Lei Mao

Artificial Intelligence
Machine Learning
Computer Science

Santa Clara, California

POSTS	CATEGORIES	TAGS
431	8	262

CATALOGUE

Layer Normalization Explained

05-31-2019 10-15-2020 BLOG 5 MINUTES READ (ABOUT 698 WORDS) 4614 VISITS

Introduction

Recently I came across with layer normalization in the Transformer model for machine translation and I found that a special normalization layer called "layer normalization" was used throughout the model, so I decided to check how it works and compare it with the batch normalization we normally used in computer vision models.

Mathematical Definition

Given inputs x over a minibatch of size m, $B = \{x_1, x_2, \dots, x_m\}$, each sample x_i contains K elements, i.e. the length of flatten x_i is K, by applying transformation of your inputs using some learned parameters γ and β, the outputs could be expressed as $B' = \{y_1, y_2, \dots, y_m\}$, where $y_i = \text{LN}_{\gamma,\beta}(x_i)$.

More concretely, we first calculate the mean and the variance of of each sample from the minibatch. For sample x_i whose flatten format is $\{x_{i,1}, x_{i,2}, \dots, x_{i,K}\}$, we have its mean μ_i and variance σ_i^2.

$$\mu_i = \frac{1}{K}\sum_{k=1}^{K} x_{i,k}$$

$$\sigma_i^2 = \frac{1}{K}\sum_{k=1}^{K}(x_{i,k} - \mu_i)^2$$

Then we normalize each sample such that the elements in the sample have zero mean and unit variance. ϵ is for numerical stability in case the denominator becomes zero by chance.

$$\hat{x}_{i,k} = \frac{x_{i,k} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$$

Finally, there is a scaling and shifting step. γ and β are learnable parameters.

$$y_i = \gamma\hat{x}_i + \beta \equiv \text{LN}_{\gamma,\beta}(x_i)$$

We can see from the math above that layer normalization has nothing to do with other samples in the batch.

Layer Normalization for Convolutional Neural Network

If layer normalization is working on the outputs from a convolution layer, the math has to be modified slightly since it does not make sense to group all the elements from distinct channels together and compute the mean and variance. Each channel is considered as an "independent" sample and all the normalization was done for that specific channel only within the sample.

Assume the input tensor has shape $[m, H, W, C]$, for each channel $c \in \{1, 2, \cdots, C\}$

$$\mu_{i,c} = \frac{1}{HW}\sum_{j=1}^{H}\sum_{k=1}^{W} x_{i,j,k,c}$$

$$\sigma_{i,c}^2 = \frac{1}{HW}\sum_{j=1}^{H}\sum_{k=1}^{W}(x_{i,j,k,c} - \mu_{i,c})^2$$

$$\hat{x}_{i,j,k,c} = \frac{x_{i,j,k,c} - \mu_{i,c}}{\sqrt{\sigma_{i,c}^2 + \epsilon}}$$

Specifically for each channel, we have learnable parameters γ_c and β_c, such that

$$y_{i,:,:,c} = \gamma_c\hat{x}_{i,:,:,c} + \beta_c \equiv \text{LN}_{\gamma_c,\beta_c}(x_{i,:,:,c})$$

Layer Normalization vs Batch Normalization

I had a simple blog post on batch normalization previously. Like that simple blog post, I am not going to talk about the advantage of layer normalization over batch normalization or how to choose normalization techniques in this blog post.

Layer Normalization vs Instance Normalization?

Instance normalization, however, only exists for 3D or higher dimensional tensor inputs, since it requires the tensor to have batch and each sample in the batch needs to have layers (channels). If the samples in batch only have 1 channel (a dummy channel), instance normalization on the batch is exactly the same as layer normalization on the batch with this single dummy channel removed. Batch normalization and layer normalization works for 2D tensors which only consists of batch dimension without layers. Surprisingly (or not?), instance normalization for 3D or 4D tensor is exactly the same as layer normalization for convolution outputs as I mentioned above, because each sample in the batch is an instance, we are layer normalizing samples which happen to have multiple channels, and we ignore batches during normalization. So we could say about instance normalization in this way, instance normalization is a natural extension of layer normalization to convolutions, or it is just a new name for an old concept.

References

- Layer Normalization
- Layer Normalization TensorFlow Implementation

Layer Normalization Explained
https://leimao.github.io/blog/Layer-Normalization/

Author	Posted on	Updated on	Licensed under
Lei Mao	05-31-2019	10-15-2020	

DEEP LEARNING

Transformer Explained in One Single Page

Build CMake with SSL Support

Comments

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Lixing Yu • 2 years ago

Hi, Great post. I just got confused. When I read this blog: https://becominghuman.ai/al...
I found that your mathematic expression is more like the Instance Normalization. I'm not sure where did I made the mistake. Based on my understanding, I think the "layer" normalization should go across the channels within one simple, am I right? Thanks for your reply in advance.

1 ▲ | ▼ • Reply • Share ›

noob • 2 years ago

I am confused about two definitions about layernormalization. In the first, x is a flattened tensor, so in my opinion, within NHWC data, it can be see as [N,H*W*C] as [m,K]. So layer normalization can be calculated. But in the second, we calculated each normalization of each channel. Which is correct?

1 ▲ | ▼ • Reply • Share ›

Lei Mao Mod ➚ noob • 2 years ago • edited

Layer normalization not only works for convolutional layers but also works for normal fully connected layers. In addition, it is related to how you define a "layer" in your tensor. If you define the entire tensor as a layer (ignore batch for now), usually something in the fully connected layers, then all the elements in the tensor could be flattened and its layer norm could be computed using the first equations. In convolutional layers, usually each channel is defined as a layer, therefore, layer normalization was applied channel-wise. The math was not changed for "different versions of layer normalization", what's changed is how you define a layer.

The blog post is a little bit old. I think I will make this more explicit in the future once I get a chance.

▲ | ▼ 1 • Reply • Share ›

noob ➚ Lei Mao • 2 years ago

Thanks for your reply, Got it! And do the scale and bias have the shape [1,1,C] (ignores batch) in convolutional layers? If so, it seems layer normalization is the same as batch normalization with batch_size = 1.

▲ | ▼ • Reply • Share ›

Lei Mao Mod ➚ noob • 2 years ago

Yes, when batch_size = 1.

▲ | ▼ • Reply • Share ›

noob ➚ Lei Mao • 2 years ago • edited

Hi, it's me again. I have some problems when I am learning to implement layer normalization. As you said before, we can choose which layer to do normalization. But I find it different in Pytorch. For example, batch datad tensor

with shape [2,2,2,4] (NHWC), I can't choose to do laver

ALSO ON **LEIMAO.GITHUB.IO**

3 months ago • 1 comment

5 months ago • 1 comment