NIPS 2017

Trends and Interesting Papers

René Donner, rene@contextflow.com

contextflow

Overview

Self-Normalizing Neural Networks

Meta Learning

- Self-play
- Population based Training of Neural Networks

GANs

- PacGANs

Differentiable Computing

- The Case for Learned Index Structures

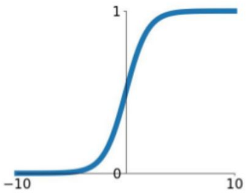
Self-Normalizing Neural Networks

Self-Normalizing Neural Networks

Activation Functions

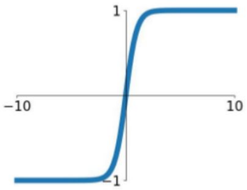
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



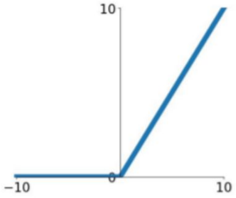
tanh

$\tanh(x)$



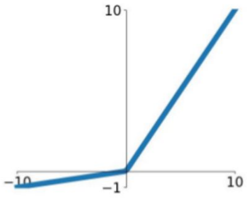
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

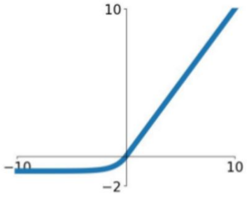


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



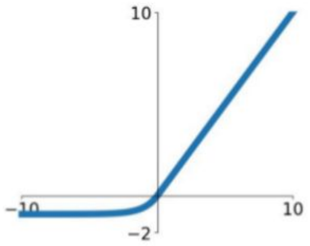
https://medium.com/machine-learning-world/how-to-debug-neural-networks-manual-dc2a200f10f2

contextflow

Self-Normalizing Neural Networks

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Scaled exponential linear units:

$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{if } x \leqslant 0 \end{cases}$$

https://arxiv.org/abs/1706.02515

Self-Normalizing Neural Networks

method	#layers / #blocks 2	3	4	6	8	16	32
SNN	*83.7* ± 0.3	**84.4** ± 0.5	**84.2** ± 0.4	**83.9** ± 0.5	**84.5** ± 0.2	**83.5** ± 0.5	**82.5** ± 0.7
Batchnorm	80.0 ± 0.5	79.8 ± 1.6	77.2 ± 1.1	77.0 ± 1.7	75.0 ± 0.9	73.7 ± 2.0	76.0 ± 1.1
WeightNorm	*83.7* ± 0.8	82.9 ± 0.8	82.2 ± 0.9	*82.5* ± 0.6	*81.9* ± 1.2	78.1 ± 1.3	56.6 ± 2.6
LayerNorm	**84.3** ± 0.3	*84.3* ± 0.5	*84.0* ± 0.2	*82.5* ± 0.8	80.9 ± 1.8	78.7 ± 2.3	78.8 ± 0.8
Highway	83.3 ± 0.9	83.0 ± 0.5	82.6 ± 0.9	82.4 ± 0.8	80.3 ± 1.4	80.3 ± 2.4	79.6 ± 0.8
MSRAinit	82.7 ± 0.4	81.6 ± 0.9	81.1 ± 1.7	80.6 ± 0.6	80.9 ± 1.1	80.2 ± 1.1	80.4 ± 1.9
ResNet	82.2 ± 1.1	80.0 ± 2.0	80.5 ± 1.2	81.2 ± 0.7	81.8 ± 0.6	*81.2* ± 0.6	na

Overview

Self-Normalizing Neural Networks

Meta Learning

- Self-play
- Population based Training of Neural Networks

GANs

- PacGANs

Differentiable Computing

- The Case for Learned Index Structures

Meta Learning – Self Play

AlphaGo / AlphaZero

Meta Learning Workshop @ NIPS

https://nips.cc/Conferences/2017/Schedule?showEvent=8767

Self Play

learn complex behaviours given a simple objective

initially: rewards for behaviours like standing / moving forward

later: reward only for winning and losing

https://www.youtube.com/watch?v=OBcjhp4KSgQ

https://blog.openai.com/competitive-self-play/

Population Based Training of Neural Networks

Optimize model and hyperparameters

Fixed computational budget

Very easy to implement

https://arxiv.org/abs/1711.09846

Hyperparameter optimization

Sequential

Manual tweaking

Bayesian models

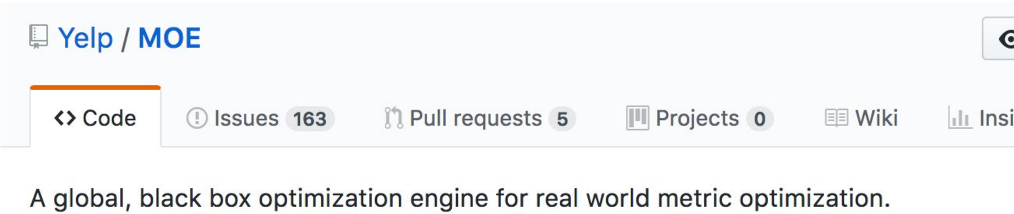
Parallel

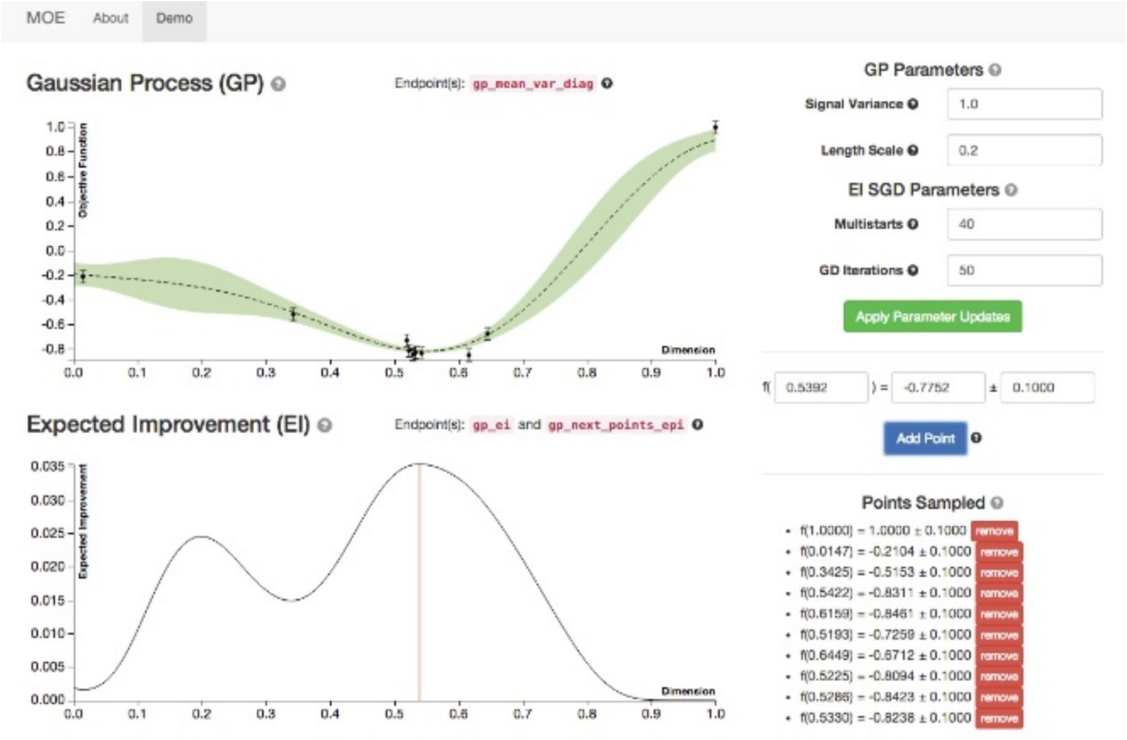
Grid search

Random search

https://arxiv.org/abs/1711.09846

Bayesian models





https://github.com/Yelp/MOE

Hyperparameter optimization

Sequential

- Manual tweaking
- Bayesian models

Parallel

- Grid search
- Random search

https://arxiv.org/abs/1711.09846

Key ideas

Run population in parallel

But only a few steps

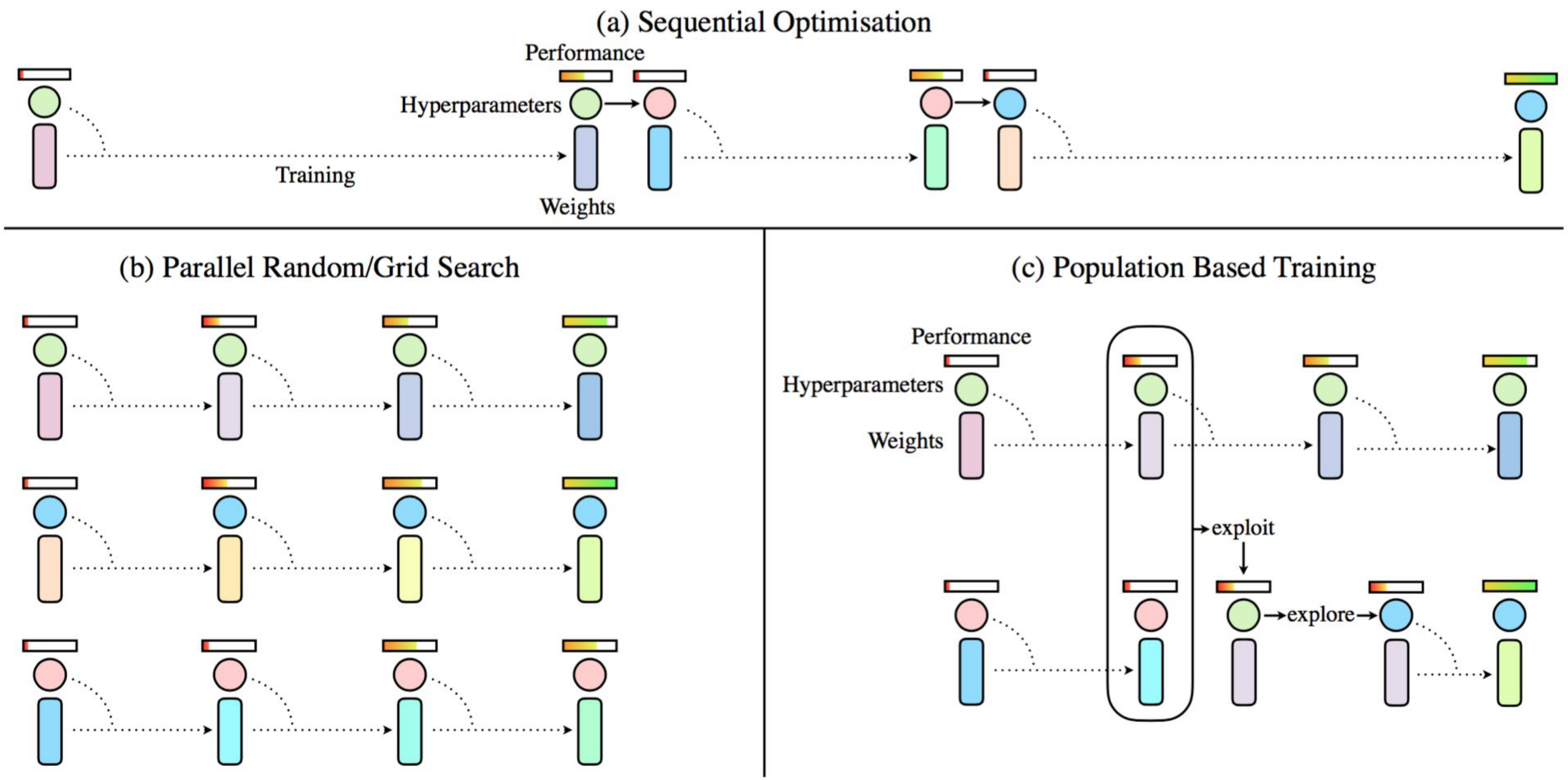
Use actual error measure for evaluation

Keep best, kill worst, mutate others

Evolutionary search in hyperparameter space

https://arxiv.org/abs/1711.09846

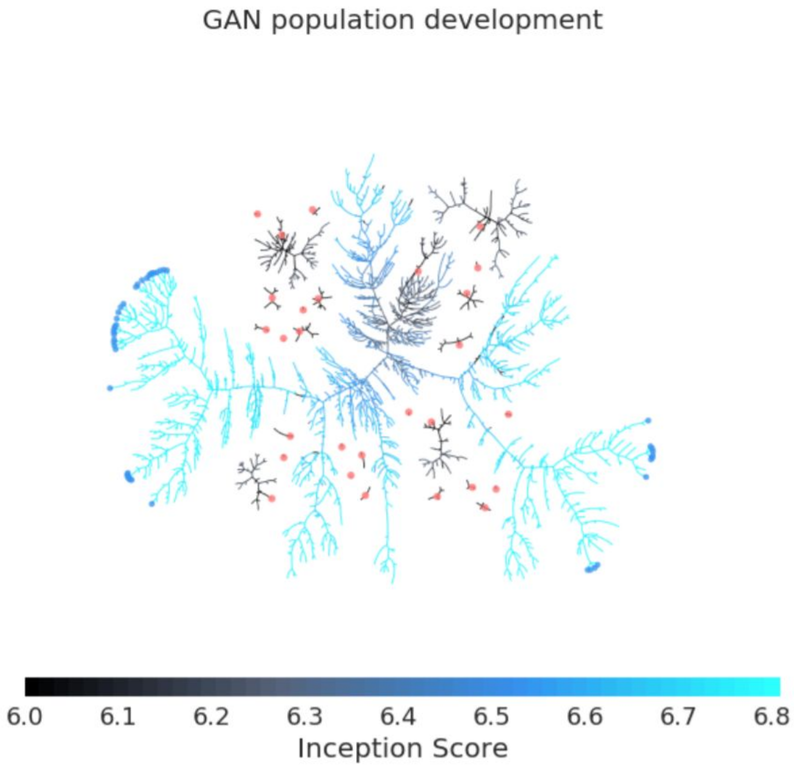
Key ideas

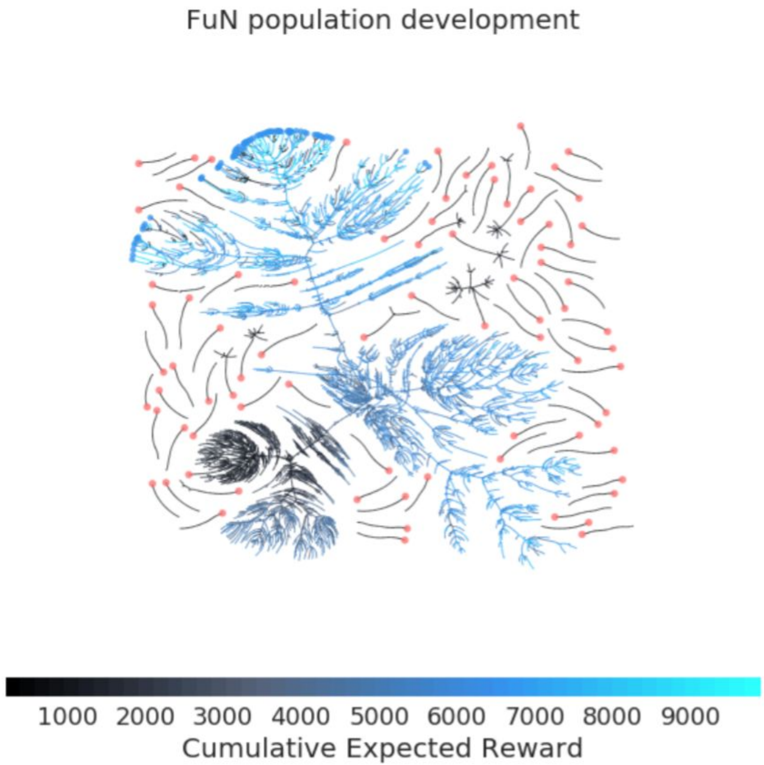


https://arxiv.org/abs/1711.09846

contextflow

Evolution of hyper-parameters

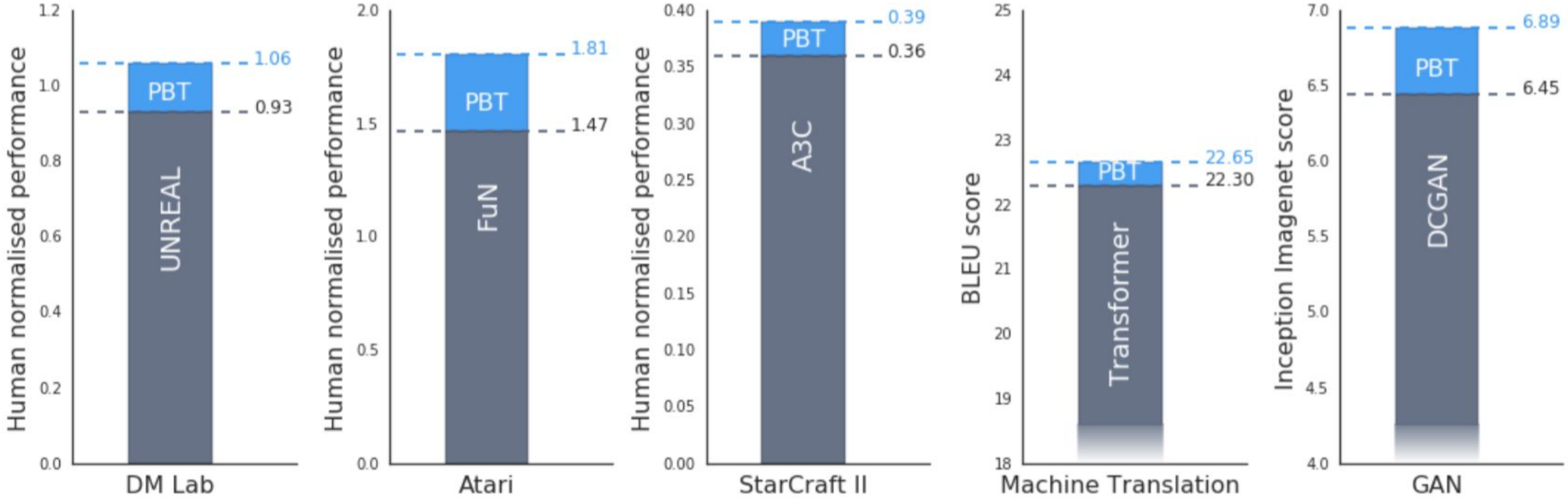




https://arxiv.org/abs/1711.09846

contextflow

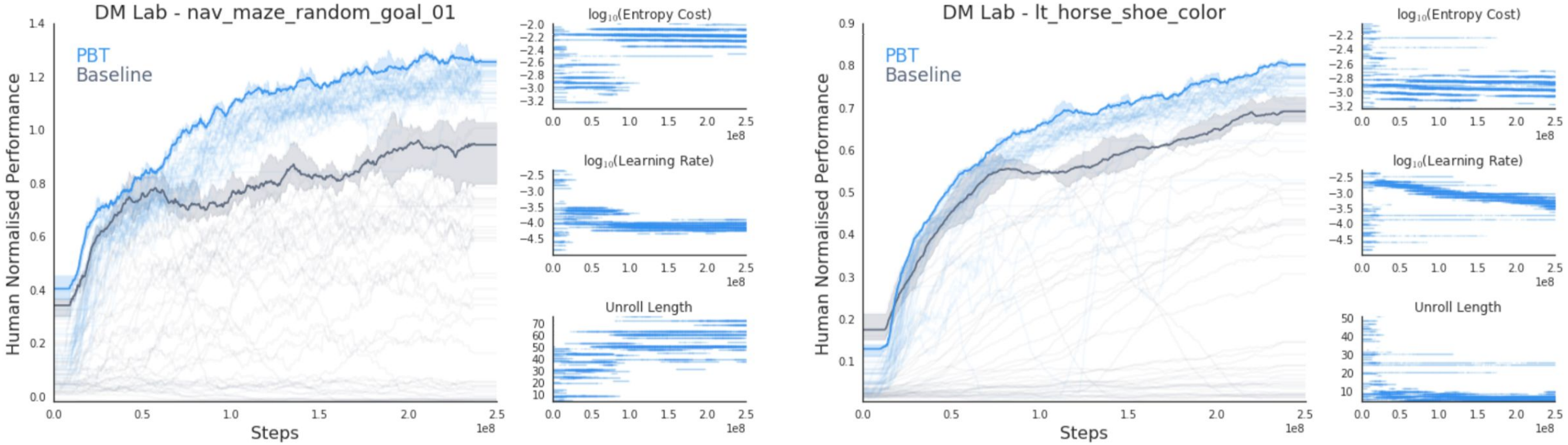
Results on RNN, GANs, Machine Translation



https://arxiv.org/abs/1711.09846

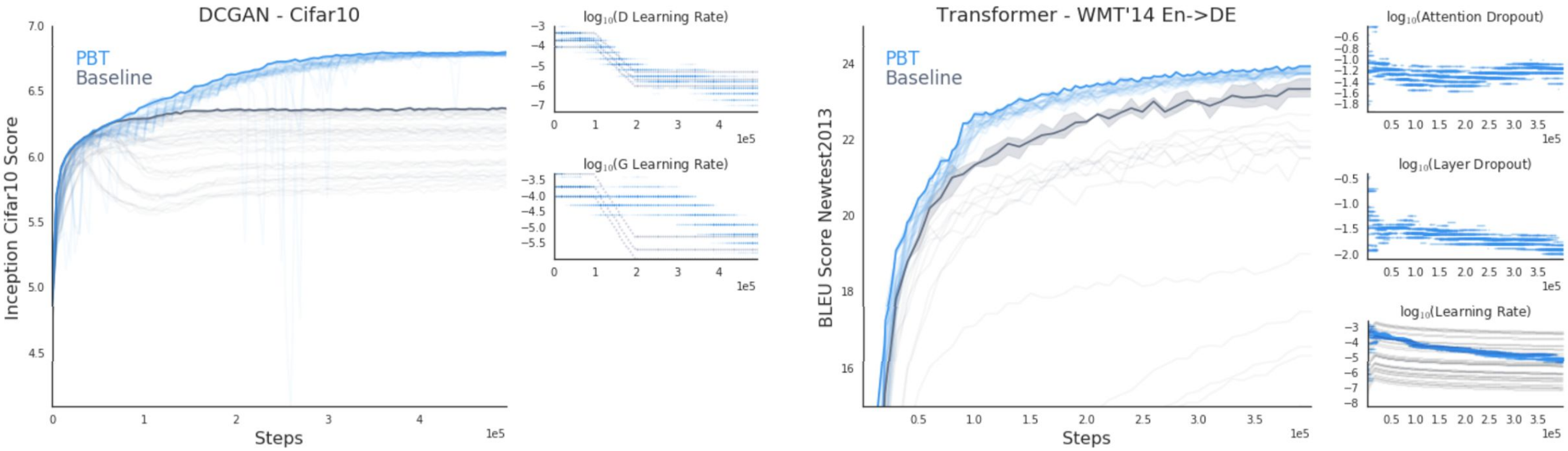
contextflow

Vs Random Search in Reinforcement



https://arxiv.org/abs/1711.09846

Vs Random Search for GANs



https://arxiv.org/abs/1711.09846

contextflow

Contributions

Automatic selection of hyperparameters

Online model selection
maximise use of computation spent on promising models

Enable non-stationary training regimes

Discovery of complex hyperparameter schedules

https://arxiv.org/abs/1711.09846

Overview

Self-Normalizing Neural Networks

Meta Learning

- Self-play
- Population based Training of Neural Networks

GANs

- PacGANs

Differentiable Computing

- The Case for Learned Index Structures

Generative Adversarial Models

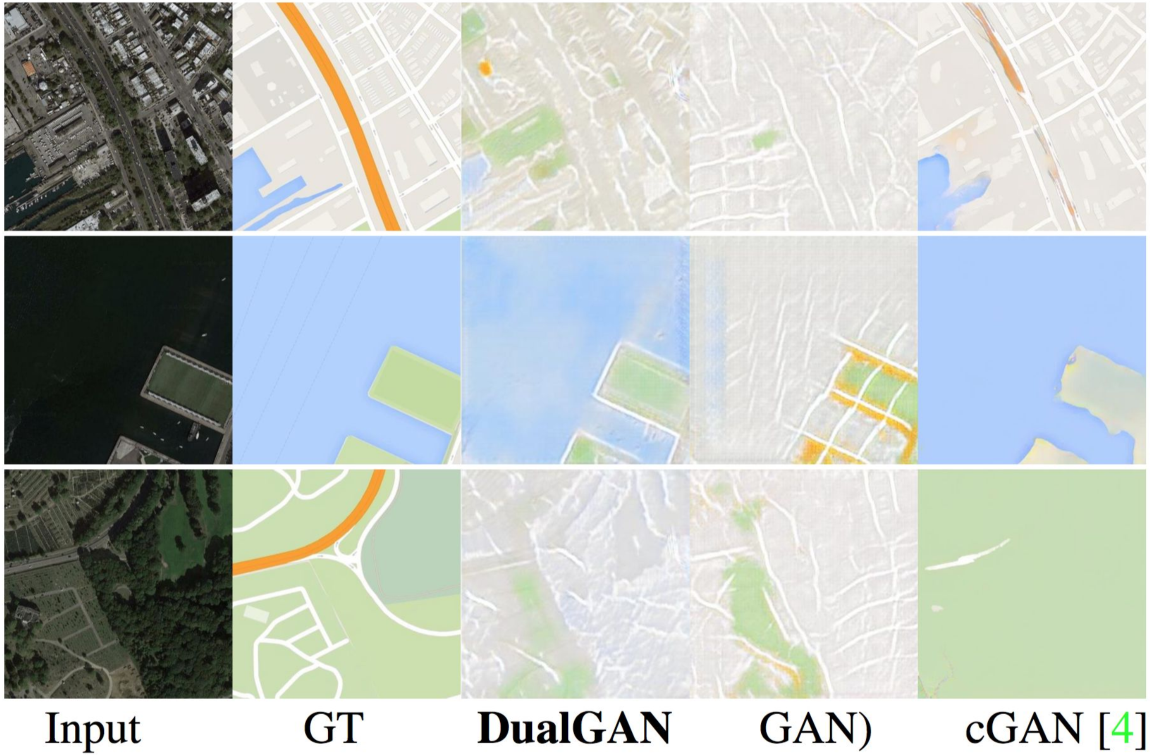
Hype slowing down (a little ;-)

Focus on domain translation

Focus on avoiding mode collapse

- e.g VEEGAN
- PacGAN

Domain translation



https://arxiv.org/abs/1704.02510

Domain translation

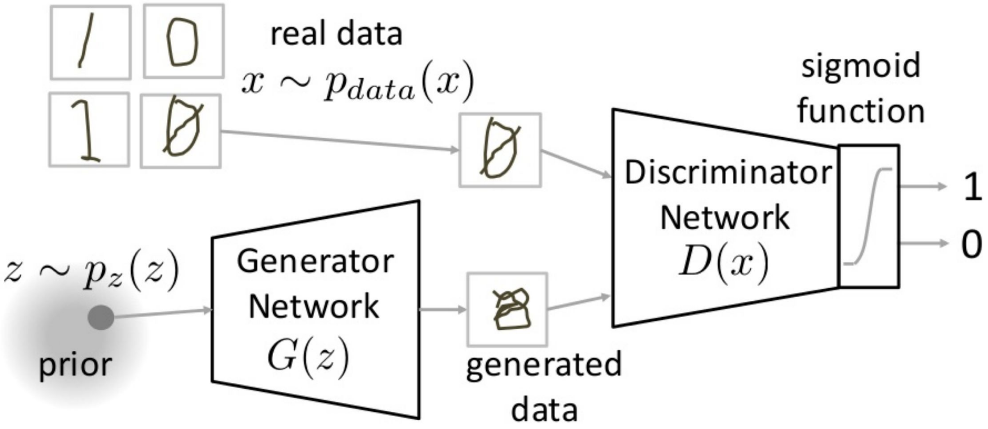
https://arxiv.org/abs/1704.02510

PacGAN

Generative Adversarial Networks

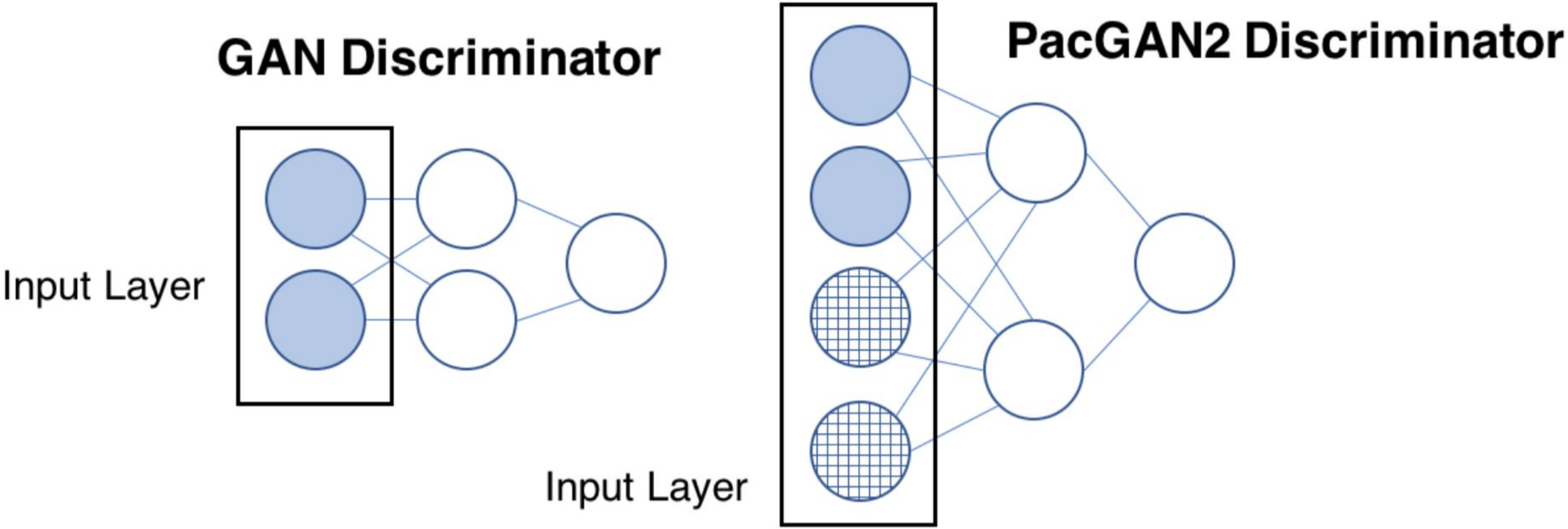
$$\min_G \max_D V(D, G)$$

$$V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$



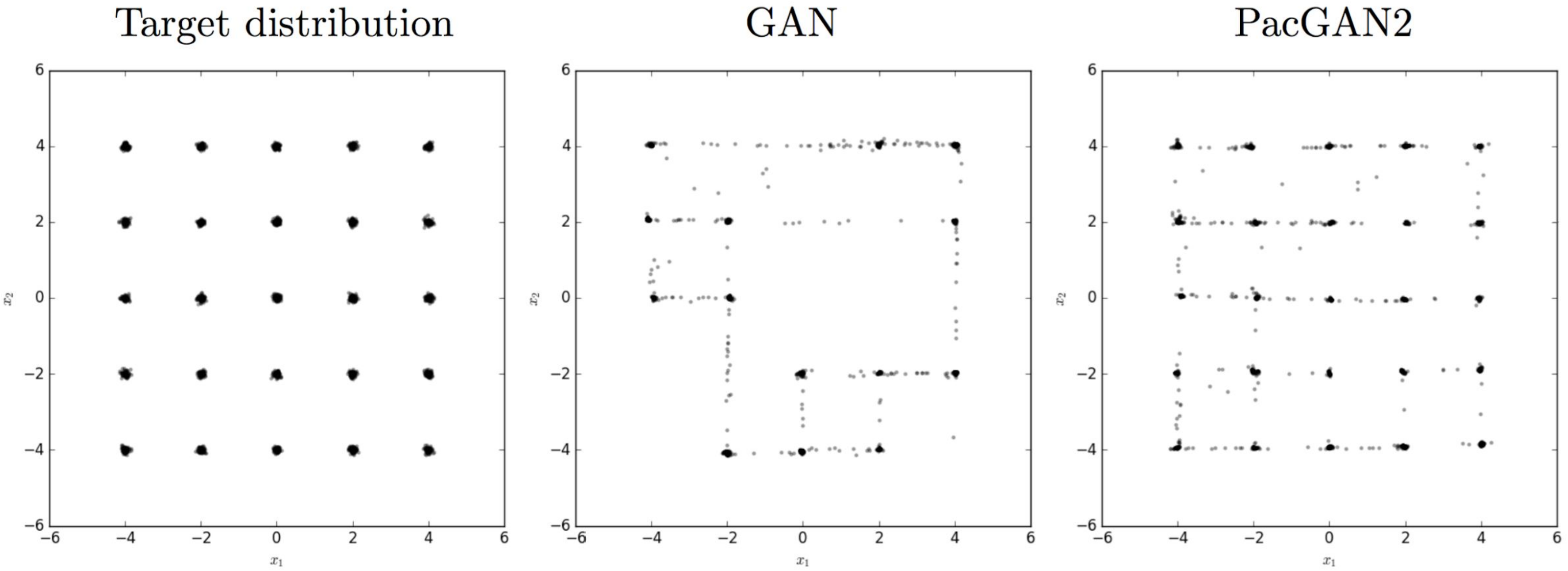
https://www.slideshare.net/ckmarkohchang/generative-adversarial-networks

PacGAN



https://arxiv.org/abs/1712.04086

PacGAN

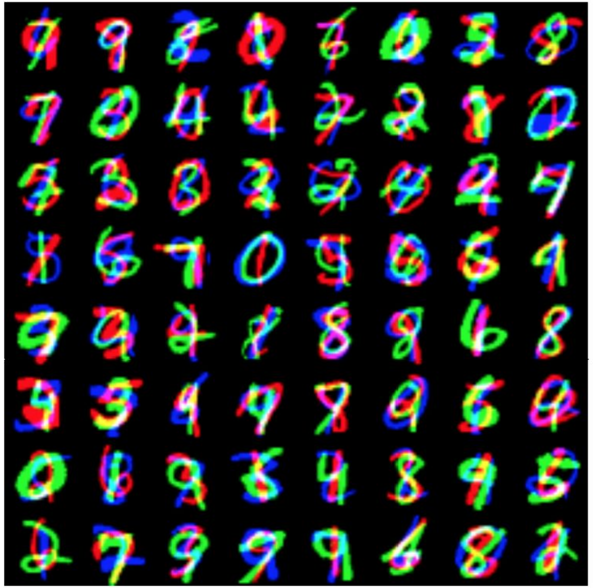


https://arxiv.org/abs/1712.04086

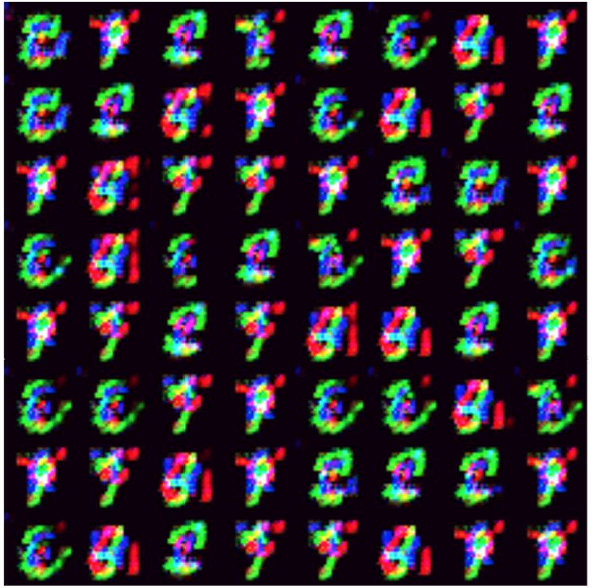
contextflow

PacGAN

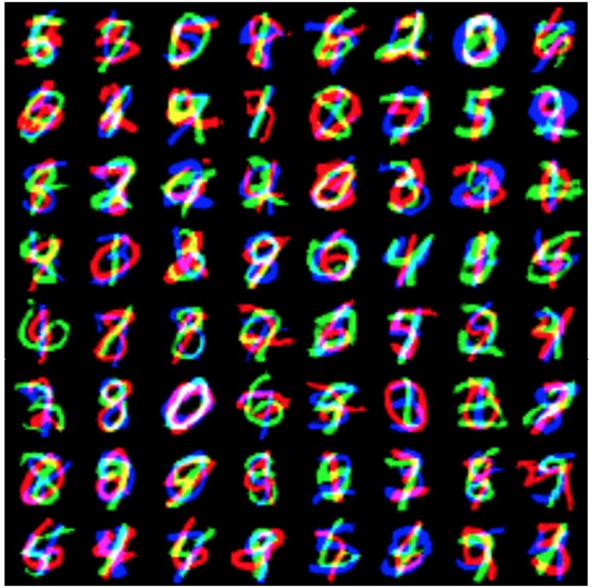
Target distribution



DCGAN



PacDCGAN2



https://arxiv.org/abs/1712.04086

contextflow

Overview

Self-Normalizing Neural Networks

Meta Learning

- Self-play
- Population based Training of Neural Networks

GANs

- PacGANs

Differentiable Computing

- The Case for Learned Index Structures

The Case for Learned Index Structures

Index Structures

B-Trees

Hash-Maps

Bloom Filters

Aim

Learn more compact / faster data structures!

https://arxiv.org/abs/1712.01208

The Case for Learned Index Structures

Index Structures

- B-Trees
- Hash-Maps
- Bloom Filters

The Case for Learned Index Structures

Tim Kraska*
MIT
Cambridge, MA
kraska@mit.edu

Alex Beutel
Google, Inc.
Mountain View, CA
alexbeutel@google.com

Ed
Goog
Mountai
edchi@g

Jeffrey Dean
Google, Inc.
Mountain View, CA
jeff@google.com

Neoklis Polyzotis
Google, Inc.
Mountain View, CA
npolyzotis@google.com

Aim

Learn more compact / faster data structures!

Motivation

Index Structures are Models

B-Trees -> regression

Hash-Maps -> classification

Bloom Filters -> classification

Future Performance

CPU: Moore’s law is dead

GPUs / TPUs

Worst-case data distribution vs task specific data distribution

Learn data structure best suited for actual data!

Motivation

Index Structures are Models

B-Trees -> regression

Hash-Maps -> classification

Bloom Filters -> classification

Future Performance

CPU: Moore's law is dead

GPUs / TPUs

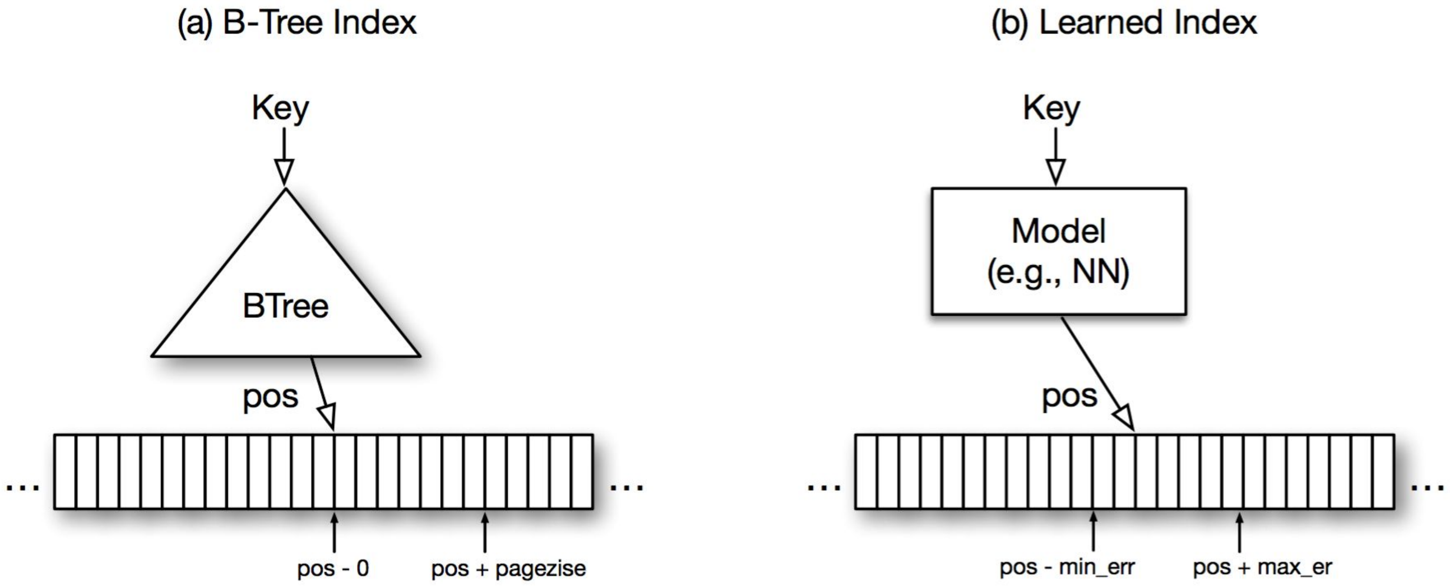
Worst-case data distribution vs task specific data distribution

Learn data structure best suited for actual data!

B-Trees

Task

Given key/datum, predict location in sorted index

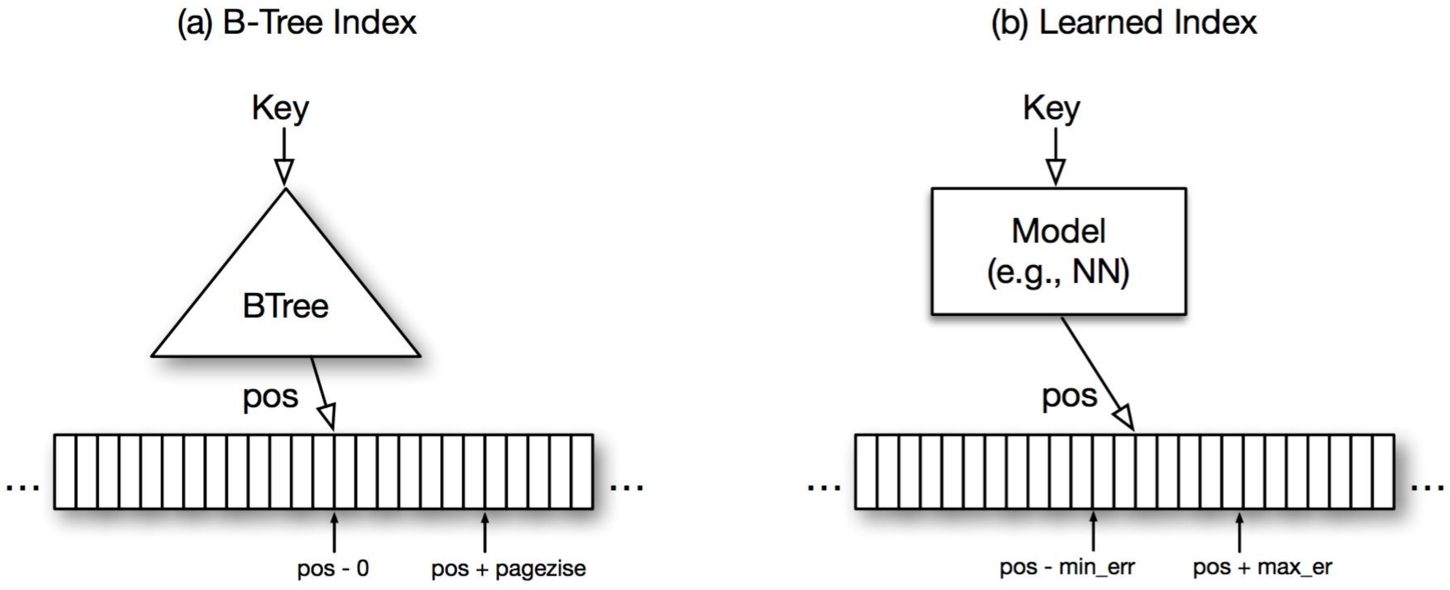


contextflow

B-Trees

Task

Given key/datum, predict location in sorted index



contextflow

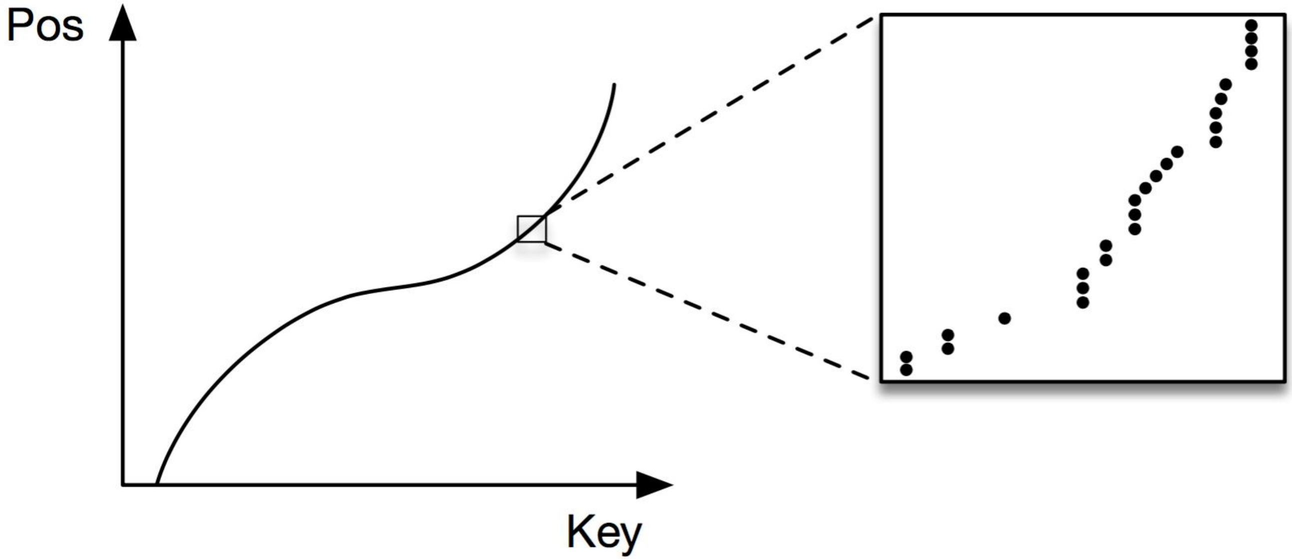
B-Trees

Task

Given key/datum, predict location in sorted index

B-Trees

Indices as cumulative distribution functions



B-Trees

Naive approach

Two layer FNN, 32 nodes

Slow due to Tensorflow overhead

Difficulty modelling fine details

Learning Index Framework

Custom C++ framework for small networks

Recursive Model Index

Recursive Model Index

Takes idea from mixture of experts

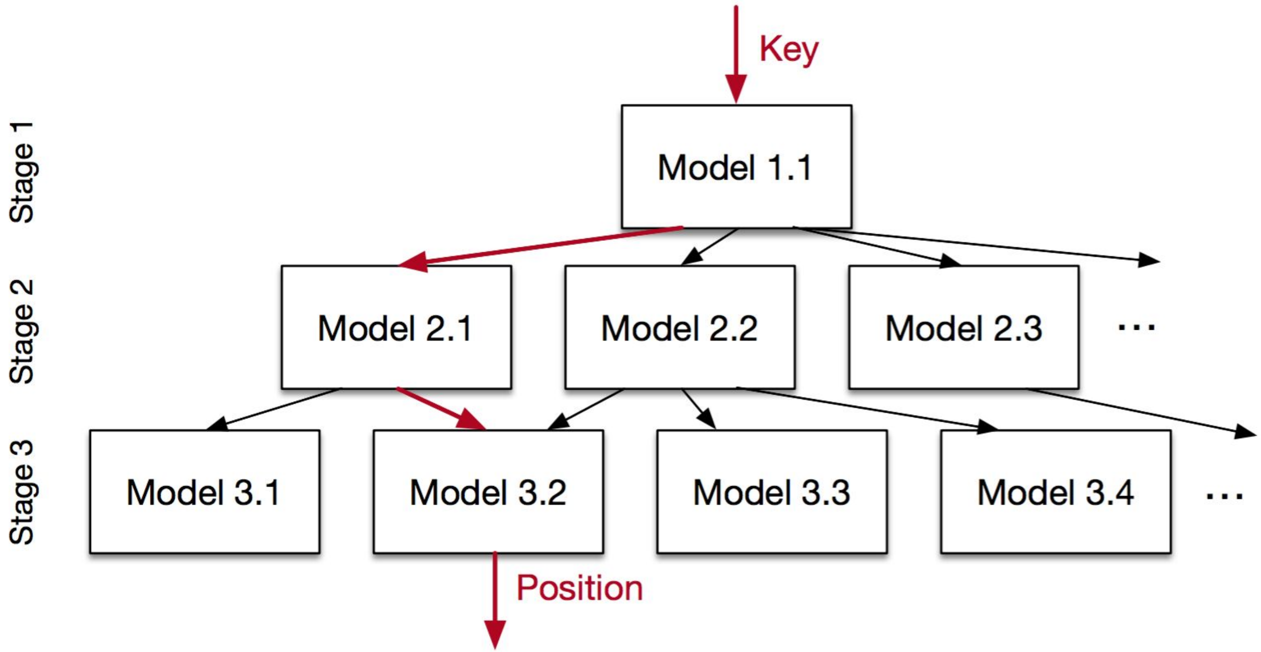


Figure 3: Staged models

Recursive Model Index

Train stage by stage

Predict all data - create new temp data sets

Last stage predicts position

Is not a tree

Hybrid model with B-Tree

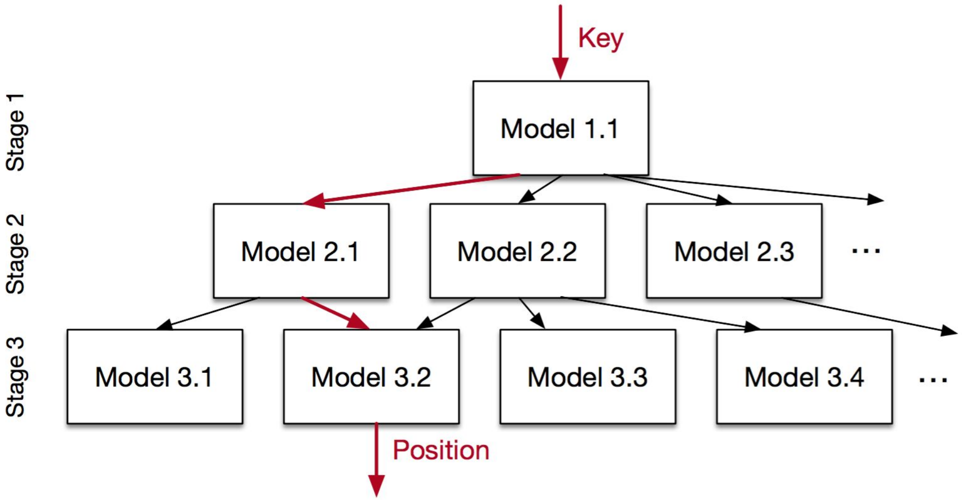


Figure 3: Staged models

Results

Type	Config	Search	Total (ns)	Model (ns)	Search (ns)	Speedup	Size (MB)	Size Savings	Model Err ± Err Var.
Btree	page size: 16	Binary	280	229	51	6%	104.91	700%	4 ± 0
	page size: 32	Binary	274	198	76	4%	52.45	300%	16 ± 0
	page size: 64	Binary	277	172	105	5%	26.23	100%	32 ± 0
	page size: 128	Binary	265	134	130	0%	13.11	0%	64 ± 0
	page size: 256	Binary	267	114	153	1%	6.56	-50%	128 ± 0
Learned Index	2nd stage size: 10,000	Binary	98	31	67	-63%	0.15	-99%	8 ± 45
		Quaternary	101	31	70	-62%	0.15	-99%	8 ± 45
	2nd stage size: 50,000	Binary	85	39	46	-68%	0.76	-94%	3 ± 36
		Quaternary	93	38	55	-65%	0.76	-94%	3 ± 36
	2nd stage size: 100,000	Binary	82	41	41	-69%	1.53	-88%	2 ± 36
		Quaternary	91	41	50	-66%	1.53	-88%	2 ± 36
	2nd stage size: 200,000	Binary	86	50	36	-68%	3.05	-77%	2 ± 36
		Quaternary	95	49	46	-64%	3.05	-77%	2 ± 36
Learned Index Complex	2nd stage size: 100,000	Binary	157	116	41	-41%	1.53	-88%	2 ± 30
		Quaternary	161	111	50	-39%	1.53	-88%	2 ± 30

Figure 4: Map data: Learned Index vs B-Tree

Hash-Maps

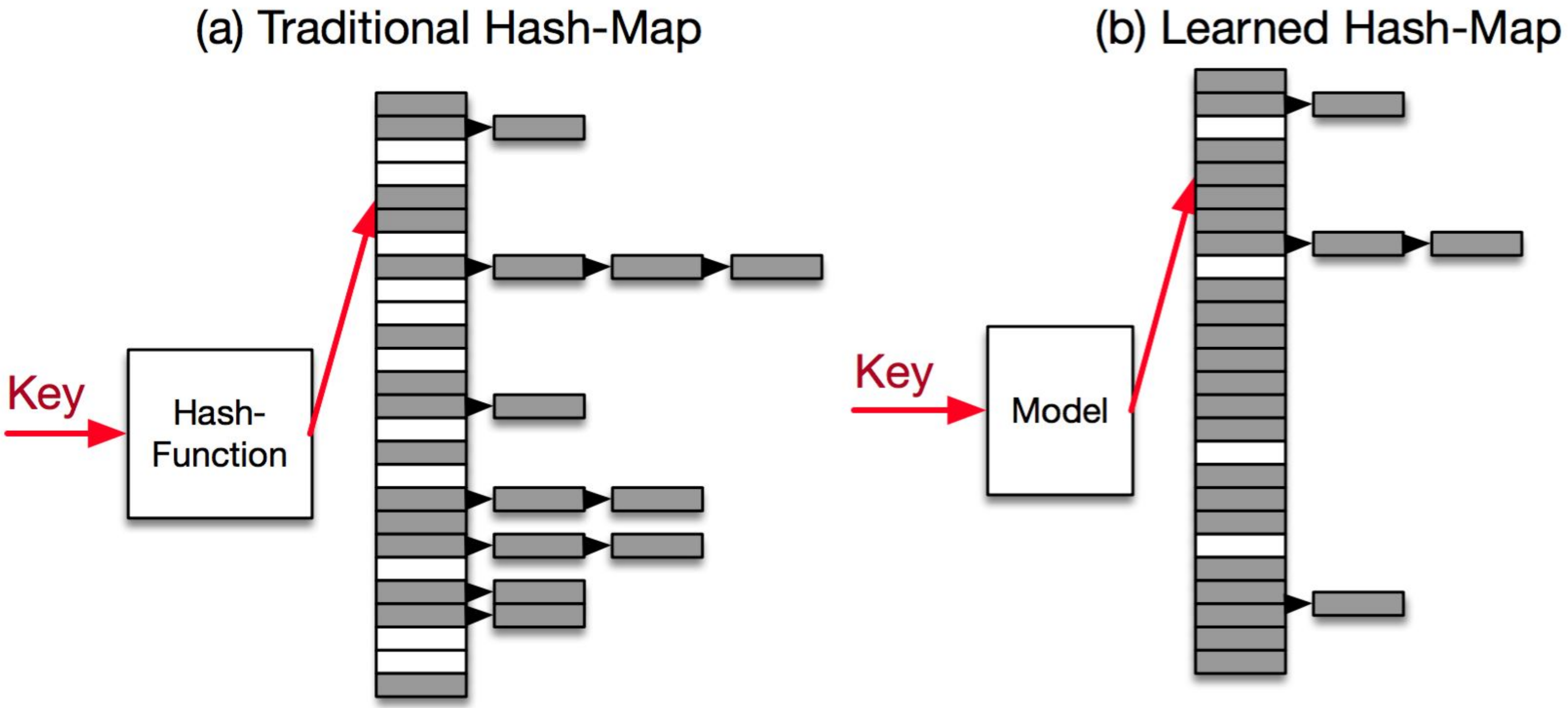


Figure 9: Traditional Hash-map vs Learned Hash-map

Hash-Maps

Dataset	Slots	Hash Type	Search Time (ns)	Empty Slots	Space Improvement
Map	75%	Model Hash	67	0.63GB (05%)	-20%
		Random Hash	52	0.80GB (25%)	
	100%	Model Hash	53	1.10GB (08%)	-27%
		Random Hash	48	1.50GB (35%)	
	125%	Model Hash	64	2.16GB (26%)	-6%
		Random Hash	49	2.31GB (43%)	
Web Log	75%	Model Hash	78	0.18GB (19%)	-78%
		Random Hash	53	0.84GB (25%)	
	100%	Model Hash	63	0.35GB (25%)	-78%
		Random Hash	50	1.58GB (35%)	
	125%	Model Hash	77	1.47GB (40%)	-39%
		Random Hash	50	2.43GB (43%)	
Log Normal	75%	Model Hash	79	0.63GB (20%)	-22%
		Random Hash	52	0.80GB (25%)	
	100%	Model Hash	66	1.10GB (26%)	-30%
		Random Hash	46	1.50GB (35%)	
	125%	Model Hash	77	2.16GB (41%)	-9%
		Random Hash	46	2.31GB (44%)	

Figure 10: Model vs Random Hash-map

Overview

Self-Normalizing Neural Networks

Meta Learning

- Self-play
- Population based Training of Neural Networks

GANs

- PacGANs

Differentiable Computing

- The Case for Learned Index Structures

image analysis
machine learning
artificial intelligence

contextflow

spinoff of the Medical University of Vienna

exploration of large-scale medical imaging data

René Donner, rene@contextflow.com