Distributions in Deep Neural Networks

Bumjin Park
bumjin@kaist.ac.kr

Abstract

The deep neural network is intractable to be analyzed as the number of parameters is huge and the training dynamics are stochastic. Despite the existence of difficulty, it is necessary to analyze the internal representation space for the debugging of models. In this work, we present the empirical results in distribution shifts of parameters and features of deep neural networks with several weight initialization methods. As the result we found that the mean of parameters is almost centered at zero while the variance increased for all layers after the training. In the case of features, they had higher variance for deeper layer compared to lower layers. In addition we analyze the gradients on the convolution features with mathematical formulation and present two statistics: emission and absorption of gradients for a single channel. The empirical result show that each channel has different variance of emission and absorption even in the same layer.

1 Introduction

A deep neural network is composed of a lot of parameters to be optimized by loss minimization. The parameters are updated in the stochastic manner and are changed by any loss if the gradient is not zero. In other words, the parameters vibrate in the training dynamics and the value of weights could be ambiguous. However, if the random signal is properly filtered out, the effects may not change the outcome results. That is, **the distractor is removed and the signal is preserved**.

Even though the signal is properly propagated to the output, the output still has randomness which is not filtered out. In addition, we could not certain whether the randomness comes from the signal or distractor and we poorly assume the distractor is removed by taking weight to the orthogonal direction. If the randomness prevails in the internal representation, it is hard for the researchers to analyze the randomness in the internal features.

The most hardness comes from the size of deep neural networks and the training dynamics that we could not tractably analyze. To alleviate the burden of analysis, we conduct two experiments on the randomness of deep neural networks. At first, we show the whole distribution of parameters and features for initialization and the end of the training so that we can see whether the whole distributions are changed or not. For the second experiment, we compute the randomness in gradients. Note that even though the gradient direction is deterministic, the randomness comes from the weights and features. Therefore, we check the distribution of the norm of gradients for each convolution channel.

2 Backgrounds

2.1 Initialization

Xavier keeps the variance after passing a single layer with Tanh activation function.

$$W \sim \mathcal{N}(0, \sigma^2) \tag{1}$$

$$\sigma = g \times \sqrt{\frac{2}{d_{in} + d_{out}}} \tag{2}$$

where g is the gain value for the given non-linearity function. For example, ReLU has gain $\sqrt{2}$. In forward, $Var(W) = 1/d_{in}$ occurs, and in backward, $Var(W) = 1/d_{out}$ occurs. In case of uniform distribution, it is defined by

$$W \sim \mathcal{U}(-a, a) \tag{3}$$

$$a = g \times \sqrt{\frac{6}{d_{in} + d_{out}}} \tag{4}$$

Kaiming is also known as He initialization, is used for ReLU activation. Kaiming initialization is defined by

$$W \sim \mathcal{N}(0, \sigma^2) \tag{5}$$

$$\sigma = g \times \sqrt{\frac{2}{d_{in}}} \tag{6}$$

for normal distribution and

$$W \sim \mathcal{U}(-a, a) \tag{7}$$

$$a = g \times \sqrt{\frac{6}{d_{in}}} \tag{8}$$

for uniform distribution.

Orthogonal initialization assumes that each weight vector (output of the neurons) is orthogonal to other weight. The representations can capture different signals and distractors more easily than non-orthogonal weight distributions.

Ones and Zeros initialization schemes assumes that all the weights are starting from the same value. In the case of Zeros, all the weights and bias are zero and the propagated values are also zero and no training happens. In the case of Ones, the output could be dependent to the input, however, as the all output dimensions have the same value, the output signal could be understood as a scalar.

2.2 Channel-wise Gradients

We measure how much gradient are emitted from the channel and how much information comes in the channel.

Proposition 1. *Let $\boldsymbol{a} \in \mathbb{R}^d$, $W \in \mathbb{R}^{c \times d}$, $\boldsymbol{v} = W\boldsymbol{a}$ be input signal, weight matrix, and output signal each. For the gradients from the upper layer $\frac{\partial L}{\partial \boldsymbol{v}} = \boldsymbol{g} \in \mathbb{R}^c$, the gradients on $\boldsymbol{a}_j$ is the linear sum of channels.*

$$\frac{\partial L}{\partial \boldsymbol{a}_j} = \frac{\partial L}{\partial \boldsymbol{v}} \cdot \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{a}_j} = \sum_{i=1}^{c} \boldsymbol{g}_i W_{ij} \tag{9}$$

Proof.

$$\frac{\partial L}{\partial \boldsymbol{v}} \cdot \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{a}_j} = (\boldsymbol{g}_1, \boldsymbol{g}_2, \cdots, \boldsymbol{g}_c)^T \cdot \frac{\partial}{\partial \boldsymbol{a}_j} \Big(\sum_{j'=1}^{d} W_{1j'} \boldsymbol{a}_{j'}, \sum_{j'=1}^{d} W_{2j'} \boldsymbol{a}_{j'}, \cdots, \sum_{j'=1}^{d} W_{cj'} \boldsymbol{a}_{j'} \Big) \tag{10}$$

$$= (\boldsymbol{g}_1, \boldsymbol{g}_2, \cdots, \boldsymbol{g}_c)^T \cdot (W_{1j}, W_{2j}, \cdots, W_{cj}) \tag{11}$$

$$= \sum_{i=1}^{c} \boldsymbol{g}_i W_{ij} \tag{12}$$

□

That is, the received gradient is the independent sum of all channels.

Proposition 2. *Let $A \in \mathbb{R}^{H_1 \times W_1}$ and $W \in \mathbb{R}^{c \times k_1 \times k_2}$ be a weight matrix and $\boldsymbol{v}_{ij} = \sum_{k_1=0}^{K-1} \sum_{k_2=0}^{K-1} \boldsymbol{w}_{k_1 k_2} A_{(i+k_1)(j+k_2)}$ the convolution of W on A where $\boldsymbol{w}_{k_1 k_2}$ is c-dimensional vector. The gradient from the upper layer is $\frac{\partial L}{\partial V} = G \in \mathbb{R}^{c \times H_2 \times H_3}$. Then the gradient on A_{pq} is the linear sum of channels*

$$\frac{\partial L}{\partial A_{pq}} = \sum_{i=1}^{c} \Big[\sum_{p-k_1}^{K} \sum_{q-k_2}^{K} \Big(G_{i,p-k_1,q-k_2} \cdot \boldsymbol{w}_{p-k_1,q-k_2}(i) \Big) \Big] \tag{13}$$

Proof.

$$\frac{\partial L}{\partial A_{pq}} = \frac{\partial L}{\partial V} \frac{\partial V}{\partial A_{pq}} \tag{14}$$

$$= \sum_{k_1}^{K} \sum_{k_2}^{K} \frac{\partial \boldsymbol{v}_{p-k_1,q-k_2}}{\partial A_{pq}} \tag{15}$$

$$= \sum_{k_1}^{K} \sum_{k_2}^{K} \Big[\sum_{i=1}^{c} \Big(G_{i,p-k_1,q-k_2} \cdot \boldsymbol{w}_{p-k_1,q-k_2}(i) \Big) \Big] \tag{16}$$

$$= \sum_{i=1}^{c} \Big[\sum_{p-k_1}^{K} \sum_{q-k_2}^{K} \Big(G_{i,p-k_1,q-k_2} \cdot \boldsymbol{w}_{p-k_1,q-k_2}(i) \Big) \Big] \tag{17}$$

Equation 16 comes from Proposition 2. □

From the Proposition 2.2, we know that the gradient towards the input A_{pq} is

$$\frac{\partial L}{\partial A_{pq}} = \sum_{i=1}^{c} \Big[\sum_{p-k_1}^{K} \sum_{q-k_2}^{K} \Big(G_{i,p-k_1,q-k_2} \cdot \boldsymbol{w}_{p-k_1,q-k_2}(i) \Big) \Big] = \sum_{i=1}^{c} LG_{pq}^{c}(x) \tag{18}$$

We define *Local Gradients (LG)* as the sum of local gradients from the inverted receptive field of a convolution. As the convolution layer is a channel to channel network, we can compute the local gradients from the upper channel c_1 in layer $l+1$ to the lower c_2 in layer l by $LG_{pq}^{c_1 \to c_2}$. When our interest is the channel statistics, we can ignore the index p, q by taking norm over the all region. To do so, we consider the pattern of $LG_{pq}^{c_1 \to c_2}$ over the image space by $LG^{c_1 \to c_2} = (LG_{1,1}^{c_1 \to c_2}, \cdots)$. Now we consider two types of statistics, *Emission* (Em^{c_1}) which considers how much gradient information is emitted from the channel and *Absorption* (Ab^{c_2}) which considers how much gradient information is coming to the channel.

$$Em^{c_1}(x) = \sum_{c2}^{C_{in}} ||LG^{c_1 \to c_2}(x)|| \tag{19}$$

$$Ab^{c_2}(x) = \sum_{c1}^{C_{out}} ||LG^{c_1 \to c_2}(x)|| \tag{20}$$

3 Experiments - First

To empirically analyze the distributions of parameters and features, we train a convolution neural network with 3 convolution layers on CIFAR10 dataset (see Table). Five initialization methods: orthogonal, Kaiming, Xavier, Ones, and Zeros are used for the comparison. In the case of Kaiming and Xavier, we sample weights from normal and uniform distributions. For training, Adam optimizer with learning rate 0.0001, and batch size 64 is used. We do not use weight decay in the optimization, as we are interested in the pure weight distributions.

Layer	Description
Convolution ($\star$)	(3,16,3,1,1)
ReLU	-
MaxPool	2
Convolution ($\star$)	(16,16,3,1,1)
ReLU	-
MaxPool	2
Convolution ($\star$)	(16,16,3,1,1)
ReLU	-
Flatten	-
Linear ($\star$)	(1024,512)
ReLU	-
Linear	(512,10)

Table 1: Model description. The solid lines indicate the end of blocks and the numbers in the convolution layers indicate (in-channels, out-channels, kernel size, stride, and padding). The distributions of ($\star$) symbol layers are used analyzed in this work.

Figure 1 shows the training loss for 50 epochs. Orthogonal initialization showed the fastest convergence followed by Kaiming initialization. Xavier showed the slower performance than Kaiming and the reason would be the variance of weights. As the Xavier as additional out dimension size in the denominator, the variance is lower than the variance of Kaiming. As the results the weights are more centered at zero and make less bias to the specific weights. In the case of Ones and Zeros, as noticed before, zeros show no dynamics while the Ones have high loss in the training.

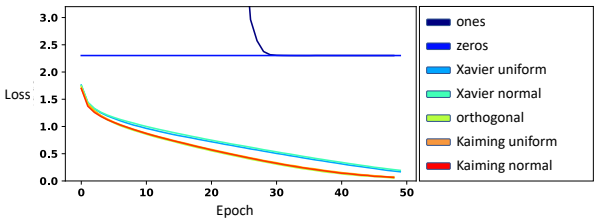

Figure 1: CIFAR10 training loss for several initialization methods.

In the next two subsections, we present the weight distributions and the feature distributions with input types of train, valid, and random. The overall analysis of targeted to the standard deviation and the shape of the distribution before and afeter training.

3.1 Weight Distribution

The orthogonal initialization had the largest variance over all layers for both weights and biases. In the case of uniform distribution, they are transformed to the Gaussian distribution. Therefore, while the normal initialization is remaining same Gaussian.

In general, the standard deviation are slightly increased after the training. We can conclude that the distribution of weights and biases are trained in the direction of normal distribution and the variance is not reduced.

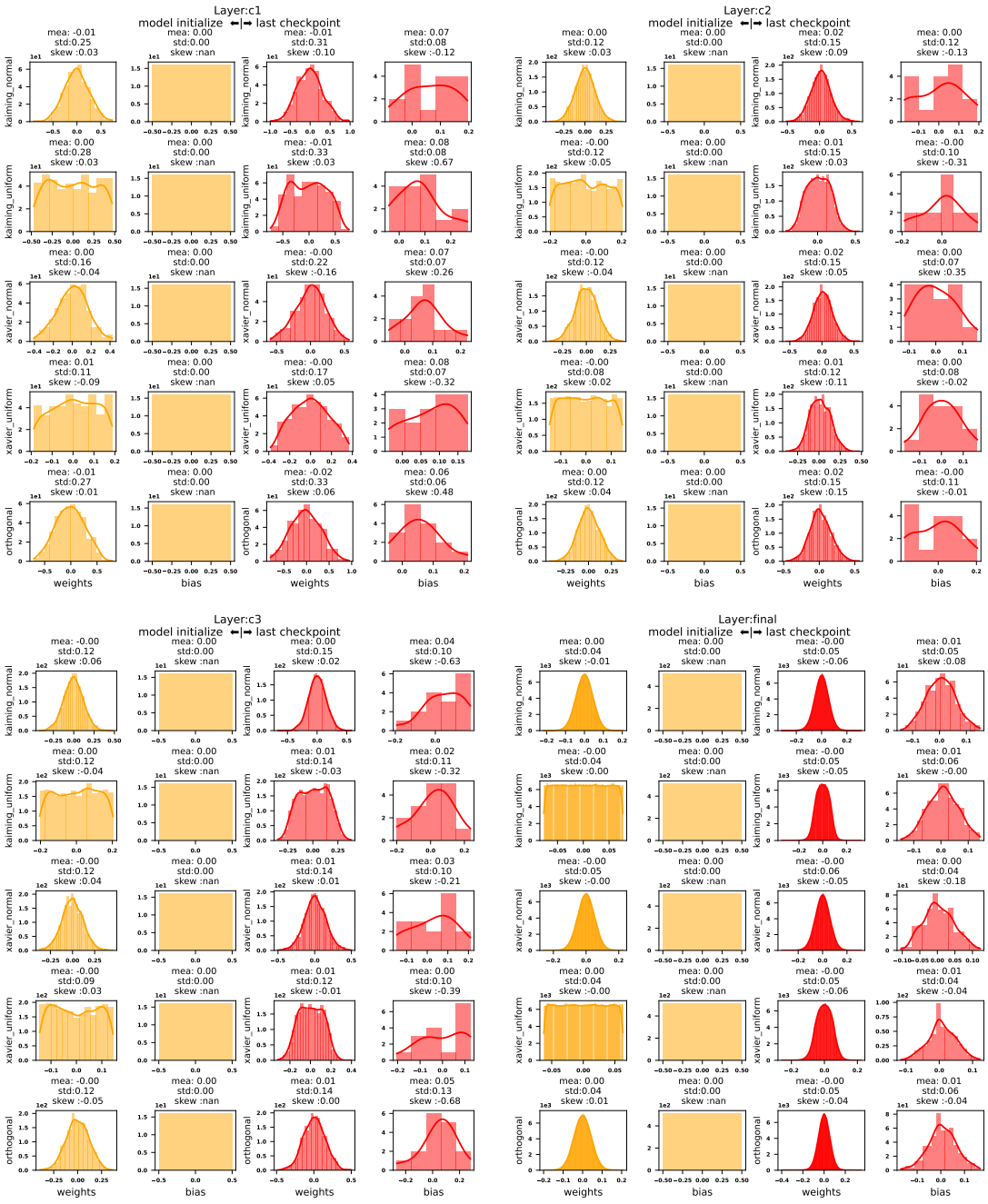

Figure 2: Weight distributions before and after training.

3.2 Feature Distribution

Note that the distribution of features are having zero mean in before training and having slightly off to the zero mean (in less than 1.0 magnitude). However, in the case of variance, it is significantly increased after the training. The standard deviation becomes larger as the layers are deeper. Remark that at the early stage, they are having mostly 1.0 standard deviation.

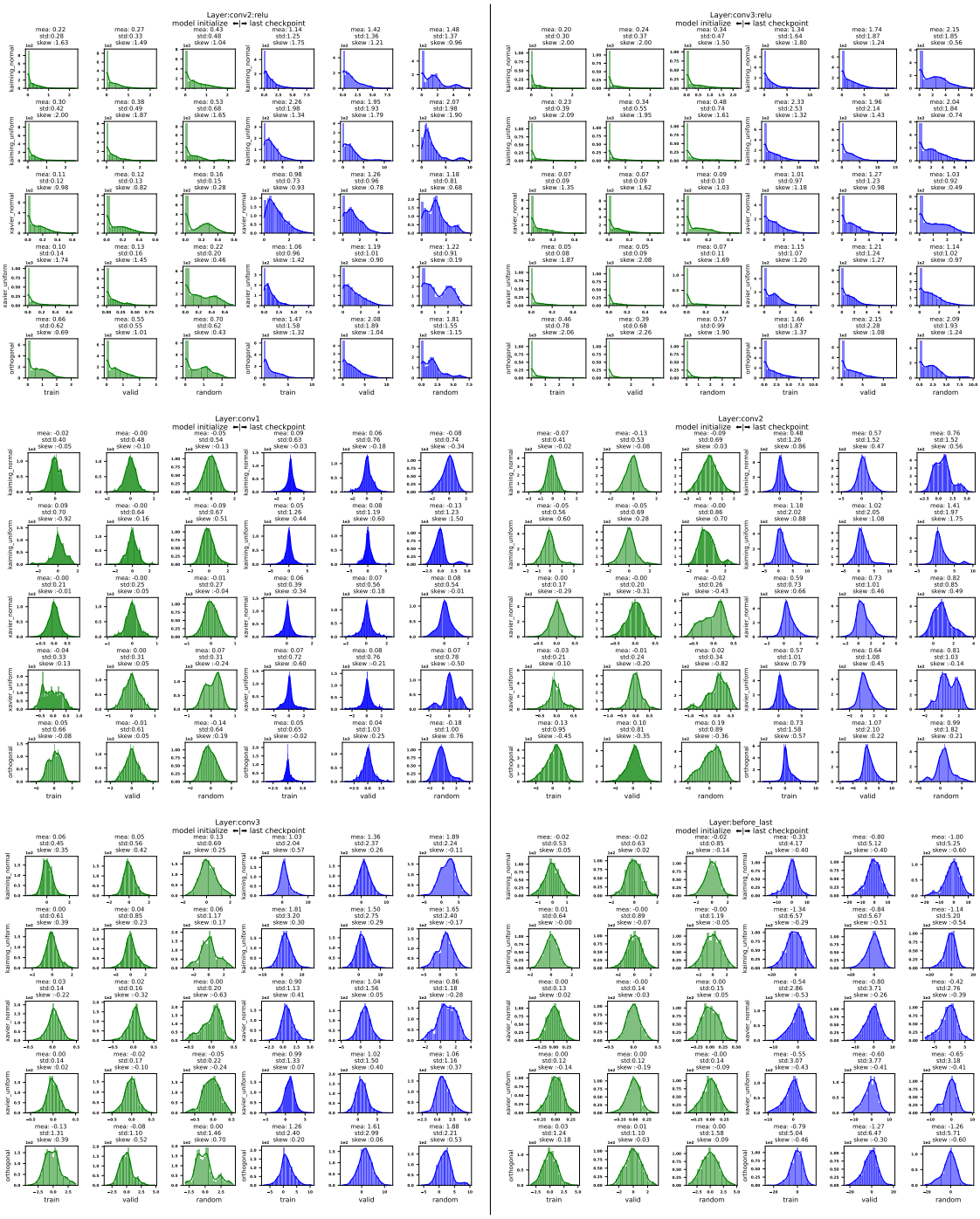

Figure 3: Feature distributions before and after training.

4 Experiment - Second

Figure 4 shows the distribution of *Absorption* and *Emission* of ResNet models computed with Imagenet21K validation images. We can see that each channel has different median and variance. The amount of outgoing gradients is similar with the amount of incoming gradients (*Absorption*), however, the outgoing gradients (*Emission*) has higher variance. The reason is that lower layer has higher variance on weights (see Figure 3) and the the outgoing gradients, which is the multiplication of incoming gradients and the weights, have higher variance.

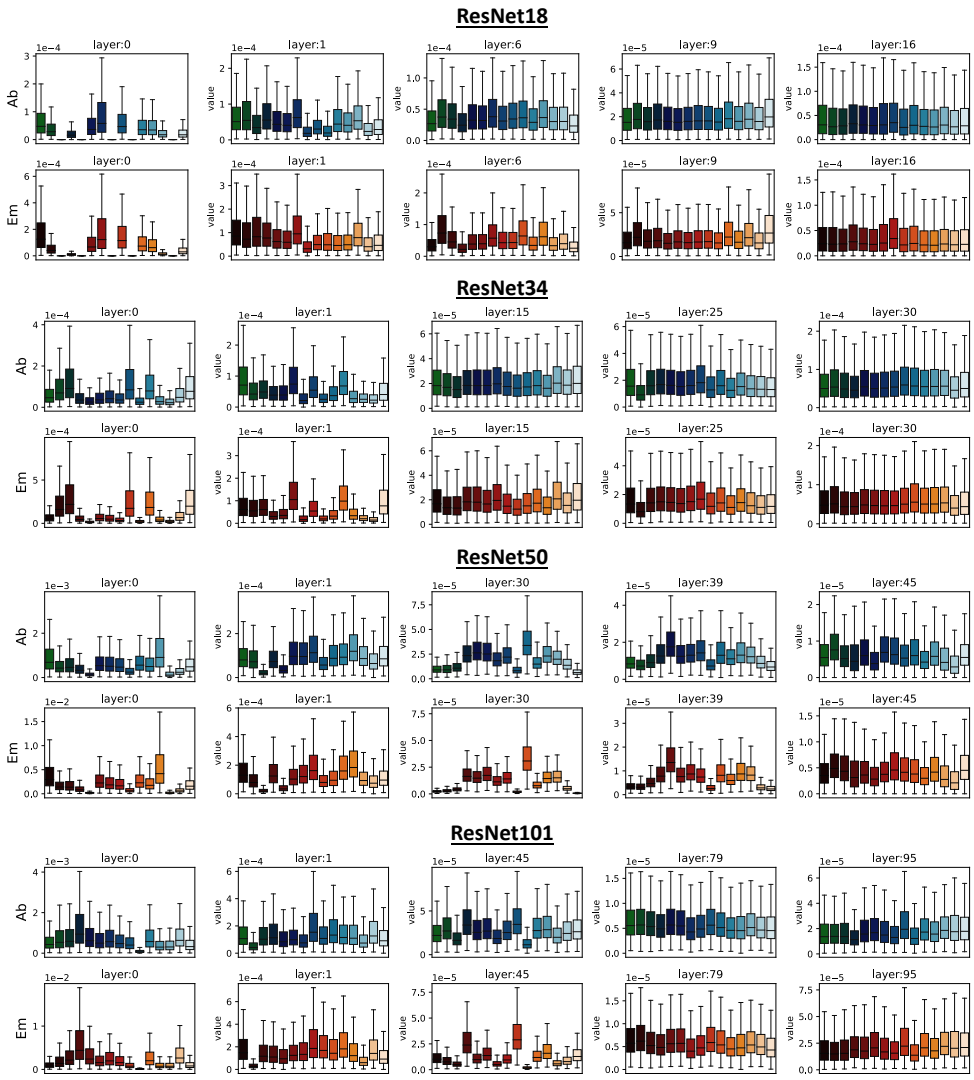

Figure 4: Distribution of *Absorption* and *Emission* of first 15 channels for 50K samples.

5 Conclusion

The training of deep neural network is highly affected by the weights and gradients and we broaden the understanding of gradients by showing distributions of them. We found that each channel of convolution layers have different amount of gradients. We acknowledge that the detailed analysis on the gradients are required for the thorough understanding of deep neural networks.