A Theoretical Framework for Deep and 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 bridges data distribution with gradient descent rules, favors disentangled representations and is compatible with common regularization techniques such as Batch Norm, after a novel discovery of its projection nature. The framework is built upon teacher-student setting, by projecting the student's forward/backward pass onto the teacher's computational graph. We do 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. 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, such a modeling misses two components: neither specific network structures nor input data distribution is considered. Both are critical 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 fail to recover 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 gradient forms and enable analysis of many properties such as global convergence. However, it is also nontrivial to extend such approaches to deep nonlinear neural networks that yield strong empirical performance.

In this paper, we propose a novel theoretical framework for deep and locally connected ReLU network that is applicable to general data distributions. Specifically, we embrace a teacher-student setting. The *teacher* computes classification labels via a computational graph that has local structures (e.g., CNN): intermediate variables in the graph, (called *summarization variables*), are computed from a subset of the input dimensions. The *student* network, with similar local structures, updates the weights to fit teacher's labels with gradient descent, without knowing the summarization variables.

One ultimate goal is to show that after training, each node in the student network is highly selective with respect to the summarization variable in the teacher. Achieving this goal will shed light to how the training of practically effective methods like CNN works, which remains a grand challenge. As a first step, we *reformulate* the forward/backward pass in gradient descent by marginalizing out the input data conditioned on the graph variables of the teacher at each layer. The reformulation has nice properties: **(1)** it relates data distribution with gradient update rules, **(2)** it is compatible with existing

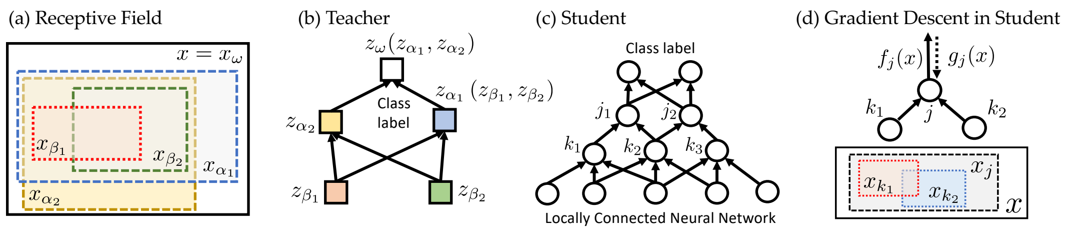


Figure 1: Problem setting. **(a)** We use Greek letters $\{\alpha, \beta, \dots \omega\}$ to represent receptive fields. Receptive fields form a hierarchy. The entire input is denoted as x (or x_ω). A local region of an input x is denoted as x_α. **(b)** For each region α, we have a latent multinomial discrete variable z_α which is computed from its immediate children $\{z_\beta\}_{\beta \in \mathrm{ch}(\alpha)}$. Given the input x, $z_\alpha = z_\alpha(x_\alpha)$ is a function of the image content x_α at α. Finally, z_ω at the top level is the *class label*. **(c)** A locally connected neural network is trained with pairs $(x, z_\omega(x))$, where $z_\omega(x)$ is the class label generated from the teacher. **(d)** For each node j, $f_j(x)$ is the activation while $g_j(x)$ is the back-propagated gradient, both as function of input x (and weights at different layers).

state-of-the-art regularization techniques such as Batch Normalization (Ioffe & Szegedy, 2015), and **(3)** it 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.

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), information bottleneck (Achille & Soatto, 2017; Tishby & Zaslavsky, 2015; Saxe et al., 2018), etc. 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, and relates it with data distribution, and **(3)** considers spatial locality of neurons, an important component in practical deep models.

2 Problem Setting

We consider multi-layer (deep) and locally connected 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 deterministically. It is hard to analyze y which does not have a structure (e.g., random labels). Here our analysis assumes the generation of y from x has a specific hierarchical structure. We use teacher-student setting to study the property: a student network learns teacher's label y via gradient descent, without knowing teacher's internal representations.

2.1 Receptive fields

An interesting characteristics in locally connected network is that each neuron only covers a fraction of the input dimension. Furthermore, for deep and locally connected network, neurons in the lower layer cover a small region while neurons in the upper layer cover a large region.

We use Greek letters $\{\alpha, \beta, \dots, \omega\}$ to represent receptive fields. For a receptive field α, x_α is the content in that region. We use ω to represent the entire image (Fig. 1(a)).

Receptive fields form a hierarchy: α is a parent of β, denoted as $\alpha \in \mathrm{pa}(\beta)$ or $\beta \in \mathrm{ch}(\alpha)$, if $\alpha \supseteq \beta$ and there exists no other receptive field $\gamma \notin \{\alpha, \beta\}$ so that $\alpha \supseteq \gamma \supseteq \beta$. Note that siblings can have substantial overlaps (e.g., β_1 and β_2 in Fig. 1(a)). With this partial ordering, we can attach layer number l to each receptive field: $\alpha \in \mathrm{pa}(\beta)$ implies $l(\beta) = l(\alpha) + 1$. For top-most layer (closest to classification label), $l = 0$ and for bottom-most layer, $l = L$.

For locally connected network, a neuron (or node) $j \in \alpha$ means its receptive field is α. Denote n_α as the number of nodes covering the same region (e.g., multi-channel case, Fig. 2(a)). The image content is $x_{\alpha(j)}$, abbreviated as x_j if no ambiguity. The parent j's receptive field covers its children's.

2.2 The teacher

We assume the label y of the input x is computed by a *teacher* in a bottom-up manner: for each region α, we compute a *summarization* variable z_α from the summarization variables of its children:

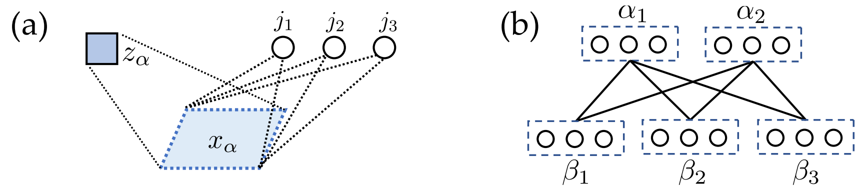


Figure 2: **(a)** Multiple nodes (neurons) share the same receptive field α. Note that n_α is the number of nodes sharing the receptive field α. **(b)** Grouping nodes with the same receptive fields together. By abuse of notation, α also represents the collection of all nodes with the same receptive field.

$z_\alpha = \phi_\alpha(\{z_\beta\}_{\beta\in\mathrm{ch}(\alpha)})$. This procedure is repeated until the top-level summarization z_ω is computed, which is the class label y. We denote $\boldsymbol{\phi} = \{\phi_\alpha\}$ as the collection of all summarization functions.

For convenience, we assume z_α be discrete variables that takes m_α possible values. Intuitively, m_α is exponential w.r.t the area of the receptive field $\mathrm{sz}(\alpha)$, for binary input, $m_\alpha \le 2^{\mathrm{sz}(\alpha)}$. We call a particular assignment of z_α, $z_\alpha = a$, an *event*. For the bottom-most layers, z is just the (discretized) value in each dimension.

At each stage, the upward function is deterministic but *lossy*: z_α does not contain all the information in $\{z_\beta\}$ for $\beta \in \mathrm{ch}(\alpha)$. Indeed, it keeps relevant information in the input region x_α with respect to the class label, and discards the irrelevant part. During training, all summarization variables $\mathcal{Z} = \{z_\alpha\}$ are unknown to the student, except for the label y.

Example of teacher networks. Locally connected network itself is a good example of teacher network, in which nodes of different channels located at one specific spatial location form some encoding of the variable z_α. Note that the relationship between a particular input x and the corresponding values of the summarization variable z at each layer is purely deterministic.

The reason why probabilistic quantities (e.g., $\mathbb{P}(z_\alpha)$ and $\mathbb{P}(z_\alpha|z_\beta)$) appear in our formulation, is due to marginalization over z (or x). This marginalization implicitly establishes a relationship between the conditional probabilities $\mathbb{P}(z_\alpha|z_\beta)$ and the input data distribution $\mathbb{P}(x)$. If we have specified $\mathbb{P}(z_\alpha|z_\beta)$ at each layer, then we implicitly specify a certain kind of data distribution $\mathbb{P}(x)$. Conversely, given a certain kind of $\mathbb{P}(x)$ and summarization function $\boldsymbol{\phi}$, we can compute $\mathbb{P}(z_\alpha|z_\beta)$ by sampling x, compute summarization variable z_α, and accumulate frequency statistics of $\mathbb{P}(z_\alpha|z_\beta)$. If there is an overlap between sibling receptive fields, then it is likely that some relationship among $\mathbb{P}(z_\alpha|z_\beta)$ might exist, which we leave for future work.

Although such an indirect specification may not be as intuitive and mathematically easy to deal with as common assumptions used in previous works (e.g., assuming Gaussian input (Tian, 2017; Du et al., 2018; Brutzkus & Globerson, 2017)), it gives much more flexibility of the distribution x and is more likely to be true empirically.

Comparison with top-down generative model. An alternative (and more traditional) way to specify data distribution is to use a top-down generative model: first sample the label y, then sample the latent variables z_α at each layer in a top-down manner, until the input layer. Marginalizing over all the latent variables z_α yields a class-conditioned data distribution $\mathbb{P}(x|y)$.

The main difficulty of this top-down modeling is that when the receptive fields α and α' of sibling latent variables overlap, the underlying graphical model becomes loopy. This makes the population loss function, which involves an integral over the input data x, very difficult to deal with. As a result, it is nontrivial to find a concise relationship between the parameters in the top-down modeling (e.g., conditional probability) and the optimization techniques applied to neural network (e.g., gradient descent). In contrast, as we will see in Sec. 3, our modeling naturally gives relationship between gradient descent rules and conditional probability between nearby summarization variables.

2.3 The student

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 (Fig. 1(d)). Note that both f_j and g_j are deterministic functions of the input x and label y, and are abbreviated as $f_j(x)$ and $g_j(x)$. [1].

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

For locally connected network, 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 since the label y is a deterministic (but unknown) function of x, for gradient we just write $g_j = g_j(x)$.

Marginalized Gradient. For locally connected network, the gradient g_j has some nice structures. From Eqn. 17 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*:

$$g_j(x_k) = \mathbb{E}_{x_{-k}|x_k}\left[g_j(x)\right] \tag{2}$$

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

$$g_k(x) = f'_k(x) \sum_{j \in \mathrm{pa}(k)} w_{jk} g_j(x) = f'_k(x_k) \sum_j w_{jk} g_j(x) \tag{3}$$

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

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

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 there is a recursive structure in marginal gradient: 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. 20 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 The Framework and its Goal

Let's first 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):

$$g_j(z_\alpha) \equiv \mathbb{E}_{X_j|z_\alpha}\left[g_j(X_j)\right] = \int g_j(x_j) \mathbb{P}(x_j|z_\alpha) \mathrm{d}x_j \tag{5}$$

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.

Goal. 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): $g_j(y = j) > 0$ and $g_j(y \neq j) < 0$, 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?

	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 $\mathbb{P}(z_\beta \vert z_\alpha)$ of events at group α and β

Table 1: Matrix Notation. See Eqn. 8 and Eqn. 59.

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. ***As a first step***, we build a theoretical framework that enables such analysis. We start with the relationship between neighboring layers:

Theorem 2 (Reformulation)**.** *For node j and k and their receptive field α and β. If the following two conditions holds:*

1) ***Onsite Conditional independence***. $\mathbb{P}(x_j|z_{\mathrm{rf}(j)}, z_{\ldots}) = \mathbb{P}(x_j|z_{\mathrm{rf}(j)})$.
2) ***Decorrelation***. *Given z_β, $g_k^{\mathrm{raw}}(\cdot)$ and $f_k^{\mathrm{raw}}(\cdot)$ are* uncorrelated *with $f'_k(\cdot)$:*

$$\mathbb{E}_{x_k|z_\beta}\left[f'_k g_k^{\mathrm{raw}}\right] = \mathbb{E}_{x_k|z_\beta}\left[f'_k\right]\mathbb{E}_{x_k|z_\beta}\left[g_k^{\mathrm{raw}}\right], \quad \mathbb{E}_{x_k|z_\beta}\left[f'_k f_k^{\mathrm{raw}}\right] = \mathbb{E}_{x_k|z_\beta}\left[f'_k\right]\mathbb{E}_{x_k|z_\beta}\left[f_k^{\mathrm{raw}}\right] \quad (6)$$

Then the following iterative equations hold:

$$f_j(z_\alpha) = f'_j(z_\alpha) \sum_{k\in\mathrm{ch}(j)} w_{jk}\mathbb{E}_{z_\beta|z_\alpha}\left[f_k(z_\beta)\right], \quad g_k(z_\beta) = f'_k(z_\beta) \sum_{j\in\mathrm{pa}(k)} w_{jk}\mathbb{E}_{z_\alpha|z_\beta}\left[g_j(z_\alpha)\right] \quad (7)$$

The reformulation becomes *exact* if z_α contains 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.*

While Thm. 3 holds in the ideal (and maybe trivial) case, both assumptions are still practically reasonable. For assumption (1), the main idea is that the image content x_α is most related to the summarization variable z_α located at the same receptive field α, and less related to others. On the other hand, assumptions (2) holds approximately if the summarization variable is fine-grained.

Intuitively, $\mathbb{P}(x_j|z_\alpha)$ is a distribution encoding how much information gets *lost* if we only know the factor z_α. Climbing up the ladder, more and more information is lost 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.

One key property of this formulation is that, it relates conditional probabilities $\mathbb{P}(z_\alpha, z_\beta)$, and thus input data distribution $\mathbb{P}(x)$ into the gradient descent rules. This is important since running backpropagation 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 backpropagation better.

Furthermore, the property of stochastic gradient descent (SGD) can be modeled as using an imperfect estimate $\tilde{\mathbb{P}}(z_\alpha, z_\beta)$ of the true probability $\mathbb{P}(z_\alpha, z_\beta)$ when running backpropagation. This is because each batch is a rough sample of the data distribution so the resulting $\mathbb{P}(z_\alpha, z_\beta)$ will also be different. This could also unite GD and SGD analysis.

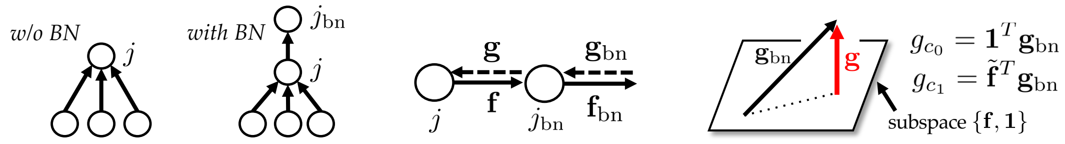
For boundary conditions, in the lowest level L, we could treat each input pixel (or a group of pixels) as a single event: $f_k(z_\beta) = \mathbb{I}\left[k = z_\beta\right]$. For top level, each node j corresponds to a class label j while the summarization variable z_α also take class labels: $g_j(z_\alpha) = a_1\mathbb{I}\left[j = z_\alpha\right] - a_2\mathbb{I}\left[j \neq z_\alpha\right]$.

If we group the nodes with the same reception field at the same level together (Fig. 2), we have the matrix form of Eqn. 7 ($\circ$ is element-wise multiplication):

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

$$F_\alpha = D_\alpha \circ \sum_{\beta\in\mathrm{ch}(\alpha)} P_{\alpha\beta} F_\beta W_{\beta\alpha}, \quad \tilde{G}_\beta = D_\beta \circ \sum_{\alpha\in\mathrm{pa}(\beta)} P_{\alpha\beta}^T \tilde{G}_\alpha W_{\beta\alpha}^T, \quad \Delta W_{\beta\alpha} = (P_{\alpha\beta}F_\beta)^T \tilde{G}_\alpha \quad (8)$$

See Tbl. 3 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 $n_\alpha = m_\alpha$.


w/o BN
j
with BN
j_{bn}
j
$\mathbf{g}$
$\mathbf{g}_{\mathrm{bn}}$
$\mathbf{f}$
$\mathbf{f}_{\mathrm{bn}}$
j
j_{bn}
$\mathbf{g}_{\mathrm{bn}}$
$\mathbf{g}$
$g_{c_0} = \mathbf{1}^T\mathbf{g}_{\mathrm{bn}}$
$g_{c_1} = \tilde{\mathbf{f}}^T\mathbf{g}_{\mathrm{bn}}$
subspace $\{\mathbf{f}, \mathbf{1}\}$

(a) Network before/after adding BN (b) Forward/backward in BN (c) Projected Gradient

Figure 3: Batch Normalization (BN) as a projection. **(a)** Add BN by inserting a new node j_{bn}. **(b)** Forward/backward pass in BN and relevant quantities. **(c)** The gradient $\mathbf{g}$ that is propagated down is a projection of input gradient $\mathbf{g}_{\mathrm{bn}}$ onto the orthogonal complementary space spanned by $\{\mathbf{f}, \mathbf{1}\}$.

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 $\mathbf{f} = [f_j(x_1), \ldots f_j(x_N)]$ where N is the batchsize. In Batch Norm, $\mathbf{f}$ is whitened to be $\tilde{\mathbf{f}}$, then linearly transformed to yield the output $\mathbf{f}_{\mathrm{bn}}$ (note that we omit node subscript j for clarity):

$$\hat{\mathbf{f}} = \mathbf{f} - \mu\mathbf{1}, \quad \tilde{\mathbf{f}} = \hat{\mathbf{f}}/\sigma, \quad \mathbf{f}_{\mathrm{bn}} = c_1\tilde{\mathbf{f}} + c_0 \tag{9}$$

where $\mu = \frac{1}{N}\mathbf{f}^T\mathbf{1}$ and $\sigma^2 = \frac{1}{N}\hat{\mathbf{f}}^T\hat{\mathbf{f}}$ and c_1, c_0 are learnable parameters.

The original Batch Norm paper (Ioffe & Szegedy, 2015) derives complicated and unintuitive weight update rules. With vector notation, the update has a compact form with a clear geometric meaning.

Theorem 5 (Backpropagation of Batch Norm)**.** *For a top-down pre-BN gradient $\mathbf{g}_{\mathrm{bn}}$ (a vector of size N-by-1,N is the batchsize), the gradient after passing BN layer is the following:*

$$\mathbf{g} = J^{BN}(\mathbf{f})\mathbf{g}_{\mathrm{bn}} = \frac{c_1}{\sigma}P^{\perp}_{\mathbf{f},\mathbf{1}}\mathbf{g}_{\mathrm{bn}}, \quad \mathbf{g}_{\mathbf{c}} \equiv [g_{c_1}, g_{c_0}]^T = S(\mathbf{f})^T\mathbf{g}_{\mathrm{bn}} \tag{10}$$

Here $P^{\perp}_{\mathbf{f},\mathbf{1}}$ is the orthogonal complementary projection onto subspace $\{\mathbf{f}, \mathbf{1}\}$ and $S(\mathbf{f}) \equiv [\tilde{\mathbf{f}}, \mathbf{1}]$.

Intuitively, the back-propagated gradient $\mathbf{g}$ is zero-mean and perpendicular to the input activation $\mathbf{f}$ of BN layer, as illustrated in Fig. 3. 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

In our reformulation, we take the expectation of input x so there is no explicit notation of batch. However, we could regard each sample in the batch as i.i.d. samples from the data distribution $\mathbb{P}(x)$. Then the analysis of Batch Norm in Sec. 4.1 could be applied in the reformulation and yield similar results, using the quantity that $\mathbb{E}_x[f_j(x)] = \mathbb{E}_{z_\alpha}[f_j(z_\alpha)]$.

In this case, we have $\mu = \mathbb{E}_{z_\alpha}[f_j]$ and $\sigma^2 = \mathbb{E}_{z_\alpha}\left[(f_j(z_\alpha) - \mu)^2\right]$, and $J^{BN}(\mathbf{f}) = \frac{c_1}{\sigma}P^{\perp;z_\alpha}_{\mathbf{f},\mathbf{1}}$. Note that the projection matrix $P^{\perp;z_\alpha}_{\mathbf{f},\mathbf{1}}$ is under the new inner product $\langle f_j, g_j\rangle_{z_\alpha} = \mathbb{E}_{z_\alpha}[f_j(z_\alpha)g_j(z_\alpha)]$ and norm $\|f\|_{z_\alpha} = \langle f, f\rangle^{1/2}_{z_\alpha}$. In comparison, Sec. 4.1 is a special case with $\mathbb{P}(z_\alpha) = \frac{1}{N}\sum_{i=1}^N \delta(z_\alpha - \phi_\alpha(x_i))$, where $x_1, \ldots, x_N$ are the batch samples.

One consequence is that for $\tilde{G}_\alpha$, we have $\mathbf{1}^T\tilde{G}_\alpha = [\mathbb{E}_{z_\alpha}[g_1(z_\alpha)], \ldots, \mathbb{E}_{z_\alpha}[g_{n_\alpha}(z_\alpha)]] = \mathbf{0}$ since $g_j(\cdot)$ is in the null space of $\mathbf{1}$ under the inner product $\langle\cdot,\cdot\rangle_{z_\alpha}$. This property will be used in Sec. 5.2.

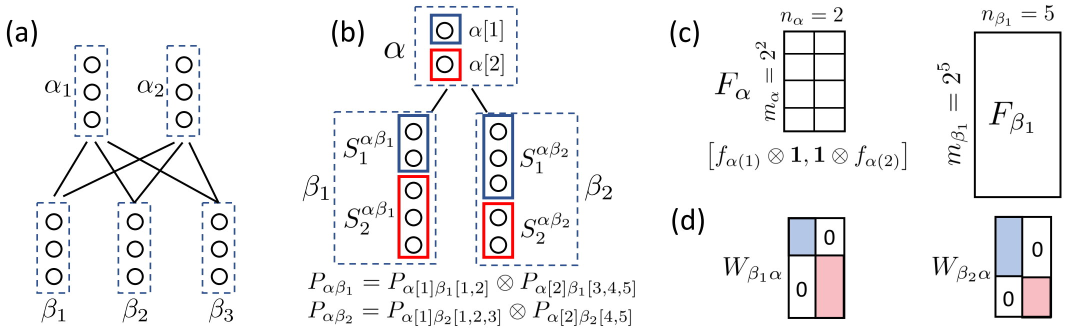


Figure 4: Disentangled representation. **(a)** Nodes are grouped according to regions. **(b)** An example of one parent region α (2 nodes) and two child regions β_1 and β_2 (5 nodes each). We assume factorization property of data distribution P. **(c)** disentangled activations, **(d)** Separable weights.

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* $\mathrm{Conv}(P) = \{P\mathbf{a}, \mathbf{a} \in \mathbf{\Delta}^{n-1}\}$, *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)*

$$\min_W Loss_{\text{ReLU}}(W) = 0, \quad \min_W Loss_{\text{Linear}}(W) = \mathcal{O}(\exp(\mathrm{sz}(\omega))) \tag{11}$$

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: $z_\alpha = [z_{\alpha[1]}, z_{\alpha[2]}, \ldots, z_{\alpha[j]}]$, 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.

Symbols	Description
$z_{\alpha[j]}$	The j-th binary factor of region α. $z_{\alpha[j]}$ can take 0 or 1.
$\mathbf{p}_{\alpha[j]}$	2-by-1 marginal probability vector of binary factor $z_{\alpha[j]}$.
$\mathbf{f}_{\alpha[j]}$, $\mathbf{g}_{\alpha[j]}$, $\tilde{\mathbf{g}}_{\alpha[j]}$	The j-th column of F_α, G_α and $\tilde{G}_\alpha$ corresponding to j-th binary factor $z_{\alpha[j]}$.
1 / **0**	All-1 / All-0 vector. Its dimension depends on context.
$F_1 \otimes F_2$	Out (or tensor) product of F_1 and F_2. $\otimes_{k=1}^K F_k$ is the product of multiple F_k.
$\{S_i^{\alpha\beta}\}$	Disjoint collection of n_α indices sets. $S_i^{\alpha\beta}$ are the indices of downstream nodes in β to i-th binary factor in α (Fig. 4(b)).
$W_{\beta\alpha}[S_j^{\alpha\beta}, j]$	The j-th subcolumn of weight matrix $W_{\beta\alpha}$, whose rows are selected by $S_j^{\alpha\beta}$.

Table 2: Symbols used in Sec. 5.2.

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. 83) 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.

6 Explanation of common behaviors in Deep Learning

In the proposed formulation, the input x in Eqn. 7 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.

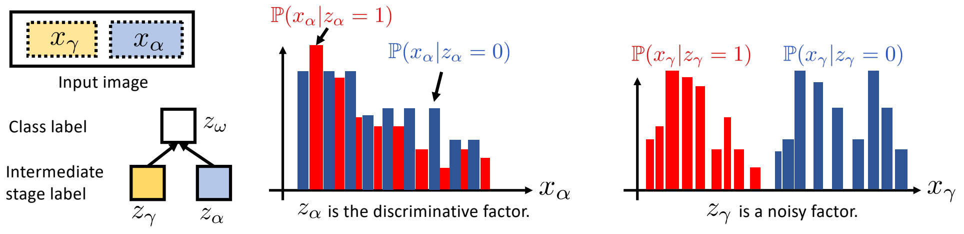


Figure 5: Overfitting Example

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:

$$\tilde{\mathbb{P}}(z_\omega = 1|z_\alpha = 1) = 1, \quad \tilde{\mathbb{P}}(z_\omega = 1|z_\alpha = 0) = 0 \tag{12}$$

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

$$\tilde{\mathbb{P}}(z_\omega = 1|z_\gamma = 1) = 0.5 + \epsilon, \quad \tilde{\mathbb{P}}(z_\omega = 1|z_\gamma = 0) = 0.5 - \epsilon \tag{13}$$

On the other hand, as shown in Fig. 5, $\mathbb{P}(x_\alpha|z_\alpha)$ 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 $\mathrm{rf}(j) = \alpha$, $f_j(z_\alpha) \approx \text{constant}$ (input almost indistinguishable):

$$\Delta w_j = \mathbb{E}_{z_\alpha}\left[f_j(z_\alpha)g_0(z_\alpha)\right] \approx 0 \tag{14}$$

where $g_0(z_\alpha) = \mathbb{E}_{z_\omega|z_\alpha}\left[g_0(z_\omega)\right] = \begin{cases} 1 & z_\alpha = 1 \\ -1 & z_\alpha = 0 \end{cases}$, 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,

$$\Delta w_k = \mathbb{E}_{z_\gamma}\left[f_j(z_\gamma)g_0(z_\gamma)\right] > 0 \tag{15}$$

where $g_0(z_\gamma) = \mathbb{E}_{z_\omega|z_\gamma}\left[g_0(z_\omega)\right] = \begin{cases} 2\epsilon & z_\gamma = 1 \\ -2\epsilon & z_\gamma = 0 \end{cases}$, 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.

7 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

Alessandro Achille and Stefano Soatto. Emergence of invariance and disentangling in deep representations. *arXiv preprint arXiv:1706.01350*, 2017.

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.

Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=ry_WPG-A-`.

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.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pp. 1–5. IEEE, 2015.

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.

8 Appendix

8.1 Problem Setting

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):

$$f_j(x) = f'_j(x) \sum_{k\in\mathrm{ch}(j)} w_{jk} f_k(x), \quad g_k(x) = f'_k(x) \sum_{j\in\mathrm{pa}(k)} w_{jk} g_j(x) \tag{16}$$

And the weight update for gradient descent is:

$$\Delta w_{jk} = \mathbb{E}_x\left[f_k(x) g_j(x)\right] \tag{17}$$

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.

8.2 Marginalized Gradient

Given the structure of locally connected network, the gradient g_j has some nice structures. From Eqn. 17 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*:

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

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

$$g_k(x) = f'_k(x) \sum_{j\in\mathrm{pa}(k)} w_{jk} g_j(x) = f'_k(x_k) \sum_j w_{jk} g_j(x) \tag{19}$$

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

$$g_k(x_k) = f'_k(x_k) g_k^{\mathrm{raw}}(x_k) = f'_k(x_k) \sum_{j\in\mathrm{pa}(k)} w_{jk} g_j(x_k) \tag{20}$$

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:

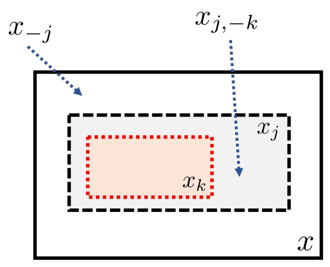


Figure 6: Notation used in Thm. 1.

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

$$g_j(x_k) = \mathbb{E}_{x_{j,-k}|x_k}\left[g_j(x_j)\right] \tag{21}$$

Proof. We have:

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

□

8.3 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:

$$f_j(z_\alpha) = f'_j(z_\alpha) \sum_{k\in\mathrm{ch}(j)} w_{jk}\mathbb{E}_{z_\beta|z_\alpha}\left[f_k(z_\beta)\right] \tag{22a}$$

$$g_k(z_\beta) = f'_k(z_\beta) \sum_{j\in\mathrm{pa}(k)} w_{jk}\mathbb{E}_{z_\alpha|z_\beta}\left[g_j(z_\alpha)\right] \tag{22b}$$

Proof. For Eqn. 22a, we have:

$$f_j^{\mathrm{raw}}(z_\alpha) = \int f_j^{\mathrm{raw}}(x)\mathbb{P}(x|z_\alpha)\mathrm{d}x \tag{23}$$

$$= \int f_j^{\mathrm{raw}}(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j \tag{24}$$

$$= \int \left(\sum_{k\in\mathrm{ch}(j)} w_{jk} f_k(x_k)\right)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j \tag{25}$$

And for each of the entry, we have:

$$\int f_k(x_k)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j = \int f_k(x_k)\mathbb{P}(x_k|z_\alpha)\mathrm{d}x_k \tag{26}$$

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

$$\mathbb{P}(x_k|z_\alpha) = \sum_{z_\beta} \mathbb{P}(x_k, z_\beta|z_\alpha) \tag{27}$$

$$= \sum_{z_\beta} \mathbb{P}(x_k|z_\beta, z_\alpha)\mathbb{P}(z_\beta|z_\alpha) \tag{28}$$

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

Therefore, following Eqn. 26, we have:

$$\int f_k(x_k)\mathbb{P}(x_k|z_\alpha)\mathrm{d}x_k = \int f_k(x_k)\sum_{z_\beta}\mathbb{P}(x_k|z_\beta)\mathbb{P}(z_\beta|z_\alpha)\mathrm{d}x_k \tag{30}$$

$$= \sum_{z_\beta}\left(\int f_k(x_k)\mathbb{P}(x_k|z_\beta)\mathrm{d}x_k\right)\mathbb{P}(z_\beta|z_\alpha) \tag{31}$$

$$= \sum_{z_\beta} f_k(z_\beta)\mathbb{P}(z_\beta|z_\alpha) \tag{32}$$

$$= \mathbb{E}_{z_\beta|z_\alpha}\left[f_k(z_\beta)\right] \tag{33}$$

Putting it back to Eqn. 25 and we have:

$$f_j^{\mathrm{raw}}(z_\alpha) = \sum_{k\in\mathrm{ch}(j)} w_{jk}\mathbb{E}_{z_\beta|z_\alpha}\left[f_k(z_\beta)\right] \tag{34}$$

For Eqn. 22b, similarly we have:

$$g_k^{\mathrm{raw}}(z_\beta) = \int g_k^{\mathrm{raw}}(x)\mathbb{P}(x|z_\beta)\mathrm{d}x \tag{35}$$

$$= \int \sum_{j\in\mathrm{pa}(k)} w_{jk} g_j(x)\mathbb{P}(x|z_\beta)\mathrm{d}x \tag{36}$$

Notice that we have:

$$\mathbb{P}(x|z_\beta) = \mathbb{P}(x_j|z_\beta)\mathbb{P}(x_{-j}|x_j, z_\beta) = \mathbb{P}(x_j|z_\beta)\mathbb{P}(x_{-j}|x_j) \tag{37}$$

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

$$\int g_j(x)\mathbb{P}(x|z_\beta)\mathrm{d}x = \int g_j(x)\mathbb{P}(x_j|z_\beta)\mathbb{P}(x_{-j}|x_j)\mathrm{d}x \tag{38}$$

$$= \int\left(\int g_j(x)\mathbb{P}(x_{-j}|x_j)\mathrm{d}x_{-j}\right)\mathbb{P}(x_j|z_\beta)\mathrm{d}x_j \tag{39}$$

$$= \int g_j(x_j)\mathbb{P}(x_j|z_\beta)\mathrm{d}x_j \tag{40}$$

Then we use the broadness of knowledge:

$$\mathbb{P}(x_j|z_\beta) = \sum_{z_\alpha}\mathbb{P}(x_j, z_\alpha|z_\beta) \tag{41}$$

$$= \sum_{z_\alpha}\mathbb{P}(x_j|z_\alpha, z_\beta)\mathbb{P}(z_\alpha|z_\beta) \tag{42}$$

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

Following Eqn. 40, we now have:

$$\int g_j(x)\mathbb{P}(x|z_\beta)\mathrm{d}x = \int g_j(x_j)\sum_{z_\alpha}\mathbb{P}(x_j|z_\alpha)\mathbb{P}(z_\alpha|z_\beta)\mathrm{d}x_j \tag{44}$$

$$= \sum_{z_\alpha}\left(\int g_j(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j\right)\mathbb{P}(z_\alpha|z_\beta) \tag{45}$$

$$= \sum_{z_\alpha} g_j(z_\alpha)\mathbb{P}(z_\alpha|z_\beta) \tag{46}$$

$$= \mathbb{E}_{z_\alpha|z_\beta}\left[g_j(z_\alpha)\right] \tag{47}$$

Putting it back to Eqn. 36 and we have:

$$g_k^{\text{raw}}(z_\beta) = \sum_{j\in\text{pa}(k)} w_{jk}\mathbb{E}_{z_\alpha|z_\beta}\left[g_j(z_\alpha)\right] \tag{48}$$

Using the definition of $g_k(z_\beta)$:

$$\begin{aligned} g_k(z_\beta) &= \int g_k(x_k)\mathbb{P}(x_k|z_\beta)\mathrm{d}x_k && (49)\\ &= \int f_k'(x_k)g_k^{\text{raw}}(x_k)\mathbb{P}(x_k|z_\beta)\mathrm{d}x_k && (50)\\ &= \mathbb{E}_{X_k|z_\beta}\left[f_k'(X_k)g_k^{\text{raw}}(X_k)\right] && (51)\end{aligned}$$

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

$$\mathbb{E}_{X_k|z_\beta}\left[f_k'g_k^{\text{raw}}\right] = \mathbb{E}_{X_k|z_\beta}\left[f_k'\right]\cdot\mathbb{E}_{X_k|z_\beta}\left[g_k^{\text{raw}}\right] \tag{52}$$

Similarly for $f_j(z_\alpha)$. □

8.4 Exactness of reformulation

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 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:

$$\mathbb{P}(x_j|z_\alpha) = \delta(x_j - \phi_j(z_\alpha)) \tag{53}$$

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 $\beta\in\text{ch}(\alpha)$, 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

$$\begin{aligned} \mathbb{E}_{X_j|z_\alpha}\left[h_1h_2\right] &= \int h_1(x_j)h_2(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j && (54)\\ &= \int h_1(x_j)h_2(x_j)\delta(x_j - \phi_j(z_\alpha))\mathrm{d}x_j && (55)\\ &= h_1(\phi_j(z_\alpha))h_2(\phi_j(z_\alpha)) && (56)\\ &= \int h_1(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j\int h_2(x_j)\mathbb{P}(x_j|z_\alpha)\mathrm{d}x_j && (57)\\ &= \mathbb{E}_{X_j|z_\alpha}\left[h_1\right]\mathbb{E}_{X_j|z_\alpha}\left[h_2\right] && (58)\end{aligned}$$

□

8.5 Matrix form

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

$$F_\alpha = D_\alpha\circ\sum_{\beta\in\text{ch}(\alpha)} P_{\alpha\beta}F_\beta W_{\beta\alpha},\quad \tilde{G}_\beta = D_\beta\circ\sum_{\alpha\in\text{pa}(\beta)} P_{\alpha\beta}^T\tilde{G}_\alpha W_{\beta\alpha}^T,\quad \Delta W_{\beta\alpha} = (P_{\alpha\beta}F_\beta)^T\tilde{G}_\alpha \tag{59}$$

	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 $\mathbb{P}(z_\beta \vert z_\alpha)$, $\mathbb{P}(z_\alpha \vert z_\beta)$ of events between group β and α.
Λ_α	m_α-by-m_α	Diagonal matrix encoding prior prob $\mathbb{P}(z_\alpha)$.

Table 3: Matrix Notation. See Eqn. 59.

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. 22 in the following matrix form:

$$F^{\text{raw}}_\alpha = P_{\alpha\beta}F_\beta W_{\beta\alpha} \tag{60a}$$

$$F_\alpha = F^{\text{raw}}_\alpha \circ D_\alpha \tag{60b}$$

$$G^{\text{raw}}_\beta = \left(P^b_{\alpha\beta}\right)^T G_\beta W^T_{\beta\alpha} \tag{60c}$$

$$G_\beta = G^{\text{raw}}_\beta \circ D_\beta \tag{60d}$$

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. 60 as follows:

$$F_\alpha = P_{\alpha\beta}F_\beta W_{\beta\alpha} \circ D_\alpha \tag{61a}$$

$$\tilde{G}_\beta = \left(P_{\alpha\beta}\right)^T \tilde{G}_\beta W^T_{\beta\alpha} \circ D_\beta \tag{61b}$$

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

$$F_\alpha = D_\alpha \circ \sum_{\beta\in\text{ch}(\alpha)} P_{\alpha\beta}F_\beta W_{\beta\alpha} \tag{62a}$$

$$\tilde{G}_\beta = D_\beta \circ \sum_{\alpha\in\text{pa}(\beta)} P^T_{\alpha\beta}\tilde{G}_\alpha W^T_{\beta\alpha} \tag{62b}$$

For the gradient update rule, from Eqn. 17 notice that:

$$\Delta w_{jk} = \mathbb{E}_x\left[f_k(x)g_j(x)\right] \tag{63}$$

$$= \int f_k(x)g_j(x)\mathbb{P}(x)\mathrm{d}x \tag{64}$$

$$= \int f_k(x)g_j(x)\sum_{z_\alpha}\mathbb{P}(x|z_\alpha)\mathbb{P}(z_\alpha)\mathrm{d}x \tag{65}$$

$$= \sum_{z_\alpha}\int f_k(x)g_j(x)\mathbb{P}(x|z_\alpha)\mathbb{P}(z_\alpha)\mathrm{d}x \tag{66}$$

We assume decorrelation so we have:

$$\Delta w_{jk} = \sum_{z_\alpha}\mathbb{E}_{X|z_\alpha}\left[f_k(x)\right]g_j(z_\alpha)\mathbb{P}(z_\alpha) \tag{67}$$

$$= \sum_{z_\alpha}\mathbb{E}_{X_k|z_\alpha}\left[f_k(x_k)\right]\tilde{g}_j(z_\alpha) \tag{68}$$

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

$$\mathbb{E}_{X_k|z_\alpha}\left[f_k(x_k)\right] = \int f_k(x_k)\mathbb{P}(x_k|z_\alpha)\mathrm{d}x_k \tag{69}$$

$$= \sum_{z_\beta}\int f_k(x_k)\mathbb{P}(x_k|z_\alpha, z_\beta)\mathbb{P}(z_\beta|z_\alpha)\mathrm{d}x_k \tag{70}$$

$$= \sum_{z_\beta}\int f_k(x_k)\mathbb{P}(x_k|z_\beta)\mathbb{P}(z_\beta|z_\alpha)\mathrm{d}x_k \tag{71}$$

$$= \sum_{z_\beta} f_k(z_\beta)\mathbb{P}(z_\beta|z_\alpha) \tag{72}$$

Put them together and we have:

$$\Delta w_{jk} = \sum_{z_\alpha}\sum_{z_\beta} f_k(z_\beta)\tilde{g}_j(z_\alpha)\mathbb{P}(z_\beta|z_\alpha) = \mathbb{E}_{z_\alpha, z_\beta}\left[f_k(z_\beta)g_j(z_\alpha)\right] \tag{73}$$

Write it in concise matrix form and we get:

$$\Delta W_{\beta\alpha} = (P_{\alpha\beta}F_\beta)^T\tilde{G}_\alpha \tag{74}$$

□

8.6 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* $\{\mathbf{f}, \mathbf{1}\}$*):*

$$\mathbf{g} = J^{BN}(\mathbf{f})\mathbf{g}_{\mathrm{bn}} = \frac{c_1}{\sigma}P^\perp_{\mathbf{f},\mathbf{1}}\mathbf{g}_{\mathrm{bn}}, \quad \mathbf{g}_{\mathbf{c}} = S(\mathbf{f})^T\mathbf{g}_{\mathrm{bn}} \tag{75}$$

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

$$\hat{f}^{(i)} = f^{(i)} - \mu, \quad \tilde{f}^{(i)} = \hat{f}^{(i)}/\sigma, \quad f^{(i)}_{\mathrm{bn}} = c_1\tilde{f}^{(i)} + c_0 \tag{76}$$

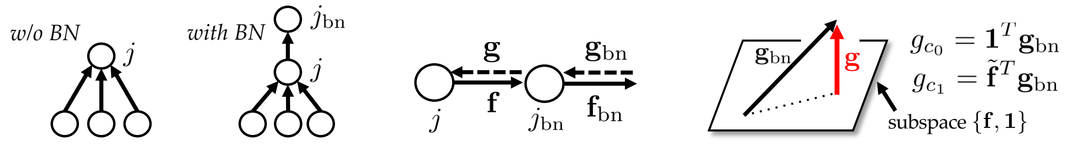
where $\mu = \frac{1}{N}\sum_i f^{(i)}$ and $\sigma^2 = \frac{1}{N}\sum_i(f^{(i)} - \mu)^2$ and c_1, c_0 are learnable parameters.

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

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

Figure 7: Original BN rule from (Ioffe & Szegedy, 2015).

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



(a) Network before/after adding BN (b) Forward/backward in BN (c) Projected Gradient

Figure 8: Batch Normalization (BN) as a projection. **(a)** Add BN by inserting a new node $j_{\rm bn}$. **(b)** Forward/backward pass in BN and relevant quantities. **(c)** The gradient $\mathbf{g}$ that is propagated down is a projection of input gradient $\mathbf{g}_{\rm bn}$ 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:

$$\hat{\mathbf{f}} = P_1^\perp \mathbf{f}, \quad \tilde{\mathbf{f}} = \hat{\mathbf{f}}/\|\hat{\mathbf{f}}\|_{\rm uni}, \quad \mathbf{f}_{\rm bn} = c_1 \tilde{\mathbf{f}} + c_0 \mathbf{1} = S(\mathbf{f})\mathbf{c} \tag{77}$$

where $\mathbf{f}$, $\hat{\mathbf{f}}$, $\tilde{\mathbf{f}}$ and $\mathbf{f}_{\rm bn}$ 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, $\sigma = \|\mathbf{f}\|_{\rm uni} = \frac{1}{\sqrt{N}}\|\mathbf{f}\|_2$ 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:

$$\mathbf{f}_{\rm bn} = c_1 \frac{P_1^\perp \mathbf{f}}{\|P_1^\perp \mathbf{f}\|_{\rm uni}} + c_0 \mathbf{1} \tag{78}$$

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

$$\frac{\mathrm{d}\left(\frac{\mathbf{f}}{\|\mathbf{f}\|}\right)}{\mathrm{d}\mathbf{f}} = \frac{1}{\|\mathbf{f}\|}\left(I - \frac{\mathbf{f}\mathbf{f}^T}{\|\mathbf{f}\|^2}\right) = \frac{1}{\|\mathbf{f}\|} P_{\mathbf{f}}^\perp \tag{79}$$

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

$$J^{BN}(\mathbf{f}) = \frac{\mathrm{d}\mathbf{f}_{\rm bn}}{\mathrm{d}\mathbf{f}} = c_1 \frac{\mathrm{d}\tilde{\mathbf{f}}}{\mathrm{d}\mathbf{f}} = c_1 \frac{\mathrm{d}\hat{\mathbf{f}}}{\mathrm{d}\mathbf{f}} \frac{\mathrm{d}\tilde{\mathbf{f}}}{\mathrm{d}\hat{\mathbf{f}}} = \frac{c_1}{\sigma} P^c_{\tilde{\mathbf{f}},\mathbf{1}} \tag{80}$$

where $P^\perp_{\tilde{\mathbf{f}},\mathbf{1}} = I - \frac{S(\mathbf{f})S(\mathbf{f})^T}{N}$ is a symmetric projection matrix that projects the input gradient to the orthogonal complement space spanned by $\tilde{\mathbf{x}}$ and $\mathbf{1}$ (Fig. 3(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 $P^\perp_{\tilde{\mathbf{f}},\mathbf{1}} = P^\perp_{\mathbf{f},\mathbf{1}}$.

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

$$J^{BN}(\mathbf{f})\mathbf{f}_{\rm bn} = J^{BN}(\mathbf{f})\mathbf{1} = J^{BN}(\mathbf{f})\mathbf{f} = \mathbf{0} \tag{81}$$

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

$$\mathbf{g}_{\mathbf{c}} = S(\mathbf{f})^T \mathbf{g}_{\rm bn}, \qquad \mathbf{g} = J^{BN}(\mathbf{f})\mathbf{g}_{\rm bn} \tag{82}$$

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). □

8.6.1 Batch Normalization in the Reformulation

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

$$\mu = \mathbb{E}_{z_\alpha}[f_j], \quad \sigma^2 = \mathbb{E}_{z_\alpha}\left[(f_j(z_\alpha) - \mu)^2\right], \quad J^{BN}(\mathbf{f}) = \frac{c_1}{\sigma} P^\perp_{\mathbf{f},\mathbf{1}} \tag{83}$$

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

8.6.2 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. 16 and taking expectation:

$$\mathbb{E}_x\left[g_j f_j\right] = \mathbb{E}_x\left[\sum_{k\in\mathrm{ch}(j)} w_{jk} f_k g_j\right] = \sum_{k\in\mathrm{ch}(j)} w_{jk}\Delta w_{jk} \tag{84}$$

Using the language of differential equation, we know that:

$$\int_0^t \mathbb{E}_x\left[g_j^{(t')} f_j^{(t')}\right]\mathrm{d}t' = E_j(t) - E_j(0) \tag{85}$$

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 $\mathbb{E}_x\left[g_j f_j\right] \equiv 0$ 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.

8.7 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.

8.7.1 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 $\mathbf{\Delta}^{n-1} = \{\mathbf{a}\in\mathbb{R}^n, a_i \ge 0, \sum_i a_i = 1\}$. 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* $\mathrm{vert}(P) = k \le m$*, 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)*

$$\min_W Loss_{\mathrm{ReLU}}(W) = 0, \quad \min_W Loss_{\mathrm{Linear}}(W) = \mathcal{O}(\exp(\mathrm{sz}(\omega))) \tag{86}$$

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 $j \ne i$. Put these weights and biases together into $W_{\beta\alpha}$ and we have

$$F_\alpha^{\mathrm{raw}} = \sum_\beta P_{\alpha\beta}F_\beta W_{\beta\alpha} = \sum_\beta P_{\alpha\beta}W_{\beta\alpha} \tag{87}$$

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 $\mathrm{rank}(F_\alpha) \le \sum_\beta \mathrm{rank}(P_{\alpha\beta}F_\beta W_{\beta\alpha}) \le \sum_\beta \mathrm{rank}(F_\beta)$, 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

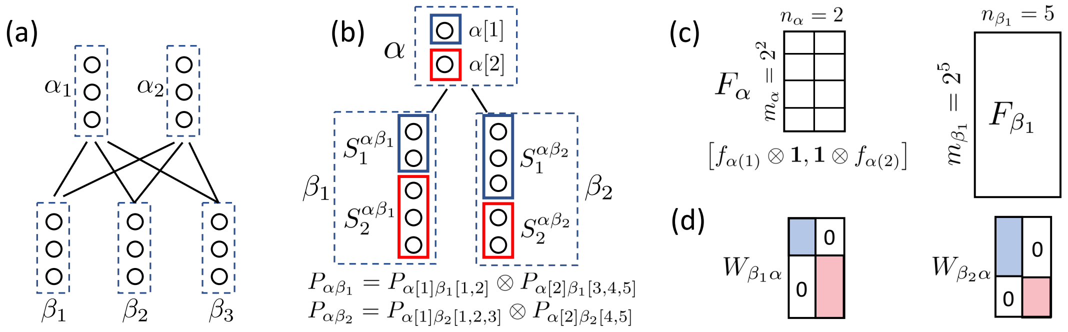


Figure 9: Disentangled representation. **(a)** Nodes are grouped according to regions. **(b)** An example of one parent region α (2 nodes) and two child regions β_1 and β_2 (5 nodes each). We assume factorization property of data distribution P. **(c)** disentangled activations, **(d)** Separable weights.

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 $Loss_{\text{linear}}$ is at least on the order of m_0, i.e., $Loss_{\text{linear}} = \mathcal{O}(m_\omega)$. Note that this also works if we have BN layer in-between, since BN does a linear transform in the forward pass. □

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.*

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:

$$p_i = \sum_{j \neq i} a_j p_j, \quad \sum_j a_j = 1, a_j \geq 0, \tag{88}$$

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

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.

8.7.2 Disentangled Representation

The analysis in the previous section 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: $z_\alpha = [z_{\alpha[1]}, z_{\alpha[2]}, \ldots, z_{\alpha[j]}]$, 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 $W_{\beta\alpha} = \mathrm{diag}\left(W_{\beta\alpha}[S_1^{\alpha\beta}, 1], W_{\beta\alpha}[S_2^{\alpha\beta}, 2], \ldots, W_{\beta\alpha}[S_{n_\alpha}^{\alpha\beta}, n_\alpha]\right)$.*

If the bottom activations are disentangled, by induction, all activations should be disentangled. The next question is whether gradient descent preserves such a structure. Here we provide a few theorems to discuss such issues.

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

$$P_{S_j} = P_{\alpha[j]\beta[S_j]} = \left[\mathbb{P}(z_{\beta[S_j]} | z_{\alpha[j]})\right] \quad (89)$$

$$\mathbf{w}_{S_j} = W_{\beta\alpha}[S_j, j] \quad (90)$$

$$\mathbf{p}_{\alpha[j]} = [\mathbb{P}(\alpha[j] = 0), \mathbb{P}(\alpha[j] = 1)]^T \quad (91)$$

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:

$$F_{\beta,:j} = \mathbf{1} \otimes \ldots \otimes \mathbf{f}_j \otimes \ldots \otimes \mathbf{1} \quad (92)$$

$$\tilde{G}_{\alpha,:j} = \mathbf{p}_{\alpha[1]} \otimes \ldots \otimes \tilde{\mathbf{g}}_j \otimes \ldots \otimes \mathbf{p}_{\alpha[n_\alpha]} \quad (93)$$

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 $P_{\alpha\beta}F_\beta$:

$$P_{\alpha\beta}F_\beta = (P_{1,2} \otimes P_3)\,[\mathbf{f}_{1,2} \otimes \mathbf{1},\ \ \mathbf{1} \otimes \mathbf{f}_3] = [P_{1,2}\mathbf{f}_{1,2} \otimes \mathbf{1},\ \ \mathbf{1} \otimes P_3\mathbf{f}_3] \quad (94)$$

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

$$F_{\beta\to\alpha}^{\mathrm{raw}} = P_{\alpha\beta}F_\beta W_{\beta\alpha} = [P_{1,2}\mathbf{f}_{1,2} \otimes \mathbf{1},\ \ \mathbf{1} \otimes P_3\mathbf{f}_3] \begin{bmatrix} \mathbf{w}_{1,2} & 0 \\ 0 & \mathbf{w}_3 \end{bmatrix} \quad (95)$$

$$= [P_{1,2}\mathbf{f}_{1,2}\mathbf{w}_{1,2} \otimes \mathbf{1},\ \ \mathbf{1} \otimes P_3\mathbf{f}_3\mathbf{w}_3] \quad (96)$$

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 $\beta \in \mathrm{ch}(\alpha)$, $F^{\mathrm{raw}}_{\beta\to\alpha}$ is disentangled. By Lemma 1, both $F^{\mathrm{raw}}_{\alpha} = \sum_{\beta\in\mathrm{ch}(\alpha)} F^{\mathrm{raw}}_{\beta\to\alpha}$ and the nonlinear response F_α are disentangled. By Eqn. 83, the forward pass of Batch Norm is a per-column element-wise function, so BN also preserves disentangledness. $\square$

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. 62 and Eqn. 94, we have:

$$\begin{aligned}
\Delta W_{\beta\alpha} &= \left(P_{\alpha\beta}F_\beta\right)^T \tilde{G}_\alpha && (97)\\
&= \begin{bmatrix} (P_{1,2}\mathbf{f}_{1,2})^T \otimes \mathbf{1}^T \\ \mathbf{1}^T \otimes (P_3\mathbf{f}_3)^T \end{bmatrix} \left[\tilde{\mathbf{g}}_1 \otimes \mathbf{p}_{\alpha[2]},\ \ \mathbf{p}_{\alpha[1]} \otimes \tilde{\mathbf{g}}_2\right] && (98)\\
&= \begin{bmatrix} (P_{1,2}\mathbf{f}_{1,2})^T\tilde{\mathbf{g}}_1 & (P_{1,2}\mathbf{f}_{1,2})^T\mathbf{p}_{\alpha[1]} \otimes \mathbf{1}^T\tilde{\mathbf{g}}_2 \\ \mathbf{1}^T\tilde{\mathbf{g}}_1 \otimes (P_3\mathbf{f}_3)^T\mathbf{p}_{\alpha[2]} & (P_3\mathbf{f}_3)^T\tilde{\mathbf{g}}_2 \end{bmatrix} && (99)
\end{aligned}$$

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,

$$\Delta W_{\beta\alpha} = \mathrm{diag}\left((P_{1,2}\mathbf{f}_{1,2})^T\tilde{\mathbf{g}}_1,\ \ (P_3\mathbf{f}_3)^T\tilde{\mathbf{g}}_2\right) \tag{100}$$

which is separable with respect to S. In particular:

$$\Delta\mathbf{w}_{1,2} = (P_{1,2}\mathbf{f}_{1,2})^T\tilde{\mathbf{g}}_1, \quad \Delta\mathbf{w}_3 = (P_3\mathbf{f}_3)^T\tilde{\mathbf{g}}_2 \tag{101}$$

$\square$

8.7.3 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:

$$\begin{aligned}
\tilde{G}^{\mathrm{raw}}_{\alpha\to\beta} &= P^T_{\alpha\beta}\tilde{G}_\alpha W^T_{\beta\alpha} && (102)\\
&= \left(P^T_{1,2}\otimes P^T_3\right)\left[\tilde{\mathbf{g}}_1 \otimes \mathbf{p}_{\alpha[2]},\ \ \mathbf{p}_{\alpha[1]} \otimes \tilde{\mathbf{g}}_2\right] \begin{bmatrix} \mathbf{w}^T_{1,2} & 0 \\ 0 & \mathbf{w}^T_3 \end{bmatrix} && (103)\\
&= \left[P^T_{1,2}\tilde{\mathbf{g}}_1 \otimes \mathbf{p}_{\beta[3]},\ \ \mathbf{p}_{\beta[1,2]} \otimes P^T_3\tilde{\mathbf{g}}_2\right] \begin{bmatrix} \mathbf{w}^T_{1,2} & 0 \\ 0 & \mathbf{w}^T_3 \end{bmatrix} && (104)\\
&= \left[P^T_{1,2}\tilde{\mathbf{g}}_1\mathbf{w}^T_{1,2} \otimes \mathbf{p}_{\beta[3]},\ \ \mathbf{p}_{\beta[1,2]} \otimes P^T_3\tilde{\mathbf{g}}_2\mathbf{w}^T_3\right] && (105)
\end{aligned}$$

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

$$P^T_3\mathbf{p}_{\alpha[2]} = \mathbf{p}_{\beta[3]}, \quad P^T_{1,2}\mathbf{p}_{\alpha[1]} = \mathbf{p}_{\beta[1,2]} \tag{106}$$

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

$$\mathbf{p}_{\beta[1,2]} = \begin{bmatrix} \mathbb{P}(z_{\beta[1]}=0, z_{\beta[2]}=0) \\ \mathbb{P}(z_{\beta[1]}=0, z_{\beta[2]}=1) \\ \mathbb{P}(z_{\beta[1]}=1, z_{\beta[2]}=0) \\ \mathbb{P}(z_{\beta[1]}=1, z_{\beta[2]}=1) \end{bmatrix} \tag{107}$$

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. 106) is very different from the forward case (Eqn. 96), 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, $\tilde{G}^{\text{raw}}_{\alpha\to\beta}$ takes the following form:

$$\tilde{G}^{\text{raw}}_{\alpha\to\beta} = \left[P^T_{S_1^{\alpha\beta}} \tilde{\mathbf{g}}_1^\alpha \mathbf{w}^T_{S_1^{\alpha\beta}} \otimes \bigotimes_{j=2}^{n_\alpha} \mathbf{p}_{\beta[S_j^{\alpha\beta}]}, \quad \mathbf{p}_{\beta[S_1^{\alpha\beta}]} \otimes P^T_{S_2^{\alpha\beta}} \tilde{\mathbf{g}}_2^\alpha \mathbf{w}^T_{S_2^{\alpha\beta}} \otimes \bigotimes_{j=3}^{n_\alpha} \mathbf{p}_{\beta[S_j^{\alpha\beta}]}, \quad \ldots \right] \tag{108}$$

One hope here is that if we consider $\sum_{\alpha\in\text{pa}(\beta)} \tilde{G}^{\text{raw}}_{\alpha\to\beta}$, the summation over parent α could lead to a better structure, even for individual α, $P^T_{S_1}\tilde{\mathbf{g}}_1^\alpha \mathbf{w}^T_{S_1}$ is not 1-order tensor. For example, if $S_j^{\alpha\beta} = S_j$, then for the first column in S_1, due to $\mathbf{1}^T\tilde{\mathbf{g}}_j = 0$, we know that:

$$\sum_{\alpha\in\text{pa}(\beta)} P^T_{\alpha,S_1}\tilde{\mathbf{g}}_1^\alpha \mathbf{w}^T_{\alpha,S_1[1]} = \sum_{\alpha\in\text{pa}(\beta)} c_\alpha(\mathbf{v}^+_{\alpha,S_1} - \mathbf{v}^-_{\alpha,S_1}) \tag{109}$$

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 $\alpha \in \text{pa}(\beta)$ 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. 109 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.