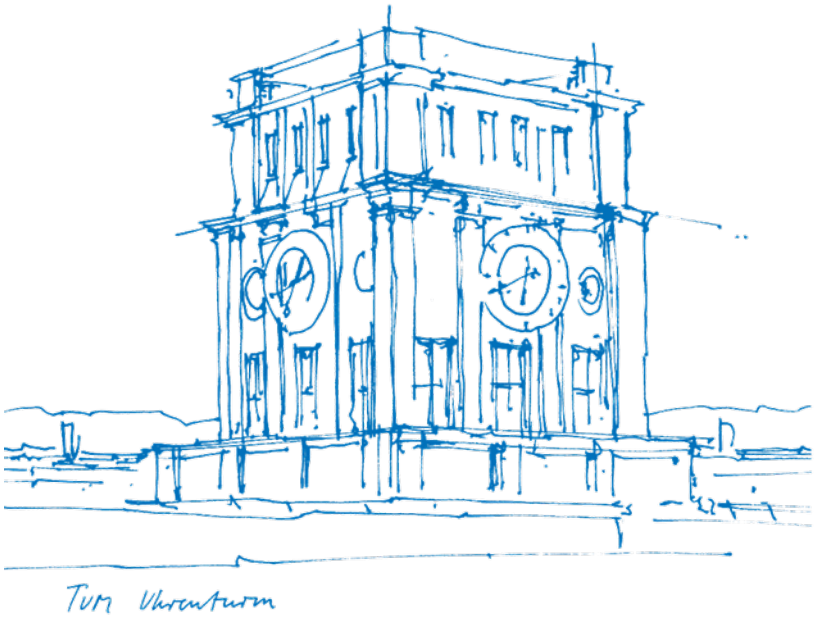
Computer Vision & Artificial Intelligence
Department of Informatics
Technical University of Munich

Deep learning: a non-alchemical view

Yuesong Shen

Computer Vision & Artificial Intelligence
Technical University of Munich
July 22, 2020



Outline

- Deep learning: the original series
- Deep learning: the next generation?
- Neural networks as chain graphs
  - Core ideas
  - Generality of the CG interpretation
  - Selected case studies of NN designs
- Conclusion and discussions

Deep learning: the original series

[1]

[1] From https://www.cbc.ca/news/technology/turing-award-ai-deep-learning-1.5070415

The deep learning boom

Success stories of DL on ImageNet:

- **AlexNet**[2] (2012): significantly outperforms other ML methods
  $\rightsquigarrow$ Deep CNN (8 layers), max-pooling, ReLU, LRN, dropout
- **ResNet**[3] (2015): reaches human-level
  $\rightsquigarrow$ Very deep CNN (100+ layers), Residual block, BatchNorm

NNs are widely used, achieving SOTA results in many domains:

Computer vision *Natural language processing* *Reinforcement learning*
Generative models *Graph learning* *Uncertainty estimation*

Many cool works from our chair use DL!

[2] Krizhevsky et al. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*. 2012, 1097–1105.
[3] He et al. "Deep Residual Learning for Image Recognition". In: *CVPR*. 2016.

Mystical arts of deep learning at a glance

We have an **overwhelming** amount of NN designs and practices:

- **Architecture**: ResNet, Inception, (bi-)RNN, Siamese, Auto-encoder, GAN ...
- **Layer**: Convolution, Pooling, LSTM, Attention, GCN, GAT, skip connection ...
- **Activation**: (leaky) ReLU, Sigmoid, tanh, softmax, maxout, ELU ...
- **Loss**: L1, L2, Hinge, Cross Entropy, NLL, KL Div., artisanal heuristics ...
- **Optimizer**: SGD, momemtum, Adagrad, Adam, RMSProp, AMSGrad ...
- **Initialization**: Xavier, Kaiming, Fixup, ReZero, Isometric ...
- **Regularization**: weight decay, dropout, Batch/Layer/Group/Weight Norm, gradient clipping, DropConnect ...
- And many, many, many more (input pre-processing, lr scheduling ...)!

The alchemy age of deep learning

As Ali Rahimi pointed out in the NIPS 2017 “Test of Time” talk:

“Machine learning has become alchemy.”

[5]

We have scalable model, but non-scalable progress!

We need the **“science”** of deep learning

[5] From https://medium.com/@Synced/lecun-vs-rahimi-has-machine-learning-become-alchemy-21cb1557920d

Deep learning: the next generation?

[6]

[6] Adapted from http://yvettecandraw.blogspot.com/2011/09/speaking-of-outer-spacethe-top-ten.html

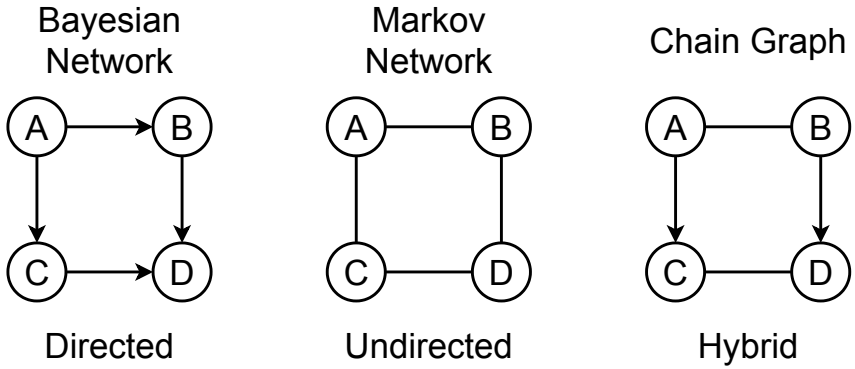
Going beyond the deep learning alchemy

- DL does have a theory: Universal approximation theorem
  $\rightsquigarrow$ Establish repr. power of NNs, Little insight, still "black-box".
- Several prior works aim to provide theoretical insights to NNs:
  - Gaussian process, kernel theory: NN as GP[7], NTK[8]
  - Continuous counterpart: Neural ODE[9]
  - Optimal transport for generative models[10]
  - Signal propagation[11], Lottery ticket hypothesis[12] ...
- **Our approach tries to make the link between PGM and DL.**

[7] Lee et al. "Deep Neural Networks as Gaussian Processes". In: *ICLR*. 2018.
[8] Jacot et al. "Neural Tangent Kernel: Convergence and Generalization in Neural Networks". In: *NeurIPS*. 2018.
[9] Chen et al. "Neural Ordinary Differential Equations". In: *NeurIPS*. 2018, 6572–6583.
[10] Genevay et al. "GAN and VAE from an optimal transport point of view". In: *arXiv:1706.01807* (2017).
[11] Poole et al. "Exponential expressivity in deep neural networks through transient chaos". In: *NIPS*. 2016.
[12] Frankle and Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *ICLR*. 2019.

Probabilistic graphical models: a new hope?

- **Representation**: graphical description of RVs and their relations
  Graph $\leftrightarrow$ Overall distrib. Node $\leftrightarrow$ Random var. Edge $\leftrightarrow$ Direct influence
- Many PGM variants:



$$
\begin{aligned}
P(A,B,C,D) &= P(A)P(B|A)P(C|A)P(D|B,C) && \text{(BN)}\\
&= \frac{1}{Z}\phi(A,B)\phi(A,C)\phi(B,D)\phi(C,D) && \text{(MN)}\\
&= P(A,B)P(C,D|A,B) && \text{(CG)}
\end{aligned}
$$

- Flexible modeling: generative, unsupervised, domain-knowledge ...

The intractability strikes back

How efficient are **inference** and **learning** on general PGM?

$\rightsquigarrow$ Bad news: exact inference is proven to be intractable![13]

How about approximations?

$\rightsquigarrow$ Bad news: Any with better-than-random guarantee is intractable![13]

Don't panic. All is not lost!

- Tractable for special cases: tree, binary submodular ...
- Many *(somewhat) practical* approx. inference / learning methods:
  - $\rightsquigarrow$ Mean field, Loopy belief propagation, Gibbs sampling ...
  - $\rightsquigarrow$ Pseudo-likelihood, Contrastive divergence ...

PGM offers flexible modeling, but is hard to do scalable learning!

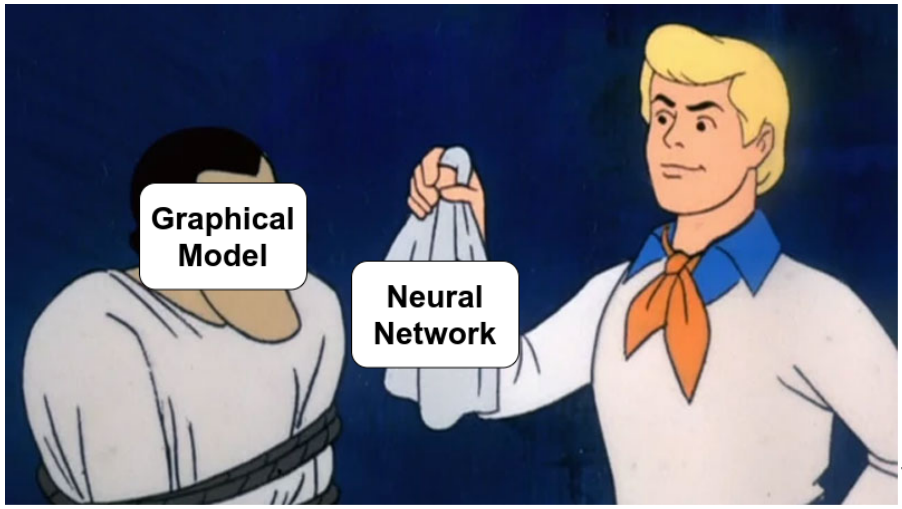
[13] Koller and Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

PGM and NN: a tale of two models

- **PGM**: elegant theory, sub-optimal scalability
- **NN**: strong practical performance, weak theoretical support

⇝ Would be nice to have the best of both worlds!

Shocking revelation: NN can actually be interpreted as PGM!

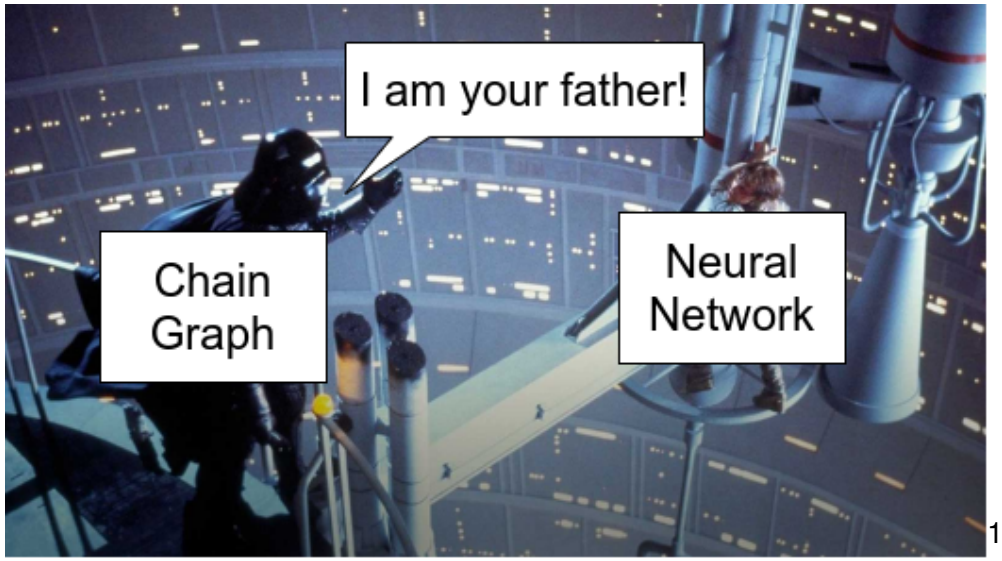


[14]

[14] Adapted from https://knowyourmeme.com/memes/lets-see-who-this-really-is

Neural networks as chain graphs

A.K.A.: Deriving Neural Network Design and Learning from the Probabilistic Framework of Chain Graphs



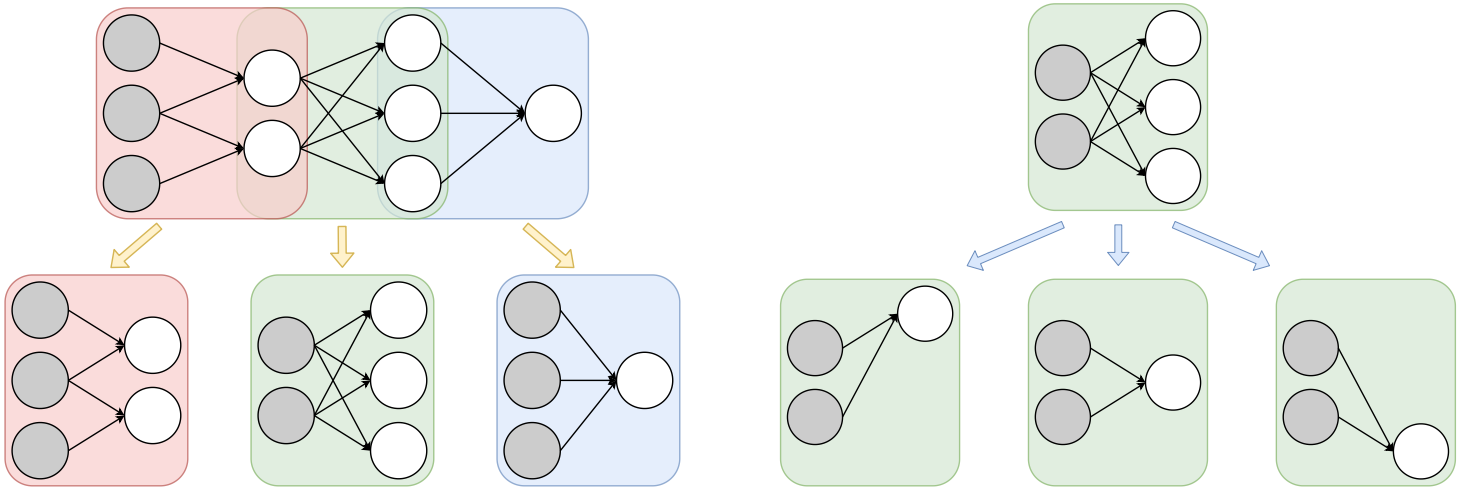
[15] Adapted from http://www.dearcastandcrew.com/content/2015/12/10/how-to-avoid-the-star-wars-syndrome.html

Outline

- Deep learning: the original series
- Deep learning: the next generation?
- Neural networks as chain graphs
  - Core ideas
  - Generality of the CG interpretation
  - Selected case studies of NN designs
- Conclusion and discussions

From a binary sequential layered CG ...

- L layers of binary RVs $X = (X^1, \ldots, X^L)$, $X_i^l \in \{\alpha^l, \beta^l\}$.



- Chain graph: $P(X^2, \ldots, X^L | X^1) = \prod_{l=2}^{L} P(X^l | X^{l-1})$;
- Bipartite chain component: $P(X^l | X^{l-1}) = \prod_{i=1}^{N^l} P(X_i^l | X^{l-1})$;
- Pairwise CRF: $P(X_i^l | X^{l-1}) = \frac{1}{Z_i^l(X^{l-1})} \exp\left(b_i^l X_i^l + \sum_{j=1}^{N^{l-1}} w_{j,i}^l X_j^{l-1} X_i^l\right)$;
- Binary RV: $P(X_i^l | X^{l-1}) = \sigma\left((2X_i^l - (\alpha^l + \beta^l))(b_i^l + (X^{l-1})^\top w_{\cdot,i}^l)\right)$.

... To a neural network!

Given an input $\tilde{x}^1$, we denote for each RV node X_i^l:

- Marginal distribution: $Q_i^l(x_i^l|\tilde{x}^1) = P(X_i^l = x_i^l|X^1 = \tilde{x}^1)$;
- Expected value: $q_i^l = \mathbb{E}_{Q_i^l}[X_i^l]$ $(q^1 = \tilde{x}^1)$.

We have the following recursive expression for Q
$Q_i^l(x_i^l|\tilde{x}^1) = \sum_{x^{l-1} \in \{\alpha^{l-1}, \beta^{l-1}\}^{N^{l-1}}} P(x_i^l|x^{l-1}) Q^{l-1}(x^{l-1}|\tilde{x}^1) = \mathbb{E}_{Q^{l-1}}[P(x_i^l|X^{l-1})]$.

$\rightsquigarrow$ Looks intractable ... How to proceed?

Do a linear approximation! $\rightsquigarrow$ Move the expectation inside.

$$q_i^l = \sum_{x_i^l \in \{\alpha^l, \beta^l\}} x_i^l \mathbb{E}_{Q^{l-1}}[P(x_i^l|X^{l-1})] \approx \frac{\beta^l - \alpha^l}{2} \tanh\left(\frac{\beta^l - \alpha^l}{2}\left(b_i^l + (q^{l-1})^\top w_{:,i}^l\right)\right) + \frac{\alpha^l + \beta^l}{2}.$$

Hey, this is just a feed-forward with q!

$$\alpha^l = 0, \beta^l = 1 \Longrightarrow \text{Sigmoid}, \alpha^l = -1, \beta^l = 1 \Longrightarrow \text{Tanh}.$$

Summarizing the core ideas

Overview of some correspondences:

NN	Layer	Neuron	Feed-forward	Non-linearity
CG	Chain component	RV node	Approx. inference	RV distrib.

Learning?

$\rightsquigarrow$ Same! Loss via MLE / SRM + SGD + Backprop.

When is the approximation accurate?

- Small pairwise weight
  $\rightsquigarrow$ weight decay is good!
- Linear neighborhood
  $\rightsquigarrow$ might explain why ReLU is good.

Clear modeling, assumptions and approximations!

Outline

- Deep learning: the original series
- Deep learning: the next generation?
- Neural networks as chain graphs
  - Core ideas
  - Generality of the CG interpretation
  - Selected case studies of NN designs
- Conclusion and discussions

Let's generalize ...

- **Network architecture**: general (Partially) Directed Acyclic Graph
  - *Multi-branch*: Inception, ResNet
  - *Time series*: RNNs
- **Layer connection**: sparse connection, shared / fixed weight
  - *NN layers*: dense, convolution, average pooling, skip connection ...
  - *Intra-connection*: non-bipartite CRF as the last layer[16][17][18]
- **Modular components**: transfer learning via layer reuse.

$\rightsquigarrow$ How about activation functions?

[16] Zheng et al. "Conditional Random Fields as Recurrent Neural Networks". In: *ICCV*. 2015.

[17] Chen et al. "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs". In: *TPAMI* 40.4 (2018), pp. 834–848.

[18] Lample et al. "Neural Architectures for Named Entity Recognition". In: *NACCL*. 2016.

General form of activation function

- We define statistics T^l that extract features from each node X_i^l.
- General nodewise CPD: (positive unit-sum) generalized linear model

$$P(X_i^l | X^{l-1}) = f_i^l\left(X_i^l, e_i^l(T^{l-1}(X^{l-1}))\right)$$

with

$$e_i^l(T^{l-1}(X^{l-1})) = b_i^l + \sum_{j=1}^{N^{l-1}} w_{j,i}^l \; T^{l-1}(X_j^{l-1}).$$

- General feed-forward expression:

$$q_i^l = \mathbb{E}_{Q_i^l}[T^l(X_i^l)] \approx \int_{x_i^l} T^l(x_i^l) f_i^l\left(x_i^l, e_i^l(q^{l-1})\right) dx_i^l.$$

- Example: multi-label node with label indicator statistics $\Longrightarrow$ softmax.
- Batch/Layer/Weight/Group Norm: reparametrization of e, as part of f.

Rectified Gaussian distribution family

The question to address: **Can we get ReLU from this?**

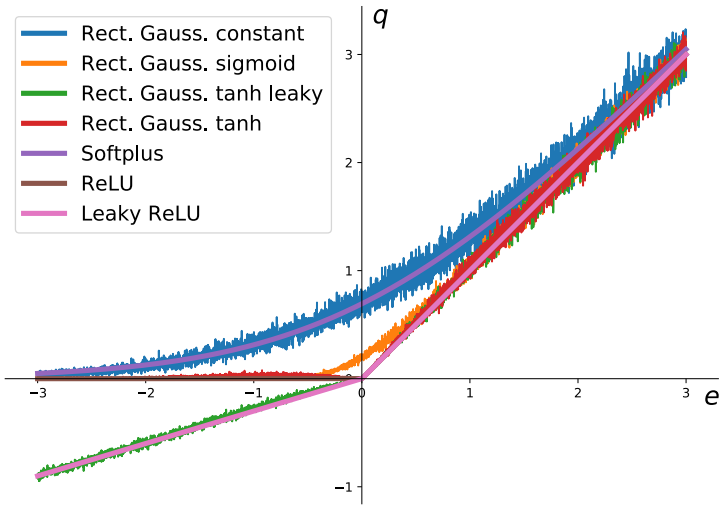
Yes we can. With rectified Gaussian nodes, defined as:

$$T_i^l(X_i^l) = X_i^l = \max(0, Y_i^l); \quad Y_i^l \sim \mathcal{N}\left(e_i^l(T^{l-1}(X^{l-1})), (s_i^l)^2\right)$$

In fact, several non-linearities can be approximated from this family:

- softplus: $s_i^l \approx 1.776$ (constant std);
- ReLU: $s_i^l = \tanh(e_i^l(T^{l-1}(X^{l-1})))$;
- ϵ-Leaky ReLU: $X_i^l = \max(\epsilon Y_i^l, Y_i^l)$.

$\rightsquigarrow$ Without rectification (or $\epsilon = 1$), we get the identity activation!

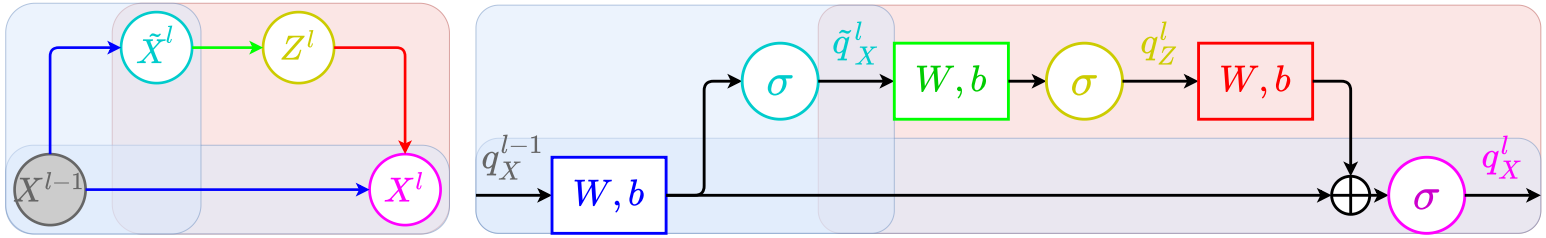


Outline

- Deep learning: the original series
- Deep learning: the next generation?
- Neural networks as chain graphs
  - Core ideas
  - Generality of the CG interpretation
  - Selected case studies of NN designs
- Conclusion and discussions

Going deeper: residual block as refinement module

Residual block is effective for going deeper $\Rightarrow$ better results!
$\rightsquigarrow$ Can we interpret it as CG? Yes we can!



Introducing the refinement module: side-chain augmentation!

- base submodule: $X^{l-1} \to X^l$;
- refining submodule: $\tilde{X}^l \to Z^l \to X^l$.

Interesting remarks:

- **Pre-activation variant** $\rightsquigarrow$ found as the preferred choice[19]!
- **Recursive refinement** $\rightsquigarrow$ sequence of consecutive res. blocks!

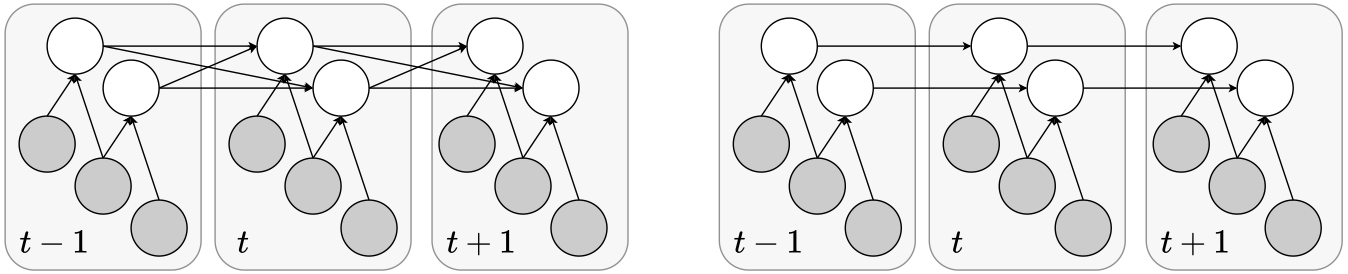
[19] He et al. "Identity Mappings in Deep Residual Networks". In: *ECCV*. 2016.

Learning sequential data

Simple (Elman) RNN $\Rightarrow$ Markov assumption

$\rightsquigarrow$ component $P(X^{l,t}|X^{l-1,t}, X^{l,t-1})$ + dense connect. $X^{l,t-1} \rightarrow X^{l,t}$

$\rightsquigarrow$ Vanishing / exploding gradient! alleviated with LSTM, GRU ...



Rethinking dependency: conditional independence through time!

$$P(X_i^{l,t}|X^{l-1,t}, X^{l,t-1}) = P(X_i^{l,t}|X^{l-1,t}, X_i^{l,t-1})$$

$\rightsquigarrow$ We have rediscovered the independently RNN[20]! Yay!

- Simpler than Elman RNN, better than LSTM!
- Can use ReLU, residual block, etc and go deeper.

[20] Li et al. "Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN". . In: *CVPR*. 2018.

Stochastic regularization with dropout

We can also replicate dropout within the CG framework:

- For each X_j^{l-1}, introduce an auxiliary RV $D_j^{l-1} \sim \text{Bernoulli}(p^{l-1})$;
- Pairwise connection $X_j^{l-1} \to X_i^l \Longrightarrow$ ternary $X_j^{l-1} \to X_i^l \leftarrow D_j^{l-1}$.

Node-wise CPD with dropout:

$$P(X_i^l | X^{l-1}, D^{l-1}) = f_i^l\left(X_i^l, e_i^l(T^{l-1}(X^{l-1}), D^{l-1})\right)$$

$$\text{with } e_i^l(T^{l-1}(X^{l-1}), D^{l-1}) = b_i^l + \sum_{j=1}^{N^{l-1}} D_j^{l-1}\, w_{j,i}^l\, T^{l-1}(X_j^{l-1}).$$

Reproducing the behavior of dropout:

- During training: sample D_j^{l-1} during each feed-forward;
- During test: marginalize $D_j^{l-1} \Rightarrow$ constant scale p^{l-1}.

Stochasticity without dropout?

We have a PGM, stochastic behavior can simply be achieved via sampling!

For directed graphs, we can use forward (ancestral) sampling!

- need a single pass like feed-forward;
- produce unbiased samples.

Issue: sampling doesn't go well with backprop!

- Continuous RV: reparametrization trick
- Discrete RV: Gumbel-softmax trick? (continuous relaxation ...)
- High variance

What can we do?

$\rightsquigarrow$ what made dropout work?

Something new: partially collapsed feed-forward

Main idea: sampling only a part of the nodes. (with proba. $p \in [0, 1]$)

$\rightsquigarrow$ we simply "mix up" feed-forward and forward sampling!

$$q_i^l = \begin{cases} E_{Q_i^l}[T^l(X_i^l)] & \text{if collapsed (feed-forward);} \\ T^l(x_i^l),\ x_i^l \sim Q_i^l & \text{if uncollapsed (forward sampling).} \end{cases}$$

Following "collapsed sampling", I dub thee partially collapsed feed-forward !

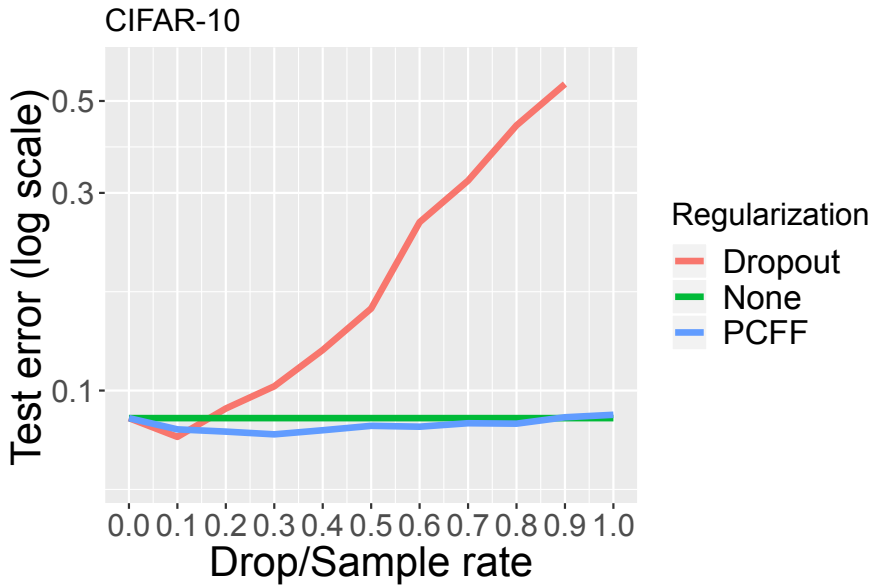
Cool things about PCFF:

- gradient flows freely (can still use reparam. trick)
- offers a bias–variance trade-off
- generalizes over feed-forward and forward sampling

Experimental evaluations of PCFF

Does PCFF actually work in practice? Yup!

Experiments with Conv. ResNet20 on CIFAR-10:



- Comparable to dropout, but more consistent improvement!
- More exp. in the paper: PCFF also work for IndRNN!

Conclusion and discussions

- The take-home message:

Neural networks are chain graphs!

- I repeat (because it is the take-home message!):

Neural networks are chain graphs!

- Some possible tracks for future work:
  1. Analyze existing designs using the NNasCG framework;
  2. Propose improvements / new designs guided by the framework;
  3. Go beyond the FeedForward + BackProp learning strategy;
  4. And more!

Many work can be done: collaborations are welcome!

Thank you! Questions?

[21] From http://barkingalien.blogspot.com/2017/02/i-choose-you.html