A THEORETICAL FRAMEWORK FOR DEEP LOCALLY CONNECTED RELU NETWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Understanding theoretical properties of deep and locally connected nonlinear network, such as deep convolutional neural network (DCNN), is still a hard problem despite its empirical success. In this paper, we propose a novel theoretical framework for such networks with ReLU nonlinearity. The framework explicitly formulates data distribution, favors disentangled representations and is compatible with common regularization techniques such as Batch Norm. The framework is built upon teacher-student setting, by expanding the student forward/backward propagation onto the teacher's computational graph. The resulting model does not impose unrealistic assumptions (e.g., Gaussian inputs, independence of activation, etc). Our framework could help facilitate theoretical analysis of many practical issues, e.g. overfitting, generalization, disentangled representations in deep networks.

1 INTRODUCTION

Deep Convolutional Neural Network (DCNN) has achieved a huge empirical success in multiple disciplines (e.g., computer vision (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; He et al., 2016), Computer Go (Silver et al., 2016; 2017; Tian & Zhu, 2016), and so on). On the other hand, its theoretical properties remain an open problem and an active research topic.

Learning deep models are often treated as non-convex optimization in a high-dimensional space. From this perspective, many properties in deep models have been analyzed: landscapes of loss functions (Choromanska et al., 2015b; Li et al., 2017; Mei et al., 2016), saddle points (Du et al., 2017; Dauphin et al., 2014), relationships between local minima and global minimum (Kawaguchi, 2016; Hardt & Ma, 2017; Safran & Shamir, 2017), trajectories of gradient descent (Goodfellow et al., 2014), path between local minima (Venturi et al., 2018), etc.

However, two components are missing: such a modeling does not consider specific network structure and input data distribution, both of which are critical factors in practice. Empirically, deep models work particular well for certain forms of data (e.g., images); theoretically, for certain data distribution, popular methods like gradient descent is shown to be unable to recover the network parameters (Brutzkus & Globerson, 2017).

Along this direction, previous theoretical works assume specific data distributions like spherical Gaussian and focus on shallow nonlinear networks (Tian, 2017; Brutzkus & Globerson, 2017; Du et al., 2018). These assumptions yield nice forms of gradient to enable analysis of many properties such as global convergence, which makes it nontrivial to extend to deep nonlinear neural networks that yield strong empirical performance.

In this paper, we propose a novel theoretical framework for deep locally connected ReLU network that is applicable to general data distributions. Specifically, we embrace a teacher-student setting: the *teacher* generates classification label via a hidden computational graph, and the *student* updates the weights to fit the labels with gradient descent. Then starting from gradient descent rule, we marginalize out the input data conditioned on the graph variables of the teacher at each layer, and arrive at a *reformulation* that **(1)** captures data distribution as explicit terms and leads to more interpretable model, **(2)** compatible with existing state-of-the-art regularization techniques such as Batch Normalization (Ioffe & Szegedy, 2015), and **(3)** favors disentangled representation when data distributions have factorizable structures. To our best knowledge, our work is the first theoretical framework to achieve these properties for deep and locally connected nonlinear networks.

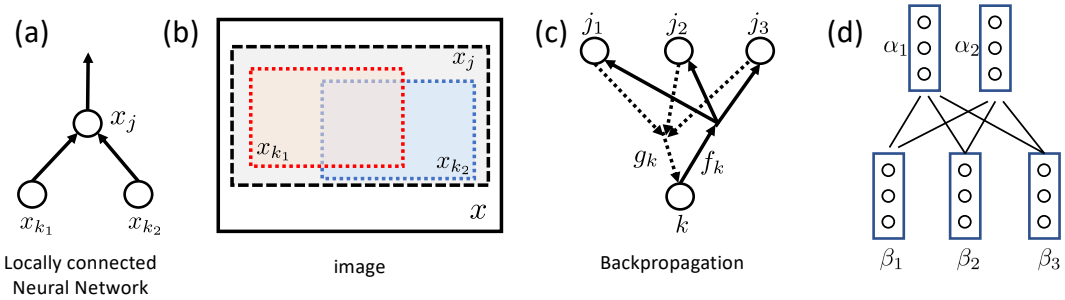


Figure 1: Problem Setting. (a) Locally connected network, (b) the receptive fields of each node. (c) notations used in backpropagation. (d) nodes with same receptive fields are grouped (Eqn. 56).

Previous works have also proposed framework to explain deep networks, e.g., renormalization group for restricted Boltzmann machines (Mehta & Schwab, 2014), spin-glass models (Amit et al., 1985; Choromanska et al., 2015a), transient chaos models (Poole et al., 2016), differential equations (Su et al., 2014; Saxe et al., 2013). In comparison, our framework **(1)** imposes mild assumptions rather than unrealistic ones (e.g., independence of activations), **(2)** explicitly deals with back-propagation which is the dominant approach used for training in practice, **(3)** considers spatial locality of neurons, an important component in practical deep models, and **(4)** models data distribution explicitly.

The paper is organized as follows: Sec. 2 introduces a novel approach to model locally connected networks. Sec. 3 introduces the teacher-student setting and label generation, followed by the proposed reformulation. Sec. 4 gives one novel finding that Batch Norm is a projection onto the orthogonal complementary space of neuron activations, and the reformulation is compatible with it. Sec. 5 shows a few applications of the framework, e.g., why nonlinearity is helpful, how factorization of the data distribution leads to disentangled representation and other issues.

2 Basic Formulation

2.1 General Setting

In this paper, we consider multi-layer (deep) network with ReLU nonlinearity. We consider supervised setting, in which we have a dataset $\{(x, y)\}$, where x is the input image and y is its label computed from x in a deterministic manner. Sec. 3 describes how x maps to y in details.

We consider a neuron (or node) j. Denote f_j as its activation after nonlinearity and g_j as the (input) gradient it receives after filtered by ReLU's gating. Note that both f_j and g_j are deterministic functions of the input x and label y. Since y is a deterministic function of x, we can write $f_j = f_j(x)$ and $g_j = g_j(x)$. Note that all analysis still holds with bias terms. We omit them for brevity.

The activation f_j and gradient g_k can be written as (note that f'_j is the binary gating function):

$$f_j(x) = f'_j(x) \sum_{k \in \mathrm{ch}(j)} w_{jk} f_k(x), \quad g_k(x) = f'_k(x) \sum_{j \in \mathrm{pa}(k)} w_{jk} g_j(x) \tag{1}$$

And the weight update for gradient descent is:

$$\Delta w_{jk} = \mathbb{E}_x \left[f_k(x) g_j(x) \right] \tag{2}$$

Here is the expectation is with respect to a training dataset (or a batch), depending on whether GD or SGD has been used. We also use f_j^{raw} and g_j^{raw} as the counterpart of f_j and g_j before nonlinearity.

2.2 Locally Connected Network

Locally connected networks have extra structures, which leads to our reformulation. As shown in Fig. 1, each node j only covers one part of the input images (i.e., *receptive field*). We use Greek letters $\{\alpha, \beta, \ldots, \omega\}$ to represent receptive fields. For a region α, x_α is the content in that region. $j \in \alpha$ means node j covers the region of α and n_α is the number of nodes that cover the same region (e.g., multi-channel case). The image content is $x_{\alpha(j)}$, abbreviated as x_j if no ambiguity. The parent j's receptive field covers its children's. Finally, ω represents the entire image.

By definition, the activation f_j of node j is only dependent on the region x_j, rather than the entire image x. This means that $f_j(x) = f_j(x_j)$ and $f_j(x_j) = f'_j(x_j)\sum_k w_{jk} f_k(x_k)$. However, the gradient g_j is determined by the entire image x, and its label y, i.e., $g_j = g_j(x, y)$. Note that here we assume that the label y is a deterministic (but unknown) function of x. Therefore, for gradient we just write $g_j = g_j(x)$.

2.3 Marginalized Gradient

Given the structure of locally connected network, the gradient g_j has some nice structures. From Eqn. 2 we knows that $\Delta w_{jk} = \mathbb{E}_x\left[f_k(x)g_j(x)\right] = \mathbb{E}_{x_k}\left[f_k(x_k)\mathbb{E}_{x_{-k}|x_k}\left[g_j(x)\right]\right]$. Define $x_{-k} = x\backslash x_k$ as the input image x except for x_k. Then we can define the *marginalized gradient*:

$$g_j(x_k) = \mathbb{E}_{x_{-k}|x_k}\left[g_j(x)\right] \tag{3}$$

as the marginalization (average) of x_{-k}, while keep x_k fixed. With this notation, we can write $\Delta w_{jk} = \mathbb{E}_{x_k}\left[f_k(x_k)g_j(x_k)\right]$.

On the other hand, the gradient which back-propagates to a node k can be written as

$$g_k(x) = f'_k(x)\sum_{j\in\mathrm{pa}(k)} w_{jk}g_j(x) = f'_k(x_k)\sum_j w_{jk}g_j(x) \tag{4}$$

where f'_k is the derivative of activation function of node k (for ReLU it is just a gating function). If we take expectation with respect to $x_{-k}|x_k$ on both side, we get

$$g_k(x_k) = f'_k(x_k)g_k^{\mathrm{raw}}(x_k) = f'_k(x_k)\sum_{j\in\mathrm{pa}(k)} w_{jk}g_j(x_k) \tag{5}$$

Note that all marginalized gradients $g_j(x_k)$ are independently computed by marginalizing with respect to all regions that are outside the receptive field x_k. Interestingly, there is a relationship between these gradients that respects the locality structure:

Theorem 1 (Recursive Property of marginalized gradient)**.** $g_j(x_k) = \mathbb{E}_{x_{j,-k}|x_k}\left[g_j(x_j)\right]$

This shows that the marginal gradient has recursive structure: we can first compute $g_j(x_j)$ for top node j, then by marginalizing over the region within x_j but outside x_k, we get its projection $g_j(x_k)$ on child k, then by Eqn. 5 we collect all projections from all the parents of node k, to get $g_k(x_k)$. This procedure can be repeated until we arrive at the leaf nodes.

3 Teacher-Student Setting

In order to analyze the behavior of neural network under backpropagation (BP), one needs to make assumptions about how the input x is generated and how the label y is related to the input x. Previous works assume Gaussian inputs and shallow networks, which yields analytic solution to gradient (Tian, 2017; Du et al., 2018), but might not align well with the practical data distribution.

3.1 The Summarization Ladder

We consider a multi-layered deterministic function as the teacher (not necessary a neural network). For each region x_α, there is a latent discrete *summarization* variable z_α that captures the information of the input x_α. Furthermore, we assume $z_\alpha = z_\alpha(x_\alpha) = z_\alpha(\{z_\beta\}_{\beta\in\mathrm{ch}(\alpha)})$, i.e., the summarizaion only relies on the ones in the immediate lower layer. In particular, the top level summarization is the *label* of the image, $y = z_\omega$, where ω represents the region of the entire image. We call a particular assignment of z_α, $z_\alpha = a$, an *event*. Finally, m_α is how many values z_α can take.

During training, all summarization functions $\mathcal{Z} = \{z_\alpha\}$ are unknown except for the label y.

3.2 Function Expansion on Summarization

Let's consider the following quantity. For each neural node j, we want to compute the expected gradient given a particular *factor* z_α, where $\alpha = \mathrm{rf}(j)$ (the reception field of node j):

$$g_j(z_\alpha) \equiv \mathbb{E}_{X_j|z_\alpha}\left[g_j(X_j)\right] = \int g_j(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j \tag{6}$$

	Dimension	Description
F_α, $\tilde{G}_\alpha$, D_α	m_α-by-n_α	Activation $f_j(z_\alpha)$, gradient $\tilde{g}_j(z_\alpha)$ and gating prob $f'_j(z_\alpha)$ at group α.
$W_{\beta\alpha}$	n_β-by-n_α	Weight matrix that links group α and β
$P_{\alpha\beta}$	m_α-by-m_β	Prob $\mathbb{P}(z_\beta \| z_\alpha)$ of events at group α and β

Table 1: Matrix Notation. See Eqn. 56.

And $\tilde{g}_j(z_\alpha) = g_j(z_\alpha)\mathbb{P}(z_\alpha)$. Similarly, $f_j(z_\alpha) = \mathbb{E}_{X_j|z_\alpha}\left[f_j(X_j)\right]$ and $f'_j(z_\alpha) = \mathbb{E}_{X_j|z_\alpha}\left[f'_j(X_j)\right]$.

Note that $\mathbb{P}(x_j|z_\alpha)$ is the *frequency count* of x_j for z_α. If z_α captures all information of x_j, then $\mathbb{P}(x_j|z_\alpha)$ is a delta function. Throughout the paper, we use frequentist interpretation of probabilities.

Intuitively, if we have $g_j(z_\alpha = a) > 0$ and $g_j(z_\alpha \neq a) < 0$, then the node j learns about the *hidden* event $z_\alpha = a$. For multi-class classification, the top level nodes (just below the softmax layer) already embrace such correlations (here j is the class label):

$$g_j(y = j) > 0, \quad g_j(y \neq j) < 0, \tag{7}$$

where we know $z_\omega = y$ is the top level factor. A natural question now arises:

Does gradient descent automatically push $g_j(z_\alpha)$ to be correlated with the factor z_α?

If this is true, then gradient descent on deep models is essentially a *weak-supervised* approach that automatically learns the intermediate events at different levels. Giving a complete answer of this question is very difficult and is beyond the scope of this paper. Here, we aim to build a theoretical framework that enables such analysis. We start with the relationship between neighboring layers:

Theorem 2 (Reformulation)**.** *Denote $\alpha = \mathrm{rf}(j)$ and $\beta = \mathrm{rf}(k)$. k is a child of j. If the following conditions hold:*

- ***Focus of knowledge***. $\mathbb{P}(x_k|z_\alpha, z_\beta) = \mathbb{P}(x_k|z_\beta)$.
- ***Broadness of knowledge***. $\mathbb{P}(x_j|z_\alpha, z_\beta) = \mathbb{P}(x_j|z_\alpha)$.
- ***Decorrelation***. *Given z_β, ($g_k^{\mathrm{raw}}(\cdot)$ and $f'_k(\cdot)$) and ($f_k^{\mathrm{raw}}(\cdot)$ and $f'_k(\cdot)$) are* uncorrelated.

Then the following iterative equations hold:

$$f_j(z_\alpha) = f'_j(z_\alpha) \sum_{k \in \mathrm{ch}(j)} w_{jk} \mathbb{E}_{z_\beta|z_\alpha}\left[f_k(z_\beta)\right], \quad g_k(z_\beta) = f'_k(z_\beta) \sum_{j \in \mathrm{pa}(k)} w_{jk} \mathbb{E}_{z_\alpha|z_\beta}\left[g_j(z_\alpha)\right] \tag{8}$$

One key property of this formulation is that, it incorporates data distributions $\mathbb{P}(z_\alpha, z_\beta)$ into the gradient descent rules. This is important since running BP on different dataset is now formulated into the same framework with different probability, i.e., frequency counts of events. By studying which family of distribution leads to the desired property, we could understand BP better.

For completeness, we also need to define boundary conditions. In the lowest level L, we could treat each input pixel (or a group of pixels) as a single event. Therefore, $f_k(z_\beta) = \mathbb{I}\left[k = z_\beta\right]$. On the top level, as we have discussed, Eqn. 7 applies and $g_j(z_\beta) = a_1 \mathbb{I}\left[j = z_\beta\right] - a_2 \mathbb{I}\left[j \neq z_\beta\right]$.

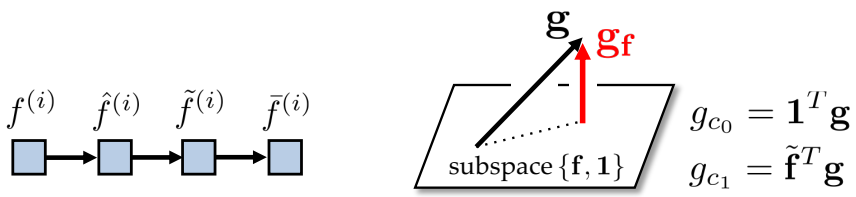
The following theorem shows that the reformulation is exact if z_α has all information of the region.

Theorem 3. *If $\mathbb{P}(x_j|z_\alpha)$ is a delta function for all α, then all conditions in Thm. 2 hold.*

In general, $\mathbb{P}(x_j|z_\alpha)$ is a distribution encoding how much information gets lost if we only know the factor z_α. When we climb up the ladder, we lose more and more information while keeping the critical part for the classification. This is consistent with empirical observations (Bau et al., 2017), in which the low-level features in DCNN are generic, and high-level features are more class-specific.

3.3 Matrix Formulation

Eqn. 8 can be hard to deal with. If we group the nodes with the same reception field at the same level together (Fig. 1(d)), we have the matrix form ($\circ$ is element-wise multiplication):



(a) Sub-layers in BN layer (b) Projected Gradient

Figure 2: Batch Normalization (BN) as a projection. **(a)** Three sublayers in BN (zero-mean, unit-variance, affine). **(b)** The gradient $\mathbf{g_f}$ that is propagated down is a projection of input gradient $\mathbf{g}$ onto the orthogonal complementary space spanned by $\{\mathbf{f}, \mathbf{1}\}$.

Theorem 4 (Matrix Representation of Reformulation)**.**

$$F_\alpha = D_\alpha \circ \sum_{\beta \in \mathrm{ch}(\alpha)} P_{\alpha\beta} F_\beta W_{\beta\alpha}, \quad \tilde{G}_\beta = D_\beta \circ \sum_{\alpha \in \mathrm{pa}(\beta)} P_{\alpha\beta}^T \tilde{G}_\alpha W_{\beta\alpha}^T, \quad \Delta W_{\beta\alpha} = (P_{\alpha\beta} F_\beta)^T \tilde{G}_\alpha \tag{9}$$

See Tbl. 2 for the notation. For this dynamics, we want $F_\omega^* = I_{n_\omega}$, i.e., the top n_ω neurons faithfully represents the classification labels. Therefore, the top level gradient is $G_\omega = I_{n_\omega} - F_\omega$. On the other side, for each region β at the bottom layer, we have $F_\beta = I_{n_\beta}$, i.e., the input contains all the preliminary factors. For all regions α in the top-most and bottom-most layers, we have $n_\alpha = m_\alpha$.

4 Batch Normalization under Reformulation

Our reformulation naturally incorporates empirical regularization technique like Batch Normalization (BN) (Ioffe & Szegedy, 2015).

4.1 Batch Normalization as a Projection

We start with a novel finding of Batch Norm: the back-propagated gradient through Batch Norm layer at a node j is a projection onto the orthogonal complementary subspace spanned by all one vectors and the current activations of node j.

Denote pre-batchnorm activations as $f^{(i)} = f_j(x_i)$ $(i = 1 \dots N)$. In Batch Norm, $f^{(i)}$ is whitened to be $\tilde{f}^{(i)}$, then linearly transformed to yield the output $\bar{f}^{(i)}$:

$$\hat{f}^{(i)} = f^{(i)} - \mu, \quad \tilde{f}^{(i)} = \hat{f}^{(i)}/\sigma, \quad \bar{f}^{(i)} = c_1 \tilde{f}^{(i)} + c_0 \tag{10}$$

where $\mu = \frac{1}{N}\sum_i f^{(i)}$ and $\sigma^2 = \frac{1}{N}\sum_i (f^{(i)} - \mu)^2$ and c_1, c_0 are learnable parameters.

The original Batch Norm paper derives complicated and unintuitive weight update rules. With vector notation, the update has a compact form with clear geometric meaning.

Theorem 5 (Backpropagation of Batch Norm)**.** *For a top-down gradient* $\mathbf{g}$*, BN layer gives the following gradient update (*$P_{\mathbf{f},\mathbf{1}}^\perp$ *is the orthogonal complementary projection of subspace* $\{\mathbf{f}, \mathbf{1}\}$*):*

$$\mathbf{g_f} = J^{BN}(\mathbf{f})\mathbf{g} = \frac{c_1}{\sigma} P_{\mathbf{f},\mathbf{1}}^\perp \mathbf{g}, \quad \mathbf{g_c} = S(\mathbf{f})^T \mathbf{g} \tag{11}$$

Intuitively, the back-propagated gradient $J^{BN}(\mathbf{f})\mathbf{g}$ is zero-mean and perpendicular to the input activation $\mathbf{f}$ of BN layer, as illustrated in Fig. 2. Unlike (Kohler et al., 2018) that analyzes BN in an approximate manner, in Thm. 5 we do not impose any assumptions.

4.2 Batch Norm under the Reformulation

The analysis of Batch Norm is compatible with the reformulation and we arrive at similar backpropagation rule, by noticing that $\mathbb{E}_x\left[f_j(x)\right] = \mathbb{E}_{z_\alpha}\left[f_j(z_\alpha)\right]$:

$$\mu = \mathbb{E}_{z_\alpha}\left[f_j\right], \quad \sigma^2 = \mathbb{E}_{z_\alpha}\left[(f_j(z_\alpha) - \mu)^2\right], \quad J^{BN}(\mathbf{f}) = \frac{c_1}{\sigma} P_{\mathbf{f},\mathbf{1}}^\perp \tag{12}$$

Note that we still have the projection property, but under the new inner product $\langle f_j, g_j \rangle_{z_\alpha} = \mathbb{E}_{z_\alpha}\left[f_j(z_\alpha) g_j(z_\alpha)\right]$ and norm $\|f\|_{z_\alpha} = \langle f, f \rangle_{z_\alpha}^{1/2}$.

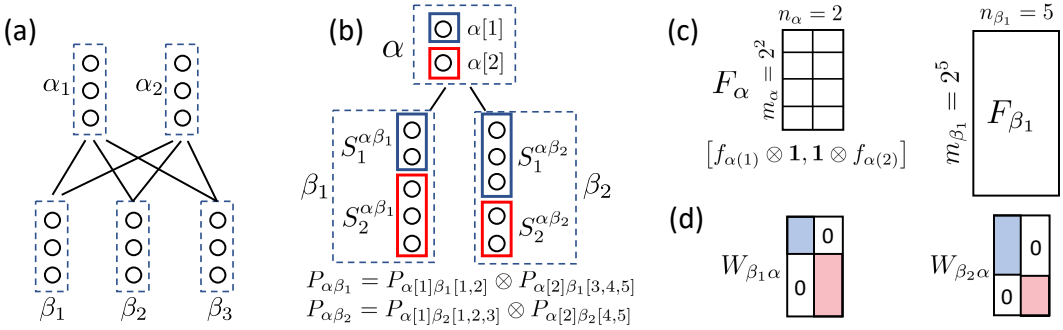


Figure 3: disentangled representation. **(a)** Nodes are grouped according to regions. **(b)** An example of one parent region α (2 nodes) and two child regions β_1 and β_2 (5 nodes each). We assume factorization property of data distribution P. **(c)** disentangled activations, **(d)** Separable weights.

5 Example applications of proposed theoretical framework

With the help of the theoretical framework, we now can analyze interesting structures of gradient descent in deep models, when the data distribution $\mathbb{P}(z_\alpha, z_\beta)$ satisfies specific conditions. Here we give two concrete examples: the role played by nonlinearity and in which condition disentangled representation can be achieved. Besides, from the theoretical framework, we also give general comments on multiple issues (e.g., overfitting, GD versus SGD) in deep learning.

5.1 Nonlinear versus linear

In the formulation, m_α is the number of possible events within a region α, which is often exponential with respect to the size $\mathrm{sz}(\alpha)$ of the region. The following analysis shows that a linear model cannot handle it, even with exponential number of nodes n_α, while a nonlinear one with ReLU can.

Definition 1 (Convex Hull of a Set)**.** *We define the convex hull* $\mathrm{Conv}(P)$ *of* m *points* $P \subset \mathbb{R}^n$ *to be* $\mathrm{Conv}(P) = \left\{P\mathbf{a}, \mathbf{a} \in \mathbf{\Delta}^{n-1}\right\}$*, where* $\mathbf{\Delta}^{n-1} = \{\mathbf{a} \in \mathbb{R}^n, a_i \geq 0, \sum_i a_i = 1\}$*. A row* p_j *is called* vertex *if* $p_j \notin \mathrm{Conv}(P \backslash p_j)$*.*

Definition 2. *A matrix* P *of size* m*-by-*n *is called* k-vert*, or* $\mathrm{vert}(P) = k \leq m$*, if its* k *rows are vertices of the convex hull generated by its rows.* P *is called* all-vert *if* $k = m$*.*

Theorem 6 (Expressibility of ReLU Nonlinearity)**.** *Assuming* $m_\alpha = n_\alpha = \mathcal{O}(\exp(\mathrm{sz}(\alpha)))$*, where* $\mathrm{sz}(\alpha)$ *is the size of receptive field of* α*. If each* $P_{\alpha\beta}$ *is all-vert, then: (*ω *is top-level receptive field)*

$$\min_W Loss_{\mathrm{ReLU}}(W) = 0, \quad \min_W Loss_{\mathrm{Linear}}(W) = \mathcal{O}(\exp(\mathrm{sz}(\omega))) \tag{13}$$

Note that here $Loss(W) \equiv \|F_\omega - I\|_F^2$. This shows the power of nonlinearity, which guarantees full rank of output, even if the matrices involved in the multiplication are low-rank. The following theorem shows that for intermediate layers whose input is not identity, the all-vert property remains.

Theorem 7. **(1)** *If* F *is full row rank, then* $\mathrm{vert}(PF) = \mathrm{vert}(P)$*.* **(2)** PF *is all-vert iff* P *is all-vert.*

This means that if all $P_{\alpha\beta}$ are all-vert and its input F_β is full-rank, then with the same construction of Thm. 6, F_α can be made identity. In particular, if we sample W randomly, then with probability 1, all F_β are full-rank, in particular the top-level input F_1. Therefore, using top-level W_1 alone would be sufficient to yield zero generalization error, as shown in the previous works that random projection could work well.

5.2 Disentangled Representation

The analysis in Sec. 5.1 assumes that $n_\alpha = m_\alpha$, which means that we have sufficient nodes, *one neuron for one event*, to convey the information forward to the classification level. In practice, this is never the case. When $n_\alpha \ll m_\alpha = \mathcal{O}(\exp(\mathrm{sz}(\alpha)))$ and the network needs to represent the information in a proper way so that it can be sent to the top level. Ideally, if the factor z_α can be written down as a list of binary factors: $z_\alpha = \left[z_{\alpha[1]}, z_{\alpha[2]}, \ldots, z_{\alpha[j]}\right]$, the output of a node j could represent $z_{\alpha[j]}$, so that all m_α events can be represented concisely with n_α nodes.

To come up with a complete theory for disentangled representation in deep nonlinear network is far from trivial and beyond the scope of this paper. In the following, we make an initial attempt by constructing factorizable $P_{\alpha\beta}$ so that disentangled representation is possible in the forward pass. First we need to formally define what is disentangled representation:

Definition 3. *The activation F_α is* disentangled*, if its j-th column $F_{\alpha,:j} = \mathbf{1} \otimes \ldots \otimes \mathbf{f}_{\alpha[j]} \otimes \ldots \otimes \mathbf{1}$, where each $\mathbf{f}_{\alpha[j]}$ and $\mathbf{1}$ is a 2-by-1 vector.*

Definition 4. *The gradient $\tilde{G}_\alpha$ is* disentangled*, if its j-th column $\tilde{G}_{\alpha,:j} = \mathbf{p}_{\alpha[1]} \otimes \ldots \otimes \tilde{\mathbf{g}}_{\alpha[j]} \otimes \ldots \otimes \mathbf{p}_{\alpha[n_\alpha]}$, where $\mathbf{p}_{\alpha[j]} = [\mathbb{P}(\alpha[j]=0), \mathbb{P}(\alpha[j]=1)]^T$ and $\tilde{\mathbf{g}}_{\alpha[j]}$ is a 2-by-1 vector.*

Intuitively, this means that each node j represents the binary factor $z_\alpha[j]$. A follow-up question is whether such disentangled properties carries over layers in the forward pass. It turns out that the disentangled structure carries if the data distribution and weights have compatible structures:

Definition 5. *The weights $W_{\beta\alpha}$ is* separable *with respect to a disjoint set $\{S_i^{\alpha\beta}\}$, if $W_{\beta\alpha} = \mathrm{diag}\left(W_{\beta\alpha}[S_1^{\alpha\beta}, 1], W_{\beta\alpha}[S_2^{\alpha\beta}, 2], \ldots, W_{\beta\alpha}[S_{n_\alpha}^{\alpha\beta}, n_\alpha]\right)$.*

Theorem 8 (Disentangled Forward)**.** *If for each $\beta \in \mathrm{ch}(\alpha)$, $P_{\alpha\beta}$ can be written as a tensor product $P_{\alpha\beta} = \bigotimes_i P_{\alpha[i]\beta[S_i^{\alpha\beta}]}$ where $\{S_i^{\alpha\beta}\}$ are $\alpha\beta$-dependent disjointed set, $W_{\beta\alpha}$ is separable with respect to $\{S_i^{\alpha\beta}\}$, F_β is disentangled, then F_α is also disentangled (with/without ReLU /Batch Norm).*

If the bottom activations are disentangled, by induction, all activations will be disentangled. The next question is whether gradient descent preserves such a structure. The answer is also conditionally yes:

Theorem 9 (Separable Weight Update)**.** *If $P_{\alpha\beta} = \bigotimes_i P_{\alpha[i]\beta[S_i]}$, F_β and $\tilde{G}_\alpha$ are both disentangled, $\mathbf{1}^T \tilde{G}_\alpha = \mathbf{0}$, then the gradient update $\Delta W_{\beta\alpha}$ is separable with respect to $\{S_i\}$.*

Therefore, with disentangled F_β and $\tilde{G}_\alpha$ and centered gradient $\mathbf{1}^T \tilde{G}_\alpha = \mathbf{0}$, the separable structure is conserved over gradient descent, given the initial $W_{\beta\alpha}^{(0)}$ is separable. Note that centered gradient is guaranteed if we insert Batch Norm (Eqn. 12) after linear layers. And the activation F remains disentangled if the weights are separable.

The hard part is whether $\tilde{G}_\beta$ remains disentangled during backpropagation, if $\{\tilde{G}_\alpha\}_{\alpha \in \mathrm{pa}(\beta)}$ are all disentangled. If so, then the disentangled representation is self-sustainable under gradient descent. This is a non-trivial problem and generally requires structures of data distribution. We put some discussion in the Appendix and leave this topic for future work.

5.3 Explanation of common behaviors in Deep Learning

In the proposed formulation, the input x in Eqn. 8 is integrated out, and the data distribution is now encoded into the probabilistic distribution $\mathbb{P}(z_\alpha, z_\beta)$, and their marginals. A change of such distribution means the input distribution has changed. For the first time, we can now analyze many practical factors and behaviors in the DL training that is traditionally not included in the formulation.

Over-fitting. Given finite number of training samples, there is always error in estimated factor-factor distribution $\tilde{\mathbb{P}}(z_\alpha, z_\beta)$ and factor-observation distribution $\tilde{\mathbb{P}}(x_\alpha | z_\alpha)$. In some cases, a slight change of distribution would drastically change the optimal weights for prediction, which is overfitting.

Here is one example. Suppose there are two different kinds of events at two disjoint reception fields: z_α and z_γ. The class label is z_ω, which equals z_α but is not related to z_γ. Therefore, we have:

$$\tilde{\mathbb{P}}(z_\omega = 1 | z_\alpha = 1) = 1, \quad \tilde{\mathbb{P}}(z_\omega = 1 | z_\alpha = 0) = 0 \tag{14}$$

Although z_γ is unrelated to the class label z_ω, with finite samples z_γ could show spurious correlation:

$$\tilde{\mathbb{P}}(z_\omega = 1 | z_\gamma = 1) = 0.5 + \epsilon, \quad \tilde{\mathbb{P}}(z_\omega = 1 | z_\gamma = 0) = 0.5 - \epsilon \tag{15}$$

On the other hand, as shown in Fig. 4, $\mathbb{P}(x_\alpha | z_\alpha)$ contains a lot of detailed structures and is almost impossible to separate in the finite sample case, while $\mathbb{P}(x_\gamma | z_\gamma)$ could be well separated for $z_\gamma = 0/1$. Therefore, for node j with $\mathrm{rf}(j) = \alpha$, $f_j(z_\alpha) \approx \mathrm{constant}$ (input almost indistinguishable):

$$\Delta w_j = \mathbb{E}_{z_\alpha}\left[f_j(z_\alpha) g_0(z_\alpha)\right] \approx 0 \tag{16}$$

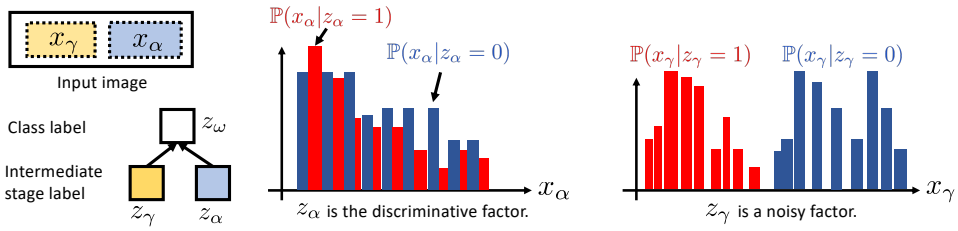


Figure 4: Overfitting Example

where $g_0(z_\alpha) = \mathbb{E}_{z_\omega|z_\alpha}[g_0(z_\omega)] = \begin{cases} 1 & z_\alpha = 1 \\ -1 & z_\alpha = 0 \end{cases}$, which is a strong gradient signal backpropagated from the top softmax level, since z_α is strongly correlated with z_ω. For node k with $\mathrm{rf}(k) = \gamma$, an easy separation of the input (e.g., random initialization) yields distinctive $f_k(z_\gamma)$. Therefore,

$$\Delta w_k = \mathbb{E}_{z_\gamma}[f_j(z_\gamma) g_0(z_\gamma)] > 0 \tag{17}$$

where $g_0(z_\gamma) = \mathbb{E}_{z_\omega|z_\gamma}[g_0(z_\omega)] = \begin{cases} 2\epsilon & z_\gamma = 1 \\ -2\epsilon & z_\gamma = 0 \end{cases}$, a weak signal because of z_γ is (almost) unrelated to the label. Therefore, we see that the weight w_j that links to meaningful receptive field z_α does not receive strong gradient, while the weight w_k that links to irrelevant (but spurious) receptive field z_γ receives strong gradient. This will lead to overfitting.

With more data, over-fitting is alleviated since (1) $\tilde{\mathbb{P}}(z_\omega|z_\gamma)$ becomes more accurate and $\epsilon \to 0$; (2) $\tilde{\mathbb{P}}(x_\alpha|z_\alpha)$ starts to show statistical difference for $z_\alpha = 0/1$ and thus $f_j(z_\alpha)$ shows distinctiveness.

Note that there exists a **second** explanation: we could argue that z_γ is a *true* but *weak* factor that contributes to the label, while z_α is a *fictitious* discriminative factor, since the appearance difference between $z_\alpha = 0$ and $z_\alpha = 1$ (i.e., $\tilde{\mathbb{P}}(x_\alpha|z_\alpha)$ for $\alpha = 0/1$) could be purely due to noise and thus should be neglected. With finite number of samples, these two cases are essentially indistinguishable. Models with different induction bias might prefer one to the other, yielding drastically different generalization error. For neural network, SGD prefers the second explanation but if under the pressure of training, it may also explore the first one by pushing gradient down to distinguish subtle difference in the input. This may explain why the same neural networks can fit random-labeled data, and generalize well for real data (Zhang et al., 2016).

Gradient Descent: Stochastic or not? Previous works (Keskar et al., 2017) show that empirically stochastic gradient decent (SGD) with small batch size tends to converge to "flat" minima and offers better generalizable solution than those uses larger batches to compute the gradient.

From our framework, SGD update with small batch size is equivalent to using a perturbed/noisy version of $\mathbb{P}(z_\alpha, z_\beta)$ at each iteration. Such an approach naturally reduces aforementioned over-fitting issues, which is due to hyper-sensitivity of data distribution and makes the final weight solution invariant to changes in $\mathbb{P}(z_\alpha, z_\beta)$, yielding a "flat" solution.

6 Conclusion and Future Work

In this paper, we propose a novel theoretical framework for deep (multi-layered) nonlinear network with ReLU activation and local receptive fields. The framework utilizes the specific structure of neural networks, and formulates input data distributions explicitly. Compared to modeling deep models as non-convex problems, our framework reveals more structures of the network; compared to recent works that also take data distribution into considerations, our theoretical framework can model deep networks without imposing idealistic analytic distribution of data like Gaussian inputs or independent activations. Besides, we also analyze regularization techniques like Batch Norm, depicts its underlying geometrical intuition, and shows that BN is compatible with our framework.

Using this novel framework, we have made an initial attempt to analyze many important and practical issues in deep models, and provides a novel perspective on overfitting, generalization, disentangled representation, etc. We emphasize that in this work, we barely touch the surface of these core issues in deep learning. As a future work, we aim to explore them in a deeper and more thorough manner, by using the powerful theoretical framework proposed in this paper.

REFERENCES

Daniel J Amit, Hanoch Gutfreund, and Haim Sompolinsky. Spin-glass models of neural networks. *Physical Review A*, 32(2):1007, 1985.

David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. *CVPR*, 2017.

Alon Brutzkus and Amir Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. *ICML*, 2017.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pp. 192–204, 2015a.

Anna Choromanska, Yann LeCun, and Gérard Ben Arous. Open problem: The landscape of the loss surfaces of multilayer networks. In *Conference on Learning Theory*, pp. 1756–1760, 2015b.

Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, pp. 2933–2941, 2014.

Simon S Du, Chi Jin, Jason D Lee, Michael I Jordan, Aarti Singh, and Barnabas Poczos. Gradient descent can take exponential time to escape saddle points. In *Advances in Neural Information Processing Systems*, pp. 1067–1077, 2017.

Simon S Du, Jason D Lee, Yuandong Tian, Barnabas Poczos, and Aarti Singh. Gradient descent learns one-hidden-layer cnn: Don't be afraid of spurious local minima. *ICML*, 2018.

Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *ICLR*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

Kenji Kawaguchi. Deep learning without poor local minima. In *Advances in Neural Information Processing Systems*, pp. 586–594, 2016.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *ICLR*, 2017.

Jonas Kohler, Hadi Daneshmand, Aurelien Lucchi, Ming Zhou, Klaus Neymeyr, and Thomas Hofmann. Towards a theoretical understanding of batch normalization. *arXiv preprint arXiv:1805.10694*, 2018.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

Pankaj Mehta and David J Schwab. An exact mapping between the variational renormalization group and deep learning. *arXiv preprint arXiv:1410.3831*, 2014.

Song Mei, Yu Bai, and Andrea Montanari. The landscape of empirical risk for non-convex losses. *arXiv preprint arXiv:1607.06534*, 2016.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pp. 3360–3368, 2016.

Itay Safran and Ohad Shamir. Spurious local minima are common in two-layer relu neural networks. *CoRR*, abs/1712.08968, 2017. URL `http://arxiv.org/abs/1712.08968`.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Weijie Su, Stephen Boyd, and Emmanuel Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pp. 2510–2518, 2014.

Yuandong Tian. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. *ICML*, 2017.

Yuandong Tian and Yan Zhu. Better computer go player with neural network and long-term prediction. *ICLR*, 2016.

Luca Venturi, Afonso Bandeira, and Joan Bruna. Neural networks with finite intrinsic dimension have no spurious valleys. *arXiv preprint arXiv:1802.06384*, 2018.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, 2016.

7 APPENDIX

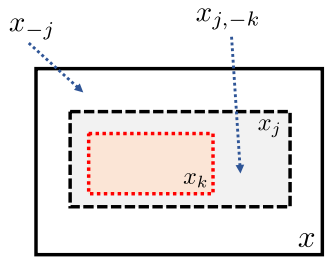


Figure 5: Notation used in Thm. 1.

7.1 HIERARCHICAL CONDITIONING

Theorem 1 (Recursive Property of marginalized gradient)**.**

$$g_j(x_k) = \mathbb{E}_{x_{j,-k}|x_k}\left[g_j(x_j)\right] \tag{18}$$

Proof. We have:

$$\begin{aligned}
g_j(x_k) &= \mathbb{E}_{x_{-k}|x_k}\left[g_j(x)\right] \\
&= \mathbb{E}_{x_{-j},x_{j,-k}|x_k}\left[g_j(x)\right] \\
&= \mathbb{E}_{x_{-j}|x_{j,-k},x_k}\left[\mathbb{E}_{x_{j,-k}|x_k}\left[g_j(x)\right]\right] \\
&= \mathbb{E}_{x_{-j}|x_j}\left[\mathbb{E}_{x_{j,-k}|x_k}\left[g_j(x)\right]\right] \\
&= \mathbb{E}_{x_{j,-k}|x_k}\left[\mathbb{E}_{x_{-j}|x_j}\left[g_j(x)\right]\right] \\
&= \mathbb{E}_{x_{j,-k}|x_k}\left[g_j(x_j)\right]
\end{aligned}$$

□

7.2 NETWORK THEOREM

Theorem 2 (Reformulation)**.** *Denote $\alpha = \mathrm{rf}(j)$ and $\beta = \mathrm{rf}(k)$. k is a child of j. If the following two conditions hold:*

- ***Focus of knowledge***. $\mathbb{P}(x_k|z_\alpha, z_\beta) = \mathbb{P}(x_k|z_\beta)$.
- ***Broadness of knowledge***. $\mathbb{P}(x_j|z_\alpha, z_\beta) = \mathbb{P}(x_j|z_\alpha)$.
- ***Decorrelation***. *Given z_β, ($g_k^{\mathrm{raw}}(\cdot)$ and $f'_k(\cdot)$) and ($f_k^{\mathrm{raw}}(\cdot)$ and $f'_k(\cdot)$) are* uncorrelated

Then the following two conditions holds:

$$f_j(z_\alpha) = f'_j(z_\alpha) \sum_{k\in\mathrm{ch}(j)} w_{jk}\mathbb{E}_{z_\beta|z_\alpha}\left[f_k(z_\beta)\right] \tag{19a}$$

$$g_k(z_\beta) = f'_k(z_\beta) \sum_{j\in\mathrm{pa}(k)} w_{jk}\mathbb{E}_{z_\alpha|z_\beta}\left[g_j(z_\alpha)\right] \tag{19b}$$

Proof. For Eqn. 19a, we have:

$$f_j^{\mathrm{raw}}(z_\alpha) = \int f_j^{\mathrm{raw}}(x)\mathbb{P}(x|z_\alpha)\mathrm{d}x \tag{20}$$

$$= \int f_j^{\mathrm{raw}}(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j \tag{21}$$

$$= \int \left(\sum_{k\in\mathrm{ch}(j)} w_{jk} f_k(x_k)\right)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j \tag{22}$$

And for each of the entry, we have:

$$\int f_k(x_k)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j = \int f_k(x_k)\mathbb{P}(x_k|z_\alpha)\mathrm{d}x_k \tag{23}$$

For $\mathbb{P}(x_k|z_\alpha)$, using focus of knowledge, we have:

$$\mathbb{P}(x_k|z_\alpha) = \sum_{z_\beta} \mathbb{P}(x_k, z_\beta|z_\alpha) \tag{24}$$

$$= \sum_{z_\beta} \mathbb{P}(x_k|z_\beta, z_\alpha)\mathbb{P}(z_\beta|z_\alpha) \tag{25}$$

$$= \sum_{z_\beta} \mathbb{P}(x_k|z_\beta)\mathbb{P}(z_\beta|z_\alpha) \tag{26}$$

Therefore, following Eqn. 23, we have:

$$\int f_k(x_k)\mathbb{P}(x_k|z_\alpha)\mathrm{d}x_k = \int f_k(x_k) \sum_{z_\beta} \mathbb{P}(x_k|z_\beta)\mathbb{P}(z_\beta|z_\alpha)\mathrm{d}x_k \tag{27}$$

$$= \sum_{z_\beta} \left(\int f_k(x_k)\mathbb{P}(x_k|z_\beta)\mathrm{d}x_k \right) \mathbb{P}(z_\beta|z_\alpha) \tag{28}$$

$$= \sum_{z_\beta} f_k(z_\beta)\mathbb{P}(z_\beta|z_\alpha) \tag{29}$$

$$= \mathbb{E}_{z_\beta|z_\alpha}\left[f_k(z_\beta)\right] \tag{30}$$

Putting it back to Eqn. 22 and we have:

$$f_j^{\mathrm{raw}}(z_\alpha) = \sum_{k\in\mathrm{ch}(j)} w_{jk}\mathbb{E}_{z_\beta|z_\alpha}\left[f_k(z_\beta)\right] \tag{31}$$

For Eqn. 19b, similarly we have:

$$g_k^{\mathrm{raw}}(z_\beta) = \int g_k^{\mathrm{raw}}(x)\mathbb{P}(x|z_\beta)\mathrm{d}x \tag{32}$$

$$= \int \sum_{j\in\mathrm{pa}(k)} w_{jk} g_j(x)\mathbb{P}(x|z_\beta)\mathrm{d}x \tag{33}$$

Notice that we have:

$$\mathbb{P}(x|z_\beta) = \mathbb{P}(x_j|z_\beta)\mathbb{P}(x_{-j}|x_j, z_\beta) = \mathbb{P}(x_j|z_\beta)\mathbb{P}(x_{-j}|x_j) \tag{34}$$

since x_j covers x_k which determines z_β. Therefore, for each item we have:

$$\int g_j(x)\mathbb{P}(x|z_\beta)\mathrm{d}x = \int g_j(x)\mathbb{P}(x_j|z_\beta)\mathbb{P}(x_{-j}|x_j)\mathrm{d}x \tag{35}$$

$$= \int \left(\int g_j(x)\mathbb{P}(x_{-j}|x_j)\mathrm{d}x_{-j} \right) \mathbb{P}(x_j|z_\beta)\mathrm{d}x_j \tag{36}$$

$$= \int g_j(x_j)\mathbb{P}(x_j|z_\beta)\mathrm{d}x_j \tag{37}$$

Then we use the broadness of knowledge:

$$\mathbb{P}(x_j|z_\beta) = \sum_{z_\alpha} \mathbb{P}(x_j, z_\alpha|z_\beta) \tag{38}$$

$$= \sum_{z_\alpha} \mathbb{P}(x_j|z_\alpha, z_\beta)\mathbb{P}(z_\alpha|z_\beta) \tag{39}$$

$$= \sum_{z_\alpha} \mathbb{P}(x_j|z_\alpha)\mathbb{P}(z_\alpha|z_\beta) \tag{40}$$

Following Eqn. 37, we now have:

$$\int g_j(x)\mathbb{P}(x|z_\beta)\mathrm{d}x = \int g_j(x_j)\sum_{z_\alpha}\mathbb{P}(x_j|z_\alpha)\mathbb{P}(z_\alpha|z_\beta)\mathrm{d}x_j \tag{41}$$

$$= \sum_{z_\alpha}\left(\int g_j(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j\right)\mathbb{P}(z_\alpha|z_\beta) \tag{42}$$

$$= \sum_{z_\alpha} g_j(z_\alpha)\mathbb{P}(z_\alpha|z_\beta) \tag{43}$$

$$= \mathbb{E}_{z_\alpha|z_\beta}\left[g_j(z_\alpha)\right] \tag{44}$$

Putting it back to Eqn. 33 and we have:

$$g_k^{\mathrm{raw}}(z_\beta) = \sum_{j\in\mathrm{pa}(k)} w_{jk}\mathbb{E}_{z_\alpha|z_\beta}\left[g_j(z_\alpha)\right] \tag{45}$$

Using Eqn. 5:

$$g_k(z_\beta) = \int g_k(x_k)\mathbb{P}(x_k|z_\beta)\mathrm{d}x_k \tag{46}$$

$$= \int f_k'(x_k)g_k^{\mathrm{raw}}(x_k)\mathbb{P}(x_k|z_\beta)\mathrm{d}x_k \tag{47}$$

$$= \mathbb{E}_{X_k|z_\beta}\left[f_k'(X_k)g_k^{\mathrm{raw}}(X_k)\right] \tag{48}$$

The un-correlation between $g_k^{\mathrm{raw}}(\cdot)$ and $f_k'(\cdot)$ means that

$$\mathbb{E}_{X_k|z_\beta}\left[f_k' g_k^{\mathrm{raw}}\right] = \mathbb{E}_{X_k|z_\beta}\left[f_k'\right]\cdot\mathbb{E}_{X_k|z_\beta}\left[g_k^{\mathrm{raw}}\right] \tag{49}$$

Similarly for $f_j(z_\alpha)$. □

7.3 Exactness of reformulation

Theorem 3. *If $\mathbb{P}(x_j|z_\alpha)$ is a delta function for all α, then the conditions of Thm. 2 hold and the reformulation becomes exact.*

Proof. The fact that $\mathbb{P}(x_j|z_\alpha)$ is a delta function means that there exists a function ϕ_j so that:

$$\mathbb{P}(x_j|z_\alpha) = \delta(x_j - \phi_j(z_\alpha)) \tag{50}$$

That is, z_α contains all information of x_j (or x_α). Therefore,

- **Broadness of knowledge.** z_α contains strictly more information than z_β for $\beta \in \mathrm{ch}(\alpha)$, therefore $\mathbb{P}(x_j|z_\alpha, z_\beta) = \mathbb{P}(x_j|z_\alpha)$.
- **Focus of knowledge.** z_β captures all information of z_k, so $\mathbb{P}(x_k|z_\alpha, z_\beta) = \mathbb{P}(x_k|z_\beta)$.
- **Decorrelation.** For any $h_1(x_j)$ and $h_2(x_j)$ we have

$$\mathbb{E}_{X_j|z_\alpha}\left[h_1 h_2\right] = \int h_1(x_j)h_2(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j \tag{51}$$

$$= \int h_1(x_j)h_2(x_j)\delta(x_j - \phi_j(z_\alpha))\mathrm{d}x_j \tag{52}$$

$$= h_1(\phi_j(z_\alpha))h_2(\phi_j(z_\alpha)) \tag{53}$$

$$= \int h_1(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j \int h_2(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j \tag{54}$$

$$= \mathbb{E}_{X_j|z_\alpha}\left[h_1\right]\mathbb{E}_{X_j|z_\alpha}\left[h_2\right] \tag{55}$$

□

	Dimension	Description
F_α, D_α	m_α-by-n_α	Activation $f_j(z_\alpha)$ and gating prob $f'_j(z_\alpha)$ in group α.
G_α, $\tilde{G}_\alpha$	m_α-by-n_α	Gradient $g_j(z_\alpha)$ and unnormalized gradient $\tilde{g}_j(z_\alpha)$ in group α.
$W_{\beta\alpha}$	n_β-by-n_α	Weight matrix that links group β and α.
$P_{\alpha\beta}$, $P^b_{\alpha\beta}$	m_α-by-m_β	Prob $\mathbb{P}(z_\beta \mid z_\alpha)$, $\mathbb{P}(z_\alpha \mid z_\beta)$ of events between group β and α.
Λ_α	m_α-by-m_α	Diagonal matrix encoding prior prob $\mathbb{P}(z_\alpha)$.

Table 2: Matrix Notation. See Eqn. 56.

7.4 MATRIX FORM

Theorem 4 (Matrix Representation of Reformulation)**.**

$$F_\alpha = D_\alpha \circ \sum_{\beta \in \mathrm{ch}(\alpha)} P_{\alpha\beta} F_\beta W_{\beta\alpha}, \quad \tilde{G}_\beta = D_\beta \circ \sum_{\alpha \in \mathrm{pa}(\beta)} P^T_{\alpha\beta} \tilde{G}_\alpha W^T_{\beta\alpha}, \quad \Delta W_{\beta\alpha} = (P_{\alpha\beta} F_\beta)^T \tilde{G}_\alpha \tag{56}$$

Proof. We first consider one certain group α and β, which uses x_α and x_β as the receptive field. For this pair, we can write Eqn. 19 in the following matrix form:

$$F^{\mathrm{raw}}_\alpha = P_{\alpha\beta} F_\beta W_{\beta\alpha} \tag{57a}$$
$$F_\alpha = F^{\mathrm{raw}}_\alpha \circ D_\alpha \tag{57b}$$
$$G^{\mathrm{raw}}_\beta = \left(P^b_{\alpha\beta}\right)^T G_\beta W^T_{\beta\alpha} \tag{57c}$$
$$G_\beta = G^{\mathrm{raw}}_\beta \circ D_\beta \tag{57d}$$

Using $\Lambda_\beta (P^b_{\alpha\beta})^T = P^T_{\alpha\beta} \Lambda_\alpha$ and $\tilde{G}_\alpha = \Lambda_\alpha G_\alpha$, we could simplify Eqn. 57 as follows:

$$F_\alpha = P_{\alpha\beta} F_\beta W_{\beta\alpha} \circ D_\alpha \tag{58a}$$
$$\tilde{G}_\beta = \left(P_{\alpha\beta}\right)^T \tilde{G}_\beta W^T_{\beta\alpha} \circ D_\beta \tag{58b}$$

Therefore, using the fact that $\sum_{j \in \mathrm{pa}(k)} = \sum_{\alpha \in \mathrm{pa}(\beta)} \sum_{j \in \alpha}$ (where $\beta = \mathrm{rf}(k)$) and $\sum_{k \in \mathrm{ch}(j)} = \sum_{\beta \in \mathrm{ch}(\alpha)} \sum_{k \in \beta}$ (where $\alpha = \mathrm{rf}(j)$), and group all nodes that share the receptive field together, we have:

$$F_\alpha = D_\alpha \circ \sum_{\beta \in \mathrm{ch}(\alpha)} P_{\alpha\beta} F_\beta W_{\beta\alpha} \tag{59a}$$
$$\tilde{G}_\beta = D_\beta \circ \sum_{\alpha \in \mathrm{pa}(\beta)} P^T_{\alpha\beta} \tilde{G}_\alpha W^T_{\beta\alpha} \tag{59b}$$

For the gradient update rule, from Eqn. 2 notice that:

$$\Delta w_{jk} = \mathbb{E}_x \left[f_k(x) g_j(x) \right] \tag{60}$$
$$= \int f_k(x) g_j(x) \mathbb{P}(x) \mathrm{d}x \tag{61}$$
$$= \int f_k(x) g_j(x) \sum_{z_\alpha} \mathbb{P}(x \mid z_\alpha) \mathbb{P}(z_\alpha) \mathrm{d}x \tag{62}$$
$$= \sum_{z_\alpha} \int f_k(x) g_j(x) \mathbb{P}(x \mid z_\alpha) \mathbb{P}(z_\alpha) \mathrm{d}x \tag{63}$$

We assume decorrelation so we have:

$$\Delta w_{jk} = \sum_{z_\alpha} \mathbb{E}_{X \mid z_\alpha} \left[f_k(x) \right] g_j(z_\alpha) \mathbb{P}(z_\alpha) \tag{64}$$
$$= \sum_{z_\alpha} \mathbb{E}_{X_k \mid z_\alpha} \left[f_k(x_k) \right] \tilde{g}_j(z_\alpha) \tag{65}$$

For $\mathbb{E}_{X_k|z_\alpha}[f_k(x_k)]$, again we use focus of knowledge:

$$\mathbb{E}_{X_k|z_\alpha}[f_k(x_k)] = \int f_k(x_k)\mathbb{P}(x_k|z_\alpha)\mathrm{d}x_k \tag{66}$$

$$= \sum_{z_\beta}\int f_k(x_k)\mathbb{P}(x_k|z_\alpha, z_\beta)\mathbb{P}(z_\beta|z_\alpha)\mathrm{d}x_k \tag{67}$$

$$= \sum_{z_\beta}\int f_k(x_k)\mathbb{P}(x_k|z_\beta)\mathbb{P}(z_\beta|z_\alpha)\mathrm{d}x_k \tag{68}$$

$$= \sum_{z_\beta} f_k(z_\beta)\mathbb{P}(z_\beta|z_\alpha) \tag{69}$$

Put them together and we have:

$$\Delta w_{jk} = \sum_{z_\alpha}\sum_{z_\beta} f_k(z_\beta)\tilde{g}_j(z_\alpha)\mathbb{P}(z_\beta|z_\alpha) = \mathbb{E}_{z_\alpha,z_\beta}[f_k(z_\beta)g_j(z_\alpha)] \tag{70}$$

Write it in concise matrix form and we get:

$$\Delta W_{\beta\alpha} = (P_{\alpha\beta}F_\beta)^T\tilde{G}_\alpha \tag{71}$$

□

7.5 Batch Norm as a projection

Theorem 5 (Backpropagation of Batch Norm)**.** *For a top-down gradient* $\mathbf{g}$*, BN layer gives the following gradient update (*$P^\perp_{\mathbf{f},\mathbf{1}}$ *is the orthogonal complementary projection of subspace* $\{\mathbf{f},\mathbf{1}\}$*):*

$$\mathbf{g_f} = J^{BN}(\mathbf{f})\mathbf{g} = \frac{c_1}{\sigma}P^\perp_{\mathbf{f},\mathbf{1}}\mathbf{g}, \quad \mathbf{g_c} = S(\mathbf{f})^T\mathbf{g} \tag{72}$$

Proof. We denote pre-batchnorm activations as $f^{(i)} = f_j(x_i)$ $(i = 1\dots N)$. In Batch Norm, $f^{(i)}$ is whitened to be $\tilde{f}^{(i)}$, then linearly transformed to yield the output $\bar{f}^{(i)}$:

$$\hat{f}^{(i)} = f^{(i)} - \mu, \quad \tilde{f}^{(i)} = \hat{f}^{(i)}/\sigma, \quad \bar{f}^{(i)} = c_1\tilde{f}^{(i)} + c_0 \tag{73}$$

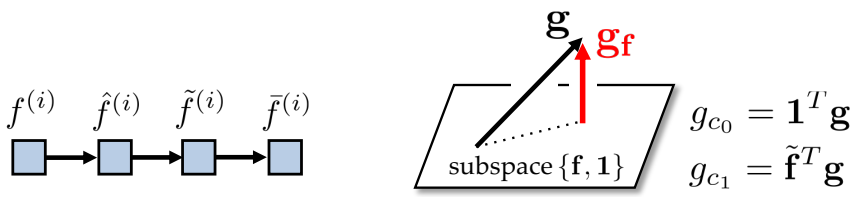
where $\mu = \frac{1}{N}\sum_i f^{(i)}$ and $\sigma^2 = \frac{1}{N}\sum_i (f^{(i)} - \mu)^2$ and c_1, c_0 are learnable parameters.

While in the original batch norm paper, the weight update rules are super complicated and unintuitive (listed here for a reference):

$$\begin{aligned}
\frac{\partial\ell}{\partial\widehat{x}_i} &= \frac{\partial\ell}{\partial y_i}\cdot\gamma \\
\frac{\partial\ell}{\partial\sigma^2_{\mathcal{B}}} &= \textstyle\sum_{i=1}^m \frac{\partial\ell}{\partial\widehat{x}_i}\cdot(x_i - \mu_{\mathcal{B}})\cdot\frac{-1}{2}(\sigma^2_{\mathcal{B}} + \epsilon)^{-3/2} \\
\frac{\partial\ell}{\partial\mu_{\mathcal{B}}} &= \left(\textstyle\sum_{i=1}^m \frac{\partial\ell}{\partial\widehat{x}_i}\cdot\frac{-1}{\sqrt{\sigma^2_{\mathcal{B}}+\epsilon}}\right) + \frac{\partial\ell}{\partial\sigma^2_{\mathcal{B}}}\cdot\frac{\sum_{i=1}^m -2(x_i-\mu_{\mathcal{B}})}{m} \\
\frac{\partial\ell}{\partial x_i} &= \frac{\partial\ell}{\partial\widehat{x}_i}\cdot\frac{1}{\sqrt{\sigma^2_{\mathcal{B}}+\epsilon}} + \frac{\partial\ell}{\partial\sigma^2_{\mathcal{B}}}\cdot\frac{2(x_i-\mu_{\mathcal{B}})}{m} + \frac{\partial\ell}{\partial\mu_{\mathcal{B}}}\cdot\frac{1}{m} \\
\frac{\partial\ell}{\partial\gamma} &= \textstyle\sum_{i=1}^m \frac{\partial\ell}{\partial y_i}\cdot\widehat{x}_i \\
\frac{\partial\ell}{\partial\beta} &= \textstyle\sum_{i=1}^m \frac{\partial\ell}{\partial y_i}
\end{aligned}$$

Figure 6: Original BN rule.

It turns out that with vector notation, the update equations have a compact vector form with clear geometric meaning.



(a) Sub-layers in BN layer (b) Projected Gradient

Figure 7: Analysis of Batch Normalization. **(a)** A Batch Normalization layer could be decomposed into three sublayers (zero-mean, unit-variance, affine). **(b)** The down gradient $\mathbf{g_f}$ is a projection of input gradient $\mathbf{g}$ onto the orthogonal complementary space spanned by $\{\mathbf{f}, \mathbf{1}\}$.

To achieve that, we first write down the vector form of forward pass of batch normalization:

$$\hat{\mathbf{f}} = P_1^{\perp}\mathbf{f}, \quad \tilde{\mathbf{f}} = \hat{\mathbf{f}}/\|\hat{\mathbf{f}}\|_{\text{uni}}, \quad \bar{\mathbf{f}} = c_1\tilde{\mathbf{f}} + c_0\mathbf{1} = S(\mathbf{f})\mathbf{c} \tag{74}$$

where $\mathbf{f}$, $\hat{\mathbf{f}}$, $\tilde{\mathbf{f}}$ and $\bar{\mathbf{f}}$ are vectors of size N, $P_1^{\perp} \equiv I - \frac{\mathbf{1}\mathbf{1}^T}{N}$ is the projection matrix that centers the data, $\sigma = \|\mathbf{f}\|_{\text{uni}} = \frac{1}{\sqrt{N}}\|\mathbf{f}\|_2$ and $\mathbf{c} \equiv [c_1, c_0]^T$ are the parameters in Batch Normalization and $S(\mathbf{f}) \equiv [\tilde{\mathbf{f}}, \mathbf{1}]$ is the standardized data. Note that $S(\mathbf{f})^T S(\mathbf{f}) = N \cdot I_2$ (I_2 is 2-by-2 identity matrix) and thus $S(\mathbf{x})$ is an column-orthogonal N-by-2 matrix. If we put everything together, then we have:

$$BN(\mathbf{f}) = c_1 \frac{P_1^{\perp}\mathbf{f}}{\|P_1^{\perp}\mathbf{f}\|_{\text{uni}}} + c_0\mathbf{1} \tag{75}$$

Using this notation, we can compute the Jacobian of batch normalization layer. Specifically, for any vector $\mathbf{f}$, we have:

$$\frac{\mathrm{d}\left(\frac{\mathbf{f}}{\|\mathbf{f}\|}\right)}{\mathrm{d}\mathbf{f}} = \frac{1}{\|\mathbf{f}\|}\left(I - \frac{\mathbf{f}\mathbf{f}^T}{\|\mathbf{f}\|^2}\right) = \frac{1}{\|\mathbf{f}\|}P_{\mathbf{f}}^{\perp} \tag{76}$$

where $P_{\mathbf{f}}^{\perp}$ projects a vector into the *orthogonal complementary* space of $\mathbf{f}$. Therefore we have:

$$J^{BN}(\mathbf{f}) = \frac{\mathrm{d}\bar{\mathbf{f}}}{\mathrm{d}\mathbf{f}} = c_1\frac{\mathrm{d}\tilde{\mathbf{f}}}{\mathrm{d}\mathbf{f}} = c_1\frac{\mathrm{d}\hat{\mathbf{f}}}{\mathrm{d}\mathbf{f}}\frac{\mathrm{d}\tilde{\mathbf{f}}}{\mathrm{d}\hat{\mathbf{f}}} = \frac{c_1}{\sigma}P_{\tilde{\mathbf{f}},\mathbf{1}}^{c} \tag{77}$$

where $P_{\tilde{\mathbf{f}},\mathbf{1}}^{\perp} = I - \frac{S(\mathbf{f})S(\mathbf{f})^T}{N}$ is a symmetric projection matrix that projects the input gradient to the orthogonal complement space spanned by $\tilde{\mathbf{x}}$ and $\mathbf{1}$ (Fig. 2(b)). Note that the space spanned by $\tilde{\mathbf{f}}$ and $\mathbf{1}$ is also the space spanned by $\mathbf{f}$ and $\mathbf{1}$, since $\tilde{\mathbf{f}} = (\mathbf{f} - \mu\mathbf{1})/\sigma$ can be represented linearly by $\mathbf{f}$ and $\mathbf{1}$. Therefore $P_{\tilde{\mathbf{f}},\mathbf{1}}^{\perp} = P_{\mathbf{f},\mathbf{1}}^{\perp}$.

An interesting property is that since $BN(\mathbf{f})$ returns a vector in the subspace of $\mathbf{f}$ and $\mathbf{1}$, for the N-by-N Jacobian matrix of Batch Normalization, we have:

$$J^{BN}(\mathbf{f})BN(\mathbf{f}) = J^{BN}(\mathbf{f})\mathbf{1} = J^{BN}(\mathbf{f})\mathbf{f} = \mathbf{0} \tag{78}$$

Following the backpropagation rule, we get the following gradient update for batch normalization. If $\mathbf{g} = \partial L/\partial\bar{\mathbf{f}}$ is the gradient from top, then

$$\mathbf{g_c} = S(\mathbf{x})^T\mathbf{g}, \qquad \mathbf{g_f} = J^{BN}(\mathbf{f})\mathbf{g} \tag{79}$$

Therefore, any gradient (vector of size N) that is back-propagated to the input of BN layer will be automatically orthogonal to that activation (which is also a vector of size N). □

7.5.1 Conserved Quantity in Batch Normalization for ReLU activation

One can find an interesting quantity, by multiplying $g_j(x)$ on both side of the forward equation in Eqn. 1 and taking expectation:

$$\mathbb{E}_x\left[g_j f_j\right] = \mathbb{E}_x\left[\sum_{k\in\text{ch}(j)} w_{jk} f_k g_j\right] = \sum_{k\in\text{ch}(j)} w_{jk}\Delta w_{jk} \tag{80}$$

Using the language of differential equation, we know that:

$$\int_0^t \mathbb{E}_x \left[g_j^{(t')} f_j^{(t')} \right] \mathrm{d}t' = E_j(t) - E_j(0) \tag{81}$$

where $E_j = \frac{1}{2} \sum_{k \in \mathrm{ch}(j)} w_{jk}^2 = \frac{1}{2} \|W_{j\cdot}\|^2$. If we place Batch Normalization layer just after ReLU activation and linear layer, by BN property, since $\mathbb{E}_x [g_j f_j] \equiv 0$ for all iterations, the row energy $E_j(t)$ of weight matrix W of the linear layer is conserved over time. This might be part of the reason why BN helps stabilize the training. Otherwise energy might "leak" from one layer to nearby layers.

7.6 NONLINEAR VERSUS LINEAR

Theorem 6 (Expressibility of ReLU Nonlinearity)**.** *Assuming $m_\alpha = n_\alpha = \mathcal{O}(\exp(\mathrm{sz}(\alpha)))$, where $\mathrm{sz}(\alpha)$ is the size of receptive field of α. If each $P_{\alpha\beta}$ is all-vert, then: (ω is top-level receptive field)*

$$\min_W Loss_{\mathrm{ReLU}}(W) = 0, \quad \min_W Loss_{\mathrm{Linear}}(W) = \mathcal{O}(\exp(\mathrm{sz}(\omega))) \tag{82}$$

Here we define $Loss(W) \equiv \|F_\omega - I\|_F^2$.

Proof. We prove that in the case of nonlinearity, there exists a weight so that the activation $F_\alpha = I$ for all α. We prove by induction. The base case is trivial since we already know that $F_\alpha = I$ for all leaf regions.

Suppose $F_\beta = I$ for any $\beta \in \mathrm{ch}(\alpha)$. Since $P_{\alpha\beta}$ is all-vert, every row is a vertex of the convex hull, which means that for i-th row p_i, there exists a weight $\mathbf{w}_i$ and b_i so that $\mathbf{w}_i^T p_i + b_i = 1/|\mathrm{ch}(\alpha)| > 0$ and $\mathbf{w}_i^T p_j + b_i < 0$ for $j \neq i$. Put these weights and biases together into $W_{\beta\alpha}$ and we have

$$F_\alpha^{\mathrm{raw}} = \sum_\beta P_{\alpha\beta} F_\beta W_{\beta\alpha} = \sum_\beta P_{\alpha\beta} W_{\beta\alpha} \tag{83}$$

All diagonal elements of F_α^{raw} are 1 while all off-diagonal elements are negative. Therefore, after ReLU, $F_\alpha = I$. Applying induction, we get $F_\omega = I$ and $G_\omega = I - F_\omega = 0$. Therefore, $Loss_{\mathrm{ReLU}}(W) = \|G_\omega\|_F^2 = 0$.

In the linear case, we know that $\mathrm{rank}(F_\alpha) \leq \sum_\beta \mathrm{rank}(P_{\alpha\beta} F_\beta W_{\beta\alpha}) \leq \sum_\beta \mathrm{rank}(F_\beta)$, which is on the order of the size $\mathrm{sz}(\alpha)$ of α's receptive field (Note that the constant relies on the overlap between receptive fields). However, at the top-level, $m_\omega = n_\omega = \mathcal{O}(\exp(\mathrm{sz}(\omega)))$, i.e., the information contained in α is exponential with respect to the size of the receptive field. By Eckart–Young–Mirsky theorem, we know that there is a lower bound for low-rank approximation. Therefore, the loss for linear network $Loss_{\mathrm{linear}}$ is at least on the order of m_0, i.e., $Loss_{\mathrm{linear}} = \mathcal{O}(m_\omega)$. Note that this also works if we have BN layer in-between, since BN does a linear transform in the forward pass. □

Theorem 7. *The following two are correct:*

(1) If F is row full rank, then $\mathrm{vert}(PF) = \mathrm{vert}(P)$.

(2) PF is all-vert iff P is all-vert.

Proof. For (1), note that each row of PF is $p_i^T F$. If F is row full rank, then F has pseudo-inverse F' so that $FF' = I$. Therefore, if p_i is not a vertex:

$$p_i = \sum_{j \neq i} a_j p_j, \quad \sum_j a_j = 1, a_j \geq 0, \tag{84}$$

then $p_i^T F$ is also not a vertex and vice versa. Therefore, $\mathrm{vert}(PF) = \mathrm{vert}(P)$. (2) follows from (1). □

7.7 DISENTANGLED REPRESENTATION

We first start with two lemmas. Both of them have simple proofs.

Lemma 1. *Distribution representations have the following property:*

(1) If $F_\alpha^{(i)}$ is disentangled, $F_\alpha = \sum_i w_i F_\alpha^{(i)}$ is also disentangled.

(2) If F_α is disentangled and h is any per-column element-wise function, then $h(F_\alpha)$ is disentangled.

(3) If $F_\alpha^{(i)}$ are disentangled, h_i are per-column element-wise function, then $h_1(F_\alpha^{(1)}) \circ h_2(F_\alpha^{(2)}) \ldots \circ h_n(F_\alpha^{(n)})$ is disentangled.

Proof. (1) follows from properties of tensor product. For (2) and (3), note that the j-th column of F_α is $F_{\alpha,:j} = \mathbf{1} \otimes \ldots \mathbf{f}_j \ldots \otimes \mathbf{1}$, therefore $h^j(F_{\alpha,:j}) = \mathbf{1} \otimes \ldots h^j(\mathbf{f}_j) \ldots \otimes \mathbf{1}$, and $h_1^j(F_{\alpha,:j}^{(1)}) \circ h_2^j(F_{\alpha,:j}^{(2)}) = \mathbf{1} \otimes \ldots h_1^j(\mathbf{f}_j^{(1)}) \circ h_2^j(\mathbf{f}_j^{(2)}) \ldots \otimes \mathbf{1}$. □

Given one child $\beta \in \mathrm{ch}(\alpha)$, denote

$$P_{S_j} = P_{\alpha[j]\beta[S_j]} = \left[\mathbb{P}(z_{\beta[S_j]}|z_{\alpha[j]})\right] \tag{85}$$

$$\mathbf{w}_{S_j} = W_{\beta\alpha}[S_j, j] \tag{86}$$

$$\mathbf{p}_{\alpha[j]} = \left[\mathbb{P}(\alpha[j]=0), \mathbb{P}(\alpha[j]=1)\right]^T \tag{87}$$

We have $P_{S_j}\mathbf{1} = \mathbf{1}$ and $\mathbf{1}^T\mathbf{p}_{\alpha[j]} = 1$. Note here for simplicity, $\mathbf{1}$ represents all-one vectors of any length, determined by the context.

Since F_α and G_β are disentangled, their j-th column can be written as:

$$F_{\beta,:j} = \mathbf{1} \otimes \ldots \otimes \mathbf{f}_j \otimes \ldots \otimes \mathbf{1} \tag{88}$$

$$\tilde{G}_{\alpha,:j} = \mathbf{p}_{\alpha[1]} \otimes \ldots \otimes \tilde{\mathbf{g}}_j \otimes \ldots \otimes \mathbf{p}_{\alpha[n_\alpha]} \tag{89}$$

For simplicity, in the following proofs, we just show the case that $n_\alpha = 2$, $n_\beta = 3$, $z_\alpha = \left[z_{\alpha[1]}, z_{\alpha[2]}\right]$ and $S = \{S_1, S_2\} = \{\{1,2\},\{3\}\}$. We write $\mathbf{f}_{1,2} = [\mathbf{f}_1 \otimes \mathbf{1}, \mathbf{1} \otimes \mathbf{f}_2]$ as a 2-column matrix. The general case is similar and we omit here for brevity.

Theorem 8 (disentangled Forward)**.** *If for each $\beta \in \mathrm{ch}(\alpha)$, $P_{\alpha\beta}$ can be written as a tensor product $P_{\alpha\beta} = \bigotimes_i P_{\alpha[i]\beta[S_i^{\alpha\beta}]}$ where $\{S_i^{\alpha\beta}\}$ are $\alpha\beta$-dependent disjointed set, $W_{\beta\alpha}$ is separable with respect to $\{S_i^{\alpha\beta}\}$, F_β is disentangled, then F_α is also disentangled (with/without ReLU /Batch Norm).*

Proof. For a certain $\beta \in \mathrm{ch}(\alpha)$, we first compute the quantity $P_{\alpha\beta}F_\beta$:

$$P_{\alpha\beta}F_\beta = (P_{1,2} \otimes P_3)\left[\mathbf{f}_{1,2} \otimes \mathbf{1},\ \ \mathbf{1} \otimes \mathbf{f}_3\right] = \left[P_{1,2}\mathbf{f}_{1,2} \otimes \mathbf{1},\ \ \mathbf{1} \otimes P_3\mathbf{f}_3\right] \tag{90}$$

Therefore, the forward information sent from β to α is:

$$F_{\beta\to\alpha}^{\mathrm{raw}} = P_{\alpha\beta}F_\beta W_{\beta\alpha} = \left[P_{1,2}\mathbf{f}_{1,2} \otimes \mathbf{1},\ \ \mathbf{1} \otimes P_3\mathbf{f}_3\right] \begin{bmatrix} \mathbf{w}_{1,2} & 0 \\ 0 & \mathbf{w}_3 \end{bmatrix} \tag{91}$$

$$= \left[P_{1,2}\mathbf{f}_{1,2}\mathbf{w}_{1,2} \otimes \mathbf{1},\ \ \mathbf{1} \otimes P_3\mathbf{f}_3\mathbf{w}_3\right] \tag{92}$$

Note that both $P_{1,2}\mathbf{f}_{1,2}\mathbf{w}_{1,2}$ and $P_3\mathbf{f}_3\mathbf{w}_3$ are 2-by-1 vectors. Therefore, for each $\beta \in \mathrm{ch}(\alpha)$, $F_{\beta\to\alpha}^{\mathrm{raw}}$ is disentangled. By Lemma 1, both $F_\alpha^{\mathrm{raw}} = \sum_{\beta\in\mathrm{ch}(\alpha)} F_{\beta\to\alpha}^{\mathrm{raw}}$ and the nonlinear response F_α are disentangled. By Eqn. 12, the forward pass of Batch Norm is a per-column element-wise function, so BN also preserves disentangledness. □

Theorem 9 (Separable Weight Update)**.** *If $P_{\alpha\beta} = \bigotimes_i P_{\alpha[i]\beta[S_i]}$, both F_β and $\tilde{G}_\alpha$ are disentangled, $\mathbf{1}^T\tilde{G}_\alpha = \mathbf{0}$, then the gradient update $\Delta W_{\beta\alpha}$ is separable with respect to $\{S_i\}$.*

Proof. Following Eqn. 59 and Eqn. 90, we have:

$$\Delta W_{\beta\alpha} = \left(P_{\alpha\beta}F_\beta\right)^T \tilde{G}_\alpha \tag{93}$$

$$= \begin{bmatrix} (P_{1,2}\mathbf{f}_{1,2})^T \otimes \mathbf{1}^T \\ \mathbf{1}^T \otimes (P_3\mathbf{f}_3)^T \end{bmatrix} \left[\tilde{\mathbf{g}}_1 \otimes \mathbf{p}_{\alpha[2]},\ \ \mathbf{p}_{\alpha[1]} \otimes \tilde{\mathbf{g}}_2\right] \tag{94}$$

$$= \begin{bmatrix} (P_{1,2}\mathbf{f}_{1,2})^T\tilde{\mathbf{g}}_1 & (P_{1,2}\mathbf{f}_{1,2})^T\mathbf{p}_{\alpha[1]} \otimes \mathbf{1}^T\tilde{\mathbf{g}}_2 \\ \mathbf{1}^T\tilde{\mathbf{g}}_1 \otimes (P_3\mathbf{f}_3)^T\mathbf{p}_{\alpha[2]} & (P_3\mathbf{f}_3)^T\tilde{\mathbf{g}}_2 \end{bmatrix} \tag{95}$$

Since $\mathbf{1}^T \tilde{G}_\alpha = \mathbf{0}$, we have for any j, $\mathbf{1}^T \tilde{G}_{\alpha,:j} = 0$ and thus $\mathbf{1}^T \tilde{\mathbf{g}}_j = 0$. Therefore,

$$\Delta W_{\beta\alpha} = \mathrm{diag}\left((P_{1,2}\mathbf{f}_{1,2})^T \tilde{\mathbf{g}}_1,\ \ (P_3\mathbf{f}_3)^T \tilde{\mathbf{g}}_2\right) \tag{96}$$

which is separable with respect to S. In particular:

$$\Delta \mathbf{w}_{1,2} = (P_{1,2}\mathbf{f}_{1,2})^T \tilde{\mathbf{g}}_1, \quad \Delta \mathbf{w}_3 = (P_3\mathbf{f}_3)^T \tilde{\mathbf{g}}_2 \tag{97}$$

□

7.7.1 Discussion about backpropagation of disentangled gradient

One problem remains. If $\{\tilde{G}_\alpha\}_{\alpha\in\mathrm{pa}(\beta)}$ are all disentangled, whether $\tilde{G}_\beta$ is disentangled? We can try computing the following quality:

$$\begin{aligned}
\tilde{G}^{\mathrm{raw}}_{\alpha\to\beta} &= P^T_{\alpha\beta}\tilde{G}_\alpha W^T_{\beta\alpha} && (98)\\
&= \left(P^T_{1,2}\otimes P^T_3\right)\left[\tilde{\mathbf{g}}_1\otimes \mathbf{p}_{\alpha[2]},\ \ \mathbf{p}_{\alpha[1]}\otimes\tilde{\mathbf{g}}_2\right]\begin{bmatrix}\mathbf{w}^T_{1,2} & 0\\ 0 & \mathbf{w}^T_3\end{bmatrix} && (99)\\
&= \left[P^T_{1,2}\tilde{\mathbf{g}}_1\otimes \mathbf{p}_{\beta[3]},\ \ \mathbf{p}_{\beta[1,2]}\otimes P^T_3\tilde{\mathbf{g}}_2\right]\begin{bmatrix}\mathbf{w}^T_{1,2} & 0\\ 0 & \mathbf{w}^T_3\end{bmatrix} && (100)\\
&= \left[P^T_{1,2}\tilde{\mathbf{g}}_1\mathbf{w}^T_{1,2}\otimes \mathbf{p}_{\beta[3]},\ \ \mathbf{p}_{\beta[1,2]}\otimes P^T_3\tilde{\mathbf{g}}_2\mathbf{w}^T_3\right] && (101)
\end{aligned}$$

Note that here we use the following equality from total probability rule:

$$P^T_3\mathbf{p}_{\alpha[2]} = \mathbf{p}_{\beta[3]}, \quad P^T_{1,2}\mathbf{p}_{\alpha[1]} = \mathbf{p}_{\beta[1,2]} \tag{102}$$

where $\mathbf{p}_{\beta[1,2]}$ is a 4-by-1 vector:

$$\mathbf{p}_{\beta[1,2]} = \begin{bmatrix}\mathbb{P}(z_{\beta[1]}=0, z_{\beta[2]}=0)\\ \mathbb{P}(z_{\beta[1]}=0, z_{\beta[2]}=1)\\ \mathbb{P}(z_{\beta[1]}=1, z_{\beta[2]}=0)\\ \mathbb{P}(z_{\beta[1]}=1, z_{\beta[2]}=1)\end{bmatrix} \tag{103}$$

Note that the ordering of these joint probability corresponds to the column order of $P_{1,2}$.

Now with this example, we see that the backward case (Eqn. 102) is very different from the forward case (Eqn. 92), in which $\tilde{G}^{\mathrm{raw}}_{\alpha\to\beta}$ is no longer disentangled. Indeed, $P^T_{1,2}\tilde{\mathbf{g}}_1\mathbf{w}^T_{1,2}$ is a 2-column matrix and $\mathbf{p}_{\beta[1,2]}$ is not a rank-1 tensor anymore. Intuitively this makes sense, if two low-level attributes have very similar behaviors, there is no way to distinguish the two via backpropagation.

Note that we also *cannot* assume independence: $\mathbf{p}_{\beta[1,2]} = \mathbf{p}_{\beta[1]}\otimes\mathbf{p}_{\beta[2]}$ since the independence property is in general not carried from layer to layer.

For general cases, $\tilde{G}^{\mathrm{raw}}_{\alpha\to\beta}$ takes the following form:

$$\tilde{G}^{\mathrm{raw}}_{\alpha\to\beta} = \left[P^T_{S_1^{\alpha\beta}}\tilde{\mathbf{g}}^\alpha_1\mathbf{w}^T_{S_1^{\alpha\beta}}\otimes\bigotimes_{j=2}^{n_\alpha}\mathbf{p}_{\beta[S_j^{\alpha\beta}]},\quad \mathbf{p}_{\beta[S_1^{\alpha\beta}]}\otimes P^T_{S_2^{\alpha\beta}}\tilde{\mathbf{g}}^\alpha_2\mathbf{w}^T_{S_2^{\alpha\beta}}\otimes\bigotimes_{j=3}^{n_\alpha}\mathbf{p}_{\beta[S_j^{\alpha\beta}]},\quad \cdots\right] \tag{104}$$

One hope here is that if we consider $\sum_{\alpha\in\mathrm{pa}(\beta)}\tilde{G}^{\mathrm{raw}}_{\alpha\to\beta}$, the summation over parent α could lead to a better structure, even for individual α, $P^T_{S_1}\tilde{\mathbf{g}}^\alpha_1\mathbf{w}^T_{S_1}$ is not 1-order tensor. For example, if $S^{\alpha\beta}_j = S_j$, then for the first column in S_1, due to $\mathbf{1}^T\tilde{\mathbf{g}}_j = 0$, we know that:

$$\sum_{\alpha\in\mathrm{pa}(\beta)} P^T_{\alpha,S_1}\tilde{\mathbf{g}}^\alpha_1\mathbf{w}^T_{\alpha,S_1[1]} = \sum_{\alpha\in\mathrm{pa}(\beta)} c_\alpha(\mathbf{v}^+_{\alpha,S_1} - \mathbf{v}^-_{\alpha,S_1}) \tag{105}$$

where $\mathbf{v}^+_{\alpha,S_1} = \mathbb{P}(z_{\beta[S_1]}|z_{\alpha[1]}=1)$ and $\mathbf{v}^-_{\alpha,S_1} = \mathbb{P}(z_{\beta[S_1]}|z_{\alpha[1]}=0)$ and $P_{\alpha,S_i} = \begin{bmatrix}\mathbf{v}^-_{\alpha,S_1}\\ \mathbf{v}^+_{\alpha,S_1}\end{bmatrix}$. If each $\alpha\in\mathrm{pa}(\beta)$ is informative in a diverse way, and $|S_1|$ is relatively small (e.g., 4), then $\mathbf{v}^+_{\alpha,S_1} - \mathbf{v}^-_{\alpha,S_1} \neq \mathbf{0}$ and spans the probability space of dimension $2^{|S_1|}-1$. Then we can always find c_α (or equivalently, weights) so that Eqn. 105 becomes rank-1 tensor (or disentangled). Besides, the gating D_β, which is disentangled as it is an element-wise function of F_β, will also play a role in regularizing $\tilde{G}_\beta$.

We will leave this part to future work.