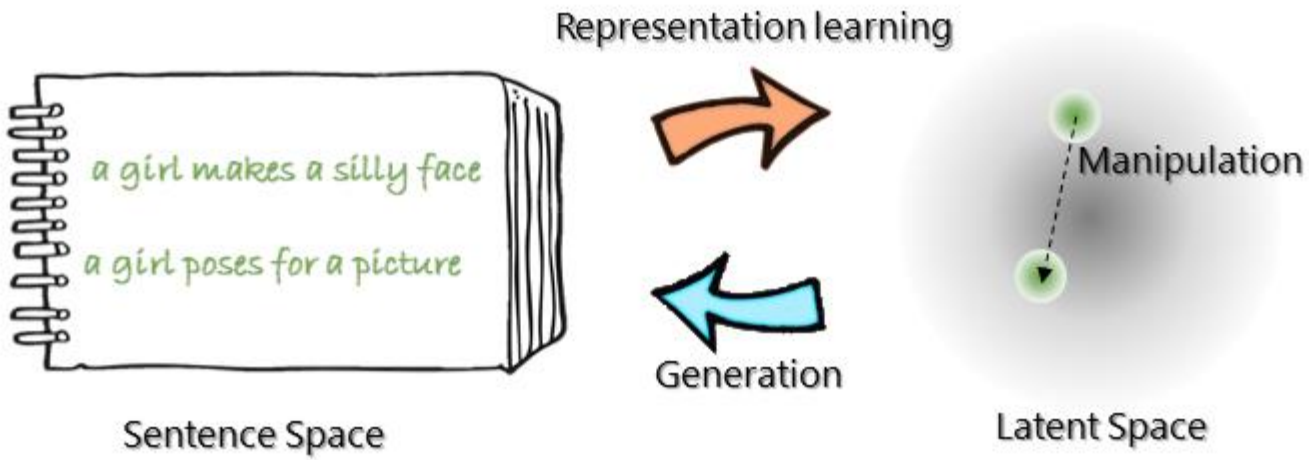
Understanding	Generation	Understanding & Generation
• BERT • Roberta • Albert	• GPT-2 • Megatron • Turing	• UniLM • T5 • BART

Issue: Lack of explicit modeling of structures in a latent space, rendering it difficult to control natural language generation / understanding from an abstract level

VAE at a small scale

Promise:

- A latent variable model, allowing generation and representation learning simultaneously
- By representing sentences in a low-dimensional latent space, VAEs allow easy manipulation of sentences using the corresponding compact vector representations



Issues of existing language VAEs:

- Too small (e.g., 2-layer LSTM) that **the trick of DGMs** breaks;
- KL vanishing, not really easy to train

Neural Language Models (NLM) & GPT-2

To generate a sentence of length *T*, $\boldsymbol{x} = [x_1, \cdots, x_T]$

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_t | x_{<t}), \qquad (1)$$

all tokens before t

Issues:

- The only source of variation is modeled in the conditionals at every step
- No high-level control of the sentence, such as tense, topics or sentiment

NLM *vs* **VAE** (decoder)

To generate a sentence of length *T*, $\boldsymbol{x} = [x_1, \cdots, x_T]$.

$$p(\boldsymbol{x}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_t | x_{<t}), \qquad (1)$$

all tokens before t

Latent variable

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_t | x_{<t}, \boldsymbol{z}). \qquad (2)$$

all tokens before t

Key Insight:

- A latent variable ***z*** indicates high-level semantics to guide the sequential language generation

VAE (the full training objective)

Framework

- Encoder or inference network $q_{\phi}(\boldsymbol{z}|\boldsymbol{x})$
- Decoder or generation network $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$

Training Objective:

$$\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq \mathcal{L}_{\text{ELBO}} = \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] - \text{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \tag{3}$$

Reconstruction Term KL Term

KL Vanishing Issue (Optional):

- KL term degenerates to 0
- VAE reduces to NLM, the learned features become identical to Gaussian prior (not informative at all)

Optimus -- **O**rganizing sentences via **P**re-**T**rai**n**ed **M**odeling of a **U**niversal **S**pace

Settings

We focus on modeling **sentences** of a moderate length

- NOT text sequence chunks of fixed length
- NOT long-form text sequence such as paragraphs, document etc.

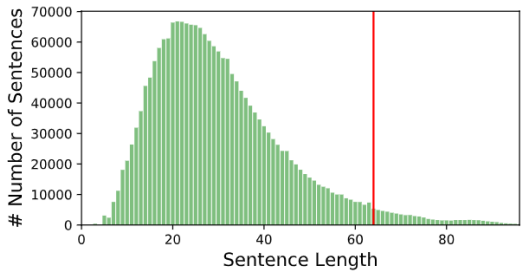
Why?

We deliberately keep a simple model:

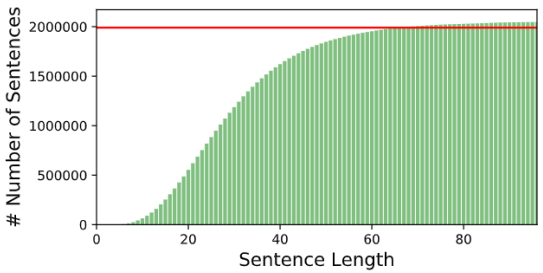
- Only a single low-dimensional latent vector to represent a sentence
- Controllability degraded for longer text sequences

This setting covers a large percentage of commonly seen sentences.

Pre-training dataset: Wikipedia



(a) Frequency distribution

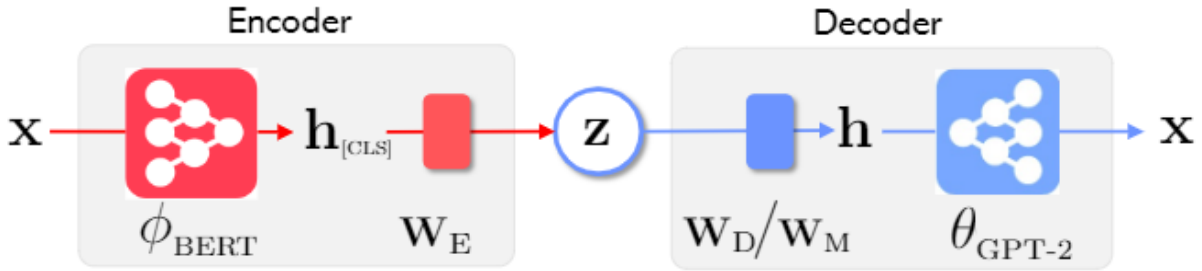


(b) Cumulative frequency distribution

We choose maximum length **64** to construct the pre-training dataset. It leads to 1990K sentences, which is **96.45%** of entire Wikipedia dataset

Optimus -- Pre-training

Architecture & Initialization



Latent Vector Injection

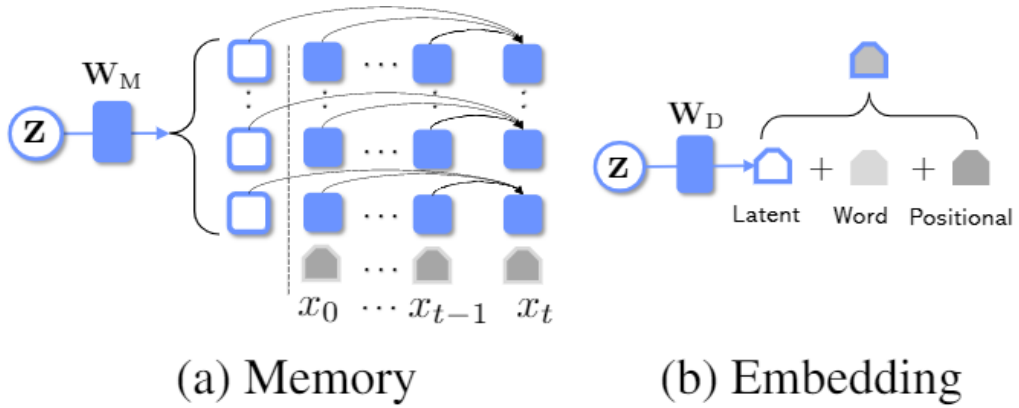


Figure 2: Illustration of two schemes to inject latent vector. (a) Memory: x_t attends both $x_{<t}$ and $\boldsymbol{h}_{\texttt{Mem}}$; (b) Embedding: latent embedding is added into old embeddings to construct new token embedding $\boldsymbol{h}'_{\texttt{Emb}}$.

Pre-training Schedule

- Cyclical annealing schedule for the KL term [*]
- Dimension-wise thresholding of the KL term, with hyper-parameter λ

[*] Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing H. Fu*,C. Li*, X. Liu, J. Gao, A. Celikyilmaz, L. Carin, NAACL 2019

Optimus -- Fine-tuning (1/3): Language Modeling

Fine-tuning pre-trained models for one epoch, evaluated with two types of metrics

- Generation capability: Perplexity(*PPL*)
- Representation learning capability: Active units (*AU*) of z and its Mutual Information (*MI*) with x.

Dataset		PTB			YELP			YAHOO			SNLI		
		LM	Repr.		LM	Repr.		LM	Repr.		LM	Repr.	
	Method	PPL ↓	MI ↑	AU ↑	PPL ↓	MI ↑	AU ↑	PPL ↓	MI ↑	AU ↑	PPL ↓	MI ↑	AU ↑
Optimus	$\lambda=0.05$	**23.58**	3.78	32	**21.99**	2.54	32	22.34	5.34	32	**13.47**	3.49	32
	$\lambda=0.10$	23.66	4.29	32	21.99	2.87	32	22.57	5.35	32	13.48	4.65	32
	$\lambda=0.25$	24.34	5.98	32	22.20	5.31	32	22.43	6.01	32	14.08	7.22	32
	$\lambda=0.50$	26.69	7.64	32	22.79	7.67	32	23.11	8.85	32	16.67	8.89	32
	$\lambda=1.00$	35.53	**8.18**	32	24.59	**9.13**	32	24.92	**9.18**	32	29.63	**9.20**	32
Small VAE	M. A.	101.40	0.00	0	40.39	0.13	1	61.21	0.00	0	21.50	1.45	2
	C. A.	108.81	1.27	5				66.93	2.77	4	23.67	3.60	5
	SA-VAE					1.70	8	60.40	2.70	10			
	Aggressive	99.83	0.83	4	39.84	2.16	12	59.77	2.90	19	21.16	1.38	5
	AE-BP	96.86	5.31	32	47.97	7.89	32	59.28	8.08	32	21.64	7.71	32
	GPT-2	24.23	-	-	23.40	-	-	22.00	-	-	19.68	-	-
	LSTM-LM	100.47	-	-	42.60	-	-	60.75	-	-	21.44	-	-
	LSTM-AE	-	8.22	32	-	9.24	32	-	9.26	32	-	9.18	32

- **Pre-training is a new way to reduce KL vanishing**
- **Lower PPL than GPT-2 due to the knowledge encoded in latent space**

Optimus -- Fine-tuning (2/3): Guided Language Generation; Simple Manipulation

Sentence transfer via arithmetic operation $\boldsymbol{x}_D \approx \boldsymbol{x}_B - \boldsymbol{x}_A + \boldsymbol{x}_C$ at the semantic level

Source $\boldsymbol{x}_A$	**Target** $\boldsymbol{x}_B$
a girl makes a silly face	two soccer players are playing soccer
Input $\boldsymbol{x}_C$	**Output** $\boldsymbol{x}_D$
• a girl poses for a picture	• two soccer players are at a soccer game.
• a girl in a blue shirt is taking pictures of a microscope	• two football players in blue uniforms are at a field hockey game
• a woman with a red scarf looks at the stars	• two men in white uniforms are field hockey players
• a boy is taking a bath	• two baseball players are at the baseball diamond
• a little boy is eating a bowl of soup	• two men are in baseball practice

Interpolating between two sentences $\boldsymbol{z}_\tau = \boldsymbol{z}_1 \cdot (1-\tau) + \boldsymbol{z}_2 \cdot \tau$

0.0	children are looking for the water to be clear.
0.1	children are looking for the water.
0.2	children are looking at the water.
0.3	the children are looking at a large group of people.
0.4	the children are watching a group of people.
0.5	the people are watching a group of ducks.
0.6	the people are playing soccer in the field.
0.7	there are people playing a sport.
0.8	there are people playing a soccer game.
0.9	there are two people playing soccer.
1.0	there are two people playing soccer.

Compare with GPT-2, these are new ways one can play with language generation

Optimus -- Fine-tuning (2/3): Guided Language Generation; Sophisticated Manipulation

- Dialog response generation `Dailydialog`

Metrics	Seq2Seq	CVAE	WAE	$iVAE_{MI}$	OPTIMUS
Recall↑	0.232	0.265	0.289	0.355	**0.362**
Precision↑	0.232	0.222	0.266	0.239	**0.313**
F1↑	0.232	0.242	0.277	0.285	**0.336**

- Stylized response generation `Dailydialog + Holmes`

Methods	Recall↑	Precision↑	F1↑	Neural↑	N-gram↑
StyleFusion	0.374	0.242	0.294	0.1050	0.1495
OPTIMUS	**0.385**	**0.268**	**0.316**	**0.1191**	**0.1645**

- Label-conditional text generation `Yelp`

Metrics	Control-Gen	ARAE	NN-Outlines	OPTIMUS
Accuracy↑	0.878	0.967	0.553	**0.998**
Bleu ↑	0.389	0.201	0.198	**0.398**
G-score↑	0.584	0.442	0.331	**0.630**
Self-Bleu↓	0.412	0.258	0.347	**0.243**

These tasks rely on a hierarchical generation process:

1. ***first the latent vector (the outlines of the target),***
2. ***then target sentences***

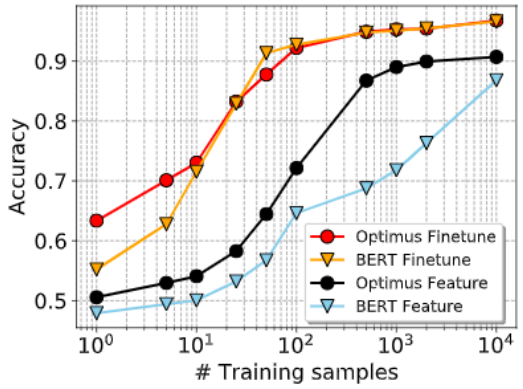
The pre-trained latent space alleviate the learning burden of downstream tasks, thus improve performance

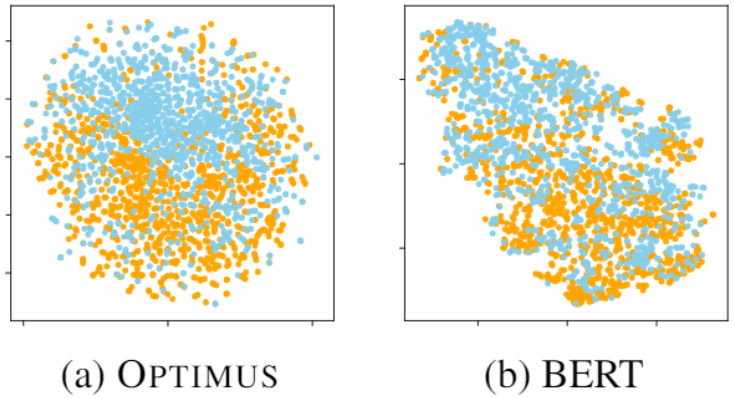
Optimus -- Fine-tuning (3/3): Low-resource Language Understanding

- *Fine-tuning:* both the pre-trained model and the linear classifier are updated;
- *Feature-based:* pre-trained mode weights are frozen to provide embeddings for the update of the classifier.

Feature-based method maintains the pre-trained smooth latent structures, and thus helps generalization

Yelp





(a) OPTIMUS (b) BERT

GLUE

System / Dataset size		MNLI 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	WNLI 634	**Average**
Feature-based	BERT	0.414	0.146	0.673	0.731	0.187	0.690	0.812	0.549	0.577	0.531± 0.011
	OPTIMUS	**0.468**	**0.662**	**0.720**	**0.789**	0.144	**0.719**	**0.816**	**0.585**	0.563	**0.607**± 0.013

② Feature Quantization Improves GAN Training

Y. Zhao*, **C. Li***, P. Yu, J. Gao, and C. Chen (*Equal contribution)

GANs

Framework

- Generator $\tilde{x} \sim p_{\theta}(x|z)$ implemented via $\tilde{x} = g_{\theta}(z),\ z \sim p(z)$
- Discriminator $f_{\omega}(x)$.

Training Objective:

$$\min_{\theta} \max_{\omega} \mathcal{L}_{\text{GAN}} = \mathbb{E}_{x \sim q(x)}[\log \sigma(f_{\omega}(x))] + \mathbb{E}_{\tilde{x} \sim p_{\theta}(x|z), z \sim p(z)}[\log(1 - \sigma(f_{\omega}(\tilde{x})))].$$

Data Distribution Matching : $p_{\theta^*}(x) = q(x)$

Feature Matching: $d_{\mathcal{F}}(p, q) = \sup_{f \in \mathcal{F}} |\mathbb{E}_{\tilde{x} \sim p(\tilde{x})} f(\tilde{x}) - E_{x \sim q(x)} f(x)|$

Poor estimate in a continuous feature space

1. Current mini-batch estimate scheme can be prohibitively inaccurate when facing large or complex datasets

2. Even worse, fake data distribution is changing during training. The underlying distribution is hard to capture.

1. Estimated using mini-batch statistics

Feature Matching: $$d_{\mathcal{F}}(p, q) = \sup_{f \in \mathcal{F}} |\mathbb{E}_{\tilde{\boldsymbol{x}} \sim p(\tilde{\boldsymbol{x}})} f(\tilde{\boldsymbol{x}}) - \mathbb{E}_{\boldsymbol{x} \sim q(\boldsymbol{x})} f(\boldsymbol{x})|$$

2. A dynamic distribution over time

From Continuous to Quantized Representations

- Limiting the feature space, enabling implicit feature matching

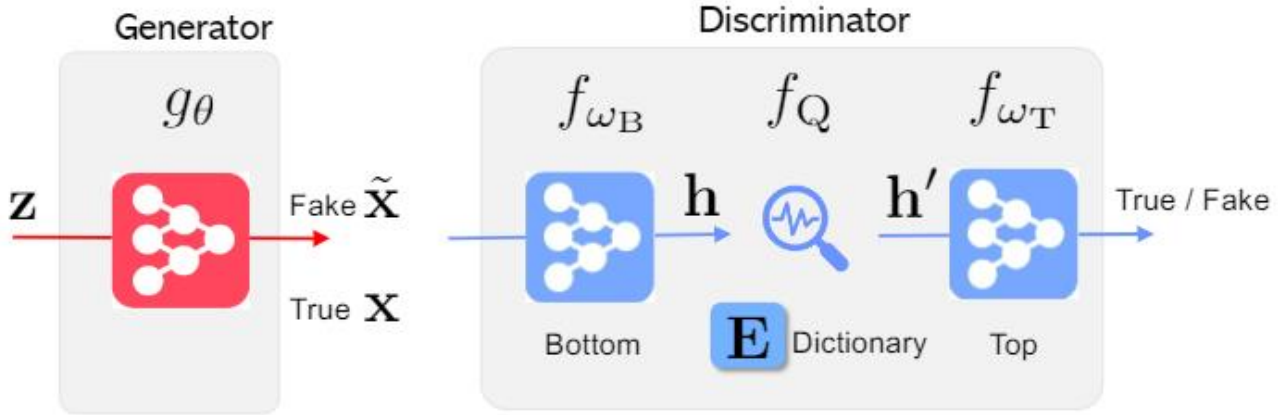
$$\mathbf{E} = \{\boldsymbol{e}_k \in \mathbb{R}^D \mid k \in 1, 2, \cdots, K\}.$$

Dictionary

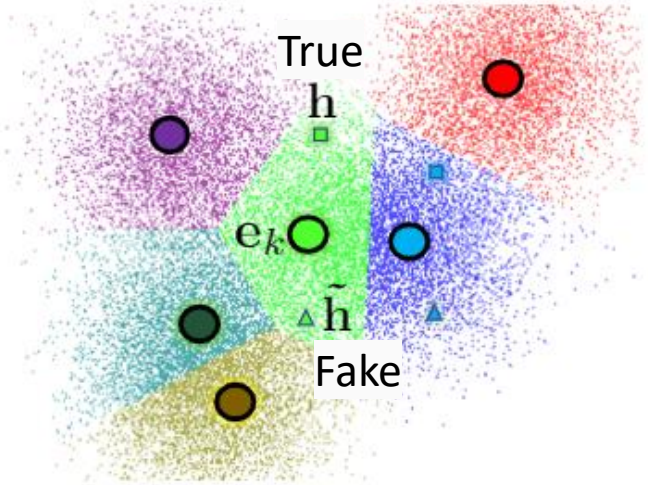
$$f_{\boldsymbol{\omega}}(\boldsymbol{x}) = f_{\boldsymbol{\omega}_T} \circ f_{\boldsymbol{\omega}_B}(\boldsymbol{x}) \longrightarrow f_{\boldsymbol{\omega}}(\boldsymbol{x}) = f_{\boldsymbol{\omega}_T} \circ f_{\mathrm{Q}} \circ f_{\boldsymbol{\omega}_B}(\boldsymbol{x}),$$

$$\boldsymbol{h}' = f_{\mathrm{Q}}(\boldsymbol{h}) = \boldsymbol{e}_k, \text{ where } k = \operatorname*{argmin}_j \|\boldsymbol{h} - \boldsymbol{e}_j\|_2$$

Quantization



(a) FQ-GAN architecture



(b) Dictionary look-up

Dictionary Learning

- Dictionary items are the feature centroids: a number of the most representative feature vectors

$$\mathcal{L}_{\mathrm{Q}} = \underbrace{\|\mathrm{sg}(\boldsymbol{h}) - \boldsymbol{e}_k\|_2^2}_{\text{dictionary loss}} + \beta \underbrace{\|\mathrm{sg}(\boldsymbol{e}_k) - \boldsymbol{h}\|_2^2}_{\text{commitment loss}}$$

sg: stop-gradient

- A dynamic & consistent dictionary:

$$\boldsymbol{e}_k \leftarrow \boldsymbol{m}_k / N_k, \text{ where } \boldsymbol{m}_k \leftarrow \lambda \boldsymbol{m}_k + (1-\lambda) \sum_{i=1}^{n_k} \boldsymbol{h}_{i,k},$$
$$N_k \leftarrow \lambda N_k + (1-\lambda) n_k, \qquad (8)$$

The current mini-batch is enqueued to the dictionary, and the oldest mini-batches in the queue are gradually removed. The dictionary always represents a set of prototypes for the recent features

FQ-GAN

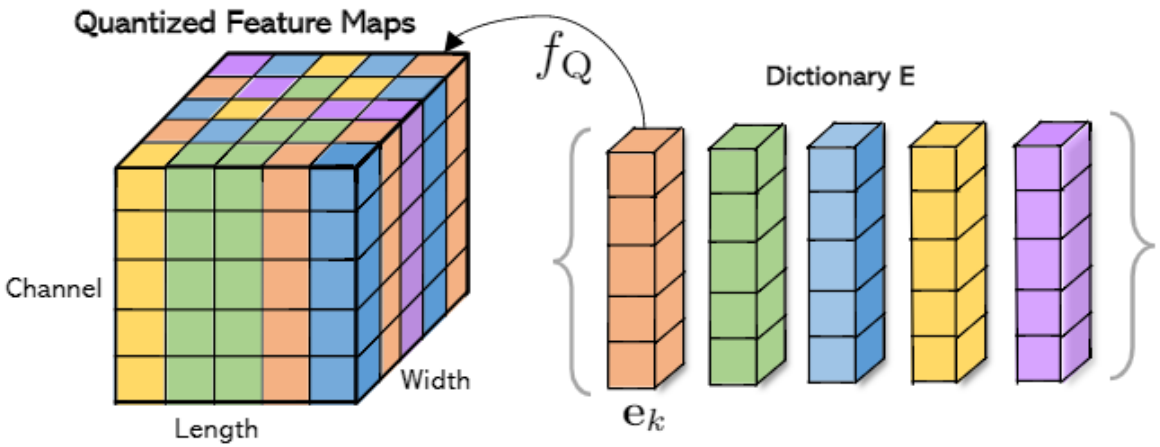
- Training objective:

$$\min_{\boldsymbol{\theta},\, \mathbf{E}} \max_{\boldsymbol{\omega}} \; \mathcal{L}_{\mathrm{FQ-GAN}} = \mathcal{L}_{\mathrm{GAN}} + \alpha \mathcal{L}_{\mathrm{Q}},$$

What's new?

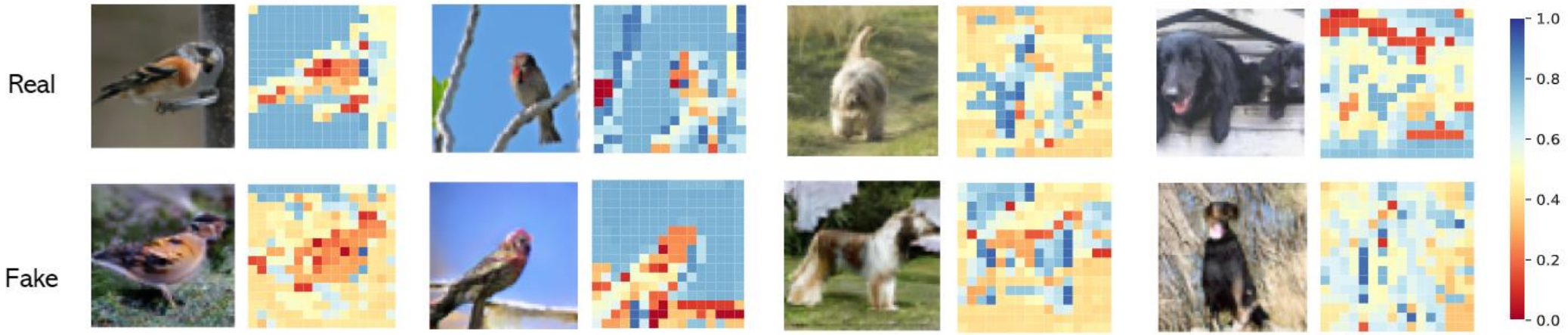
- Additional Parameters: Dictionary items
- Additional Regularizer: Feature quantization

- Applications to Image generation: position-wise quantization



At a given position on the feature map, the feature vector characterizes the local image region. It is quantized into its nearest dictionary item, leading to a new quantized feature map containing calibrated local feature prototypes

Quantized Feature Maps



The dictionary items are visualized in 1D as the color-bar using t-SNE.

Image regions with similar semantics utilize the same/similar dictionary items. For example, bird neck is in dark red, sky or clear background is in shallow blue, grass is in orange

Results (1/3): BigGANs for Image Generation

ajbrock / BigGAN-PyTorch | Watch 49 | Star 1.8k | Fork 265

Brock, et al. "Large scale GAN training for high fidelity natural image synthesis." *ICLR 2018*

Model	FID* ↓ / IS* ↑	FID ↓ / IS ↑
SN-GAN	14.26 / 8.22	–
R-MMD-GAN	16.21 / 8.29†	–
BigGAN	6.04 / 8.43	6.30±.20 / 8.31±.12
FQ-BigGAN	**5.34 / 8.50**	**5.59**±.12 / **8.48**±.03

Table 1. Comparison on CIFAR-10. †This number is quoted from (Wang et al., 2019)

Model	FID* ↓ / IS* ↑	FID ↓ / IS ↑
SN-GAN	16.77 / 7.01	–
TAC-GAN	7.22 / 9.34†	–
FQ-TAC-GAN	**7.15 / 9.74**	**7.21**±.10 / **9.69**±.04
BigGAN	8.64 / 9.46	9.01±.44 / 9.36±.10
FQ-BigGAN	**7.36 / 9.62**	**7.42**±.07 / **9.59**±.04

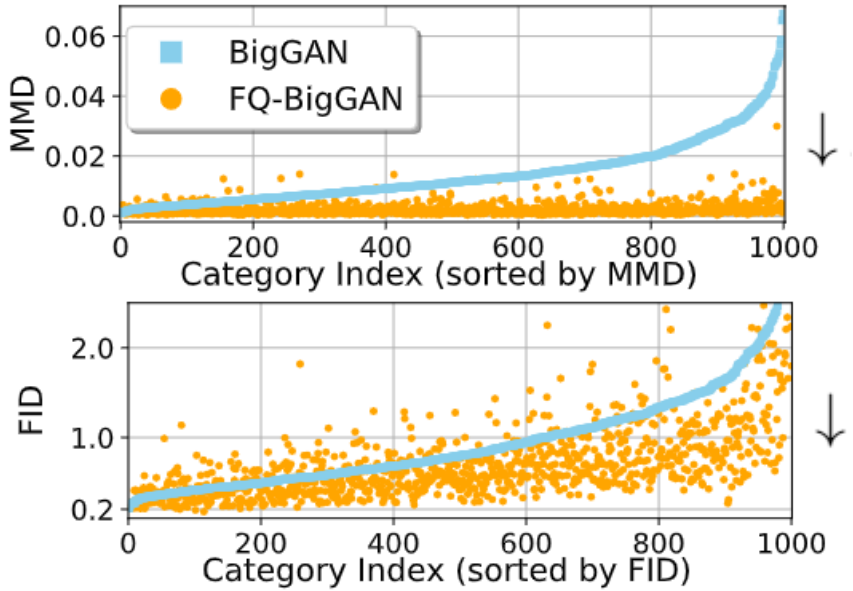
Table 2. Comparison on CIFAR-100. †This number is quoted from (Gong et al., 2019).

Models		64 × 64	128 × 128
		FID* ↓ / IS* ↑	FID* ↓ / IS* ↑
Half	TAC-GAN	-	23.75 / 28.86±0.29‡
	BigGAN	12.75 / 21.84±0.34	22.77 / 38.05±0.79‡
	FQ-BigGAN	**12.62 / 21.99**±0.32	**19.11 / 41.92**±1.15
256K	BigGAN	10.55 / 25.43±0.15	14.88 / 63.03±1.42†
	FQ-BigGAN	**9.67 / 25.96**±0.24	**14.08** / 54.36±1.07

Table 3. Comparison on ImageNet-1000 for two resolutions. Both models were trained for 256K iterations if not diverge early. The top and bottom block shows the best results within *half* and *full* of the entire training procedure, respectively. ‡ from (Gong et al., 2019), † from (Brock et al., 2018), we cannot reproduce it using their codebase, as the training diverges early.

TAC-GAN: "Twin Auxilary Classifiers GAN." NeurIPS 2019
M. Gong, Y. Xu, **C. Li**, K. Zhang, and K. B..

Results (1/3): BigGANs for Image Generation



Feature matching quality per class

Image generation quality per class

Model	ImageNet	CIFAR-100	CIFAR-10
BigGAN	7d16h	12h12m	17h37m
FQ-BigGAN	7d19h	12h35m	17h50m

Training time comparison; Only 1~3% slower

FQ-GAN significantly improves feature matching;
Improving GAN performance with minor computational overhead

Results (2/3): StyleGAN for Face Synthesis

Karras et al. “A Style-Based Generator Architecture for Generative Adversarial Networks”, CVPR 2019

Model	32×32	64×64	128×128
StyleGAN	3.28	4.82	6.33
FQ-StyleGAN	**3.01**	**4.36**	**5.98**

Table 5. StyleGAN: Best FID-50k scores in FFHQ at different resolutions.

Full resolution FFHQ (1024 x 1024).

StyleGAN2: 3.31
FQ-StyleGAN2: **3.19**

Results (3/3): Unsupervised Image-to-Image Translation

Kim et al. “U-GAT-IT: Unsupervised Generative Attentional Networks with Adaptive Layer-Instance Normalization for Image-to-Image Translation”, ICLR 2020

Model	selfie2anime	horse2zebra	cat2dog	photo2portrait	photo2vangogh
UNIT	14.71 ± 0.59	10.44 ± 0.67	8.15 ± 0.48	1.20 ± 0.31	4.26 ± 0.29
CycleGAN	13.08 ± 0.49	8.05 ± 0.72	8.92 ± 0.69	1.84 ± 0.34	5.46 ± 0.33
U-GAT-IT	11.61 ± 0.57	7.06 ± 0.8	7.07 ± 0.65	1.79 ± 0.34	4.28 ± 0.33
FQ-U-GAT-IT	**11.40 ± 0.28**	**2.93 ± 0.36**	**6.44 ± 0.35**	**1.09 ± 0.17**	6.54 ± 0.18

Model	anime2selfie	zebra2horse	dog2cat	portrait2photo	vangogh2photo
UNIT	26.32 ± 0.92	14.93 ± 0.75	9.81 ± 0.34	1.42 ± 0.24	9.72 ± 0.33
CycleGAN	11.84 ± 0.74	8.0 ± 0.66	9.94 ± 0.36	1.82 ± 0.36	4.68 ± 0.36
U-GAT-IT	11.52 ± 0.57	7.47 ± 0.71	8.15 ± 0.66	1.69 ± 0.53	5.61 ± 0.32
FQ-U-GAT-IT	**10.23 ± 0.40**	**7.10 ± 0.42**	8.90 ± 0.32	**0.73 ± 0.16**	5.21 ± 0.22

Table 6. KID ×100 for different image translation datasets. All numbers except for our FQ variant are from (Kim et al., 2020).

Model	baseline	FQ
selfie2anime	44.7	**55.3**
horse2zebra	36.2	**63.8**
cat2dog	34.0	**66.0**
photo2portrait	42.5	**57.5**
photo2vangogh	48.8	**51.2**

Table 7. User perceptual study on translated image preference (in percentage) between U-GAT-IT and its FQ variant using AMT.

③ Towards Learning a Generic Agent for Vision-and-Language Navigation via Pre-training

W. Hao∗, **C. Li**∗, X. Li, L. Carin and J. Gao (∗Equal contribution), CVPR 2020

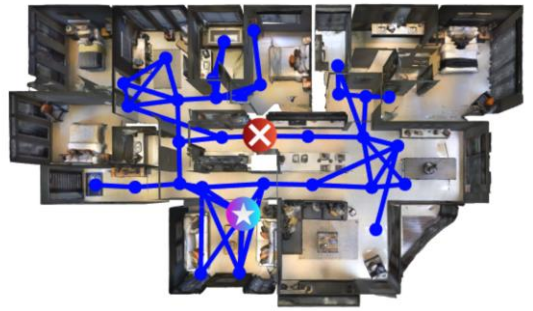
https://arxiv.org/abs/2002.10638

What is Vision-and-Language Navigation (VLN)?

- **Input**: language instructions $\mathbf{x}$; visual states $\mathbf{s_t}$ at each time step t
- **Output**: take an action $\mathbf{a_t}$ (which direction to navigate) each step
- **Goal**: From a starting location, train an agent to navigate to the target location

State is a panoramic view

$\mathbf{s_t}$



Failure case

Instruction is given as natural languages

$$\boldsymbol{x} = [x_1, x_2, ..., x_L] \Rightarrow \boldsymbol{\tau} = [\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \cdots, \boldsymbol{s}_T, \boldsymbol{a}_T].$$

Trajectory

Action is taken as one of discrete directions

Success case

A Generic Agent for Navigation Tasks

PREVALENT: PRE-TRAINED VISION-AND-LANGUAGE BASED NAVIGATOR

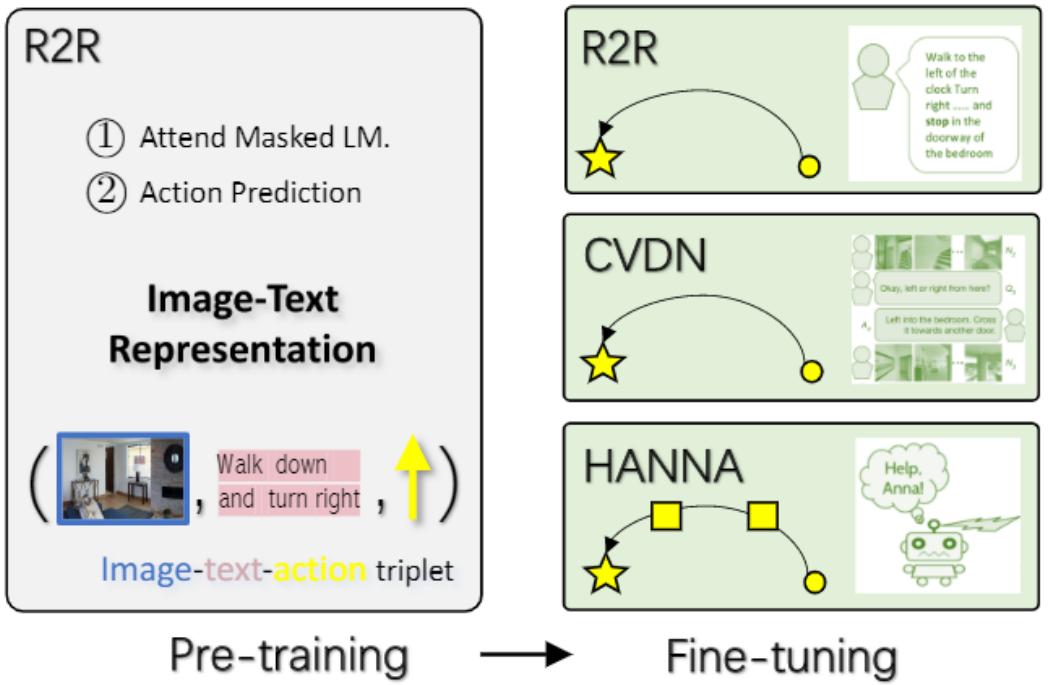


Figure 1: Illustration of the proposed pre-training and fine-tuning paradigm for VLN. The image-text-action triplets are collected from the R2R dataset. The model is pre-trained with two self-supervised learning objectives, and fine-tuned for three tasks: R2R, CVND and HANNA. R2R is an in-domain task, where the language instruction is given at the beginning, describing the full navigation path. CVND and HANNA are out-of-domain tasks; the former is to navigate based on dialog history, while the latter is an interactive environment, where intermediate instructions are given in the middle of navigation.

Pre-training dataset

We construct our pre-training dataset based on the Mat-terport3D Simulator

$\mathcal{D}_1$: The training datasets of R2R: **104K** image-text-action triplets

$\mathcal{D}_2$: We train an auto-regressive model (Speaker) on R2R, and employ the model to synthesize 1,020K instructions for the shortest-path trajectories on the Simulator: **6,482K** image-text-action triplets.

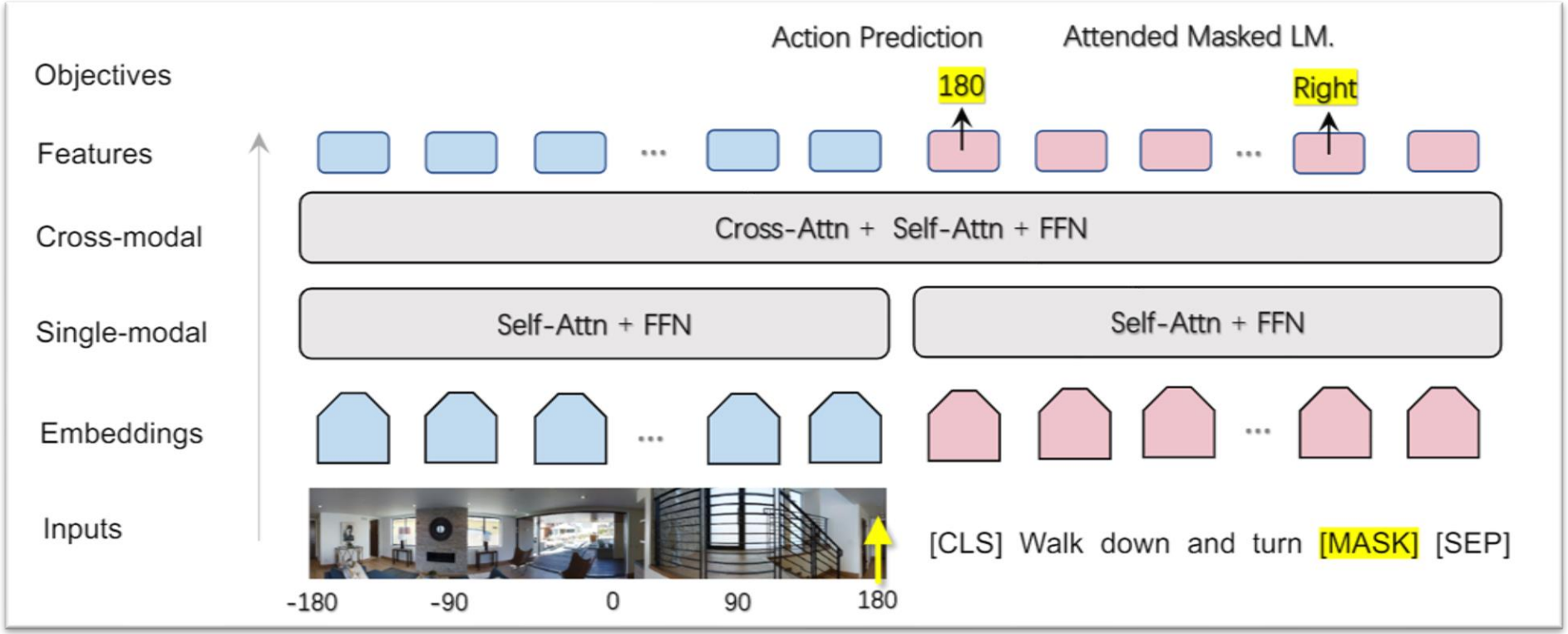
Therefore, the size of pre-training dataset $\mathcal{D} = \mathcal{D}_1 + \mathcal{D}_2$ is **6,582K**.

Pre-training

$$\mathcal{L}_{\text{Pre-training}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{AP}}$$

$$\mathcal{L}_{\text{MLM}} = -\mathbb{E}_{\boldsymbol{s}\sim p(\boldsymbol{\tau}),(\boldsymbol{\tau},\boldsymbol{x})\sim\mathcal{D}_E} \log p(x_i|\boldsymbol{x}_{\setminus i}, \boldsymbol{s})$$

$$\mathcal{L}_{\text{AP}} = -\mathbb{E}_{(\boldsymbol{a},\boldsymbol{s})\sim p(\boldsymbol{\tau}),(\boldsymbol{\tau},\boldsymbol{x})\sim\mathcal{D}_E} \log p(\boldsymbol{a}|x_{[\text{CLS}]}, \boldsymbol{s})$$



Fine-tuning

SoTA on three navigation tasks; serving a strong baseline for future self-supervised learning methods for VLN

	Agent	Validation Seen TL ↓	NE ↓	SR ↑	SPL ↑	Validation Unseen TL ↓	NE ↓	SR ↑	SPL ↑	Test Unseen TL ↓	NE ↓	SR ↑	SPL ↑
Greedy, **S**	RANDOM	9.58	9.45	16	-	9.77	9.23	16	-	9.93	9.77	13	12
	SEQ2SEQ	11.33	6.01	39	-	8.39	7.81	22	-	8.13	7.85	20	18
	RPA	-	5.56	43	-	-	7.65	25	-	9.15	7.53	25	23
	SPEAKER-FOLLOWER	-	3.36	66	-	-	6.62	35	-	14.82	6.62	35	28
	SMNA	-	-	-	-	-	-	-	-	18.04	5.67	48	35
	RCM+SIL(TRAIN)	10.65	3.53	67	-	11.46	6.09	43	-	11.97	6.12	43	38
	REGRETFUL	-	3.23	69	63	-	5.32	50	41	13.69	5.69	48	40
	FAST	-	-	-	-	21.17	4.97	56	43	22.08	5.14	54	41
	ENVDROP	11.00	3.99	62	59	10.70	5.22	52	48	11.66	5.23	51	47
	PRESS	10.57	4.39	58	55	10.36	5.28	49	45	10.77	5.49	49	45
	PREVALENT (ours)	10.32	3.67	69	**65**	10.19	**4.71**	**58**	**53**	10.51	5.30	**54**	**51**
M	PRESS	10.35	3.09	71	67	10.06	4.31	59	55	10.52	4.53	57	53
	PREVALENT	**10.31**	3.31	67	63	**9.98**	**4.12**	**60**	**57**	**10.21**	**4.52**	**59**	**56**
	Human	-	-	-	-	-	-	-	-	11.85	1.61	86	76

Table 1: Comparison with the state-of-the-art methods on R2R. Blue indicates the best value in a given setting. **S** indicates the single-instruction setting, **M** indicates the multiple-instruction setting.

Agent	Validation Unseen Oracle	Navigator	Mixed	Test Unseen Oracle	Navigator	Mixed
RANDOM	1.09	1.09	1.09	0.83	0.83	0.83
SEQ2SEQ	1.23	1.98	2.10	1.25	2.11	2.35
PREVALENT (Ours)	**2.58**	**2.99**	**3.15**	**1.67**	**2.39**	**2.44**
SHORTEST PATH AGENT	8.36	7.99	9.58	8.06	8.48	9.76

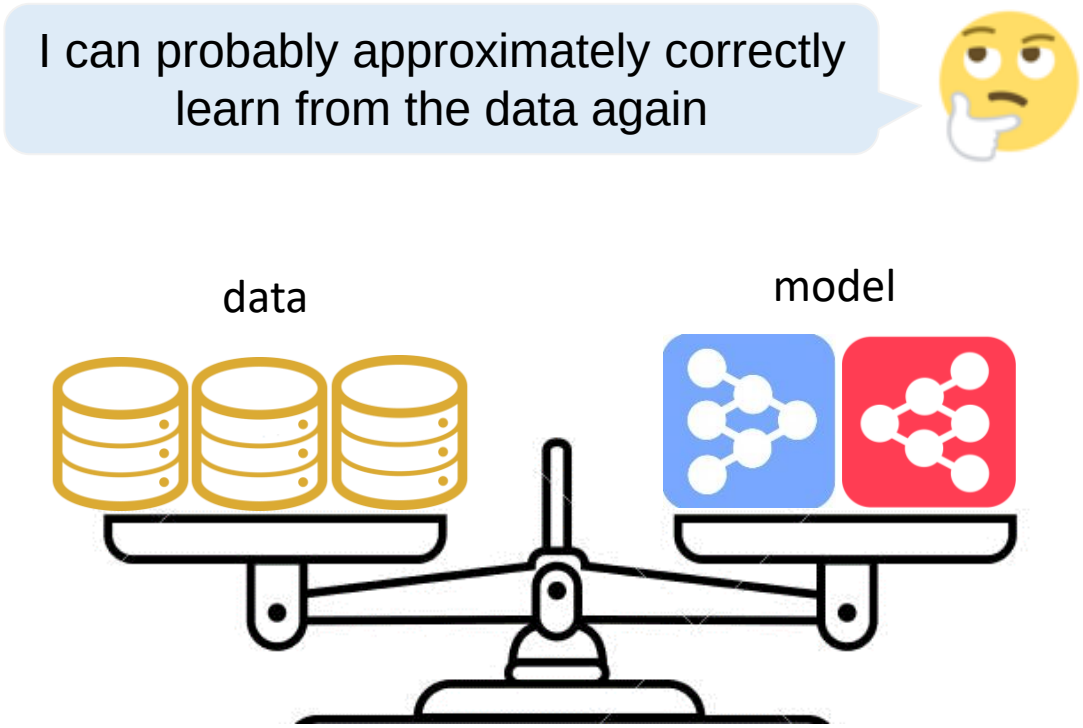
Table 2: Results on CVDN measured by Goal Progress. Blue indicates the best value in a given setting.

	Agent	SEEN-ENV SR ↑	SPL ↑	NE ↓	#R ↓	UNSEEN-ALL SR ↑	SPL ↑	NE ↓	#R ↓
Rule	RANDOM WALK	0.54	0.33	15.38	0.0	0.46	0.23	15.34	0.0
	FORWARD 10	5.98	4.19	14.61	0.0	6.36	4.78	13.81	0.0
	NO ASSISTANCE	17.21	13.76	11.48	0.0	8.10	4.23	13.22	0.0
	ANNA	88.37	63.92	1.33	2.9	47.45	25.50	7.67	5.8
	PREVALENT (Ours)	83.82	59.38	1.47	3.4	**52.91**	**28.72**	**5.29**	6.6
Skyline	SHORTEST	100.00	100.00	0.00	0.0	100.00	100.00	0.00	0.0
	Perfect assistance	90.99	68.87	0.91	2.5	83.56	56.88	1.83	3.2

Table 3: Results on test splits of HANNA. The agent with "perfect assistance" uses the teacher navigation policy to make decisions when executing a subtask from the assistant. Blue indicates the best value.

Conclusions

Model Types	At a large scale	Key Comments
VAE	**OPTIMUS** -- Opportunities in Language Modeling	The first pre-trained VAE model in comparison with BERT & GPT-2
GAN	**FQ-GAN** -- Challenges in Image Generation	Stabilizing mini-batch estimates for large datasets
Autoregressive models	**PREVALENT**: -- Applications to Vision-and-Language Navigation	Generating samples to augment datasets for pre-training



https://en.wikipedia.org/wiki/Probably_approximately_correct_learning