

Open in app ↗

Medium

 Search Write

★ Get unlimited access to the best of Medium for less than \$1/week. [Become a member](#)



Difference between Local Response Normalization and Batch Normalization

A short tutorial on different normalization techniques used in Deep Neural Networks.



Aqeel Anwar · Follow

Published in Towards Data Science · 7 min read · Jun 19, 2019



539



10



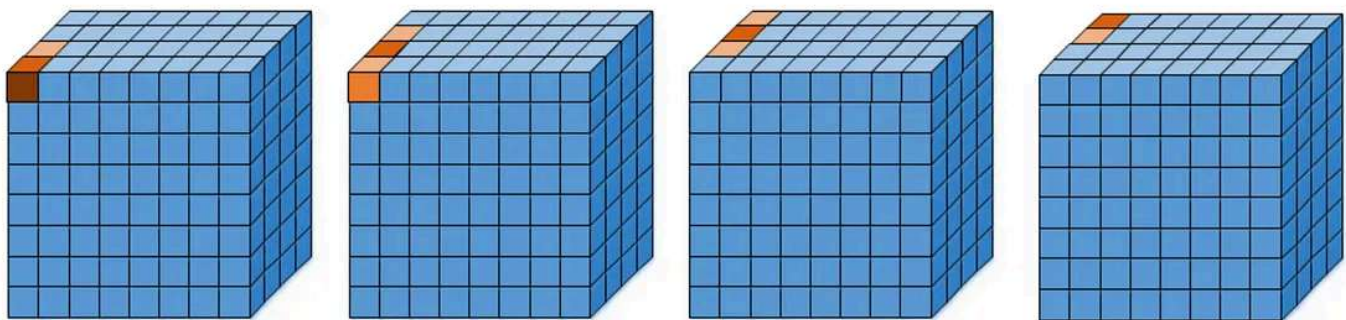
Why Normalization?

Normalization has become important for deep neural networks that compensate for the unbounded nature of certain activation functions such as ReLU, ELU, etc. With these activation functions, the output layers are not constrained within a bounded range (such as $[-1, 1]$ for \tanh), rather they can grow as high as the training allows it. To limit the unbounded activation from increasing the output layer values, normalization is used just before the activation function. There are two common normalization techniques used in deep neural networks and are often misunderstood by beginners. In this tutorial, a detailed explanation of both the normalization techniques will be discussed highlighting their key differences.

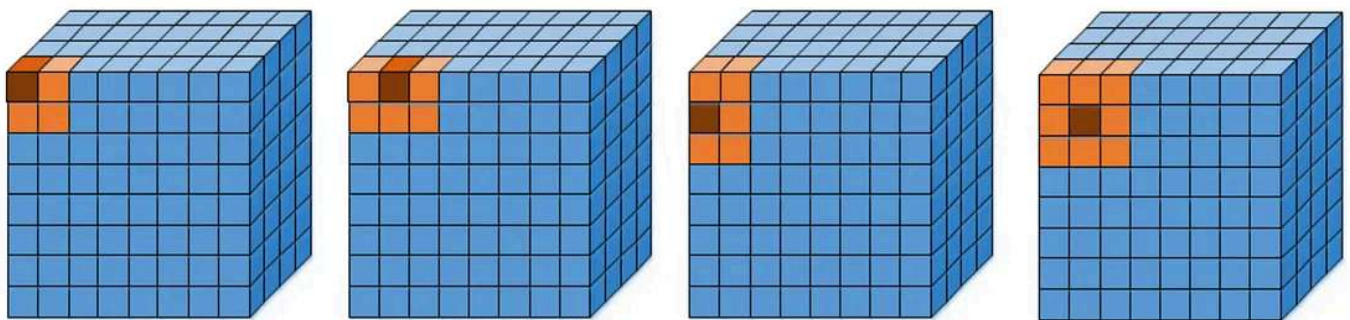
Local Response Normalization

Local Response Normalization (LRN) was first introduced in AlexNet architecture where the activation function used was *ReLU* as opposed to the more common *tanh* and *sigmoid* at that time. Apart from the reason mentioned above, the reason for using LRN was to encourage *lateral inhibition*. It is a concept in Neurobiology that refers to the capacity of a neuron to reduce the activity of its neighbors [1]. In DNNs, the purpose of this lateral inhibition is to carry out local contrast enhancement so that locally maximum pixel values are used as excitation for the next layers.

LRN is a **non-trainable layer** that square-normalizes the pixel values in a feature map within a local neighborhood. There are two types of LRN based on the neighborhood defined and can be seen in the figure below.



a) Inter-Channel LRN (n=2)



b) Intra-Channel LRN (n=2)

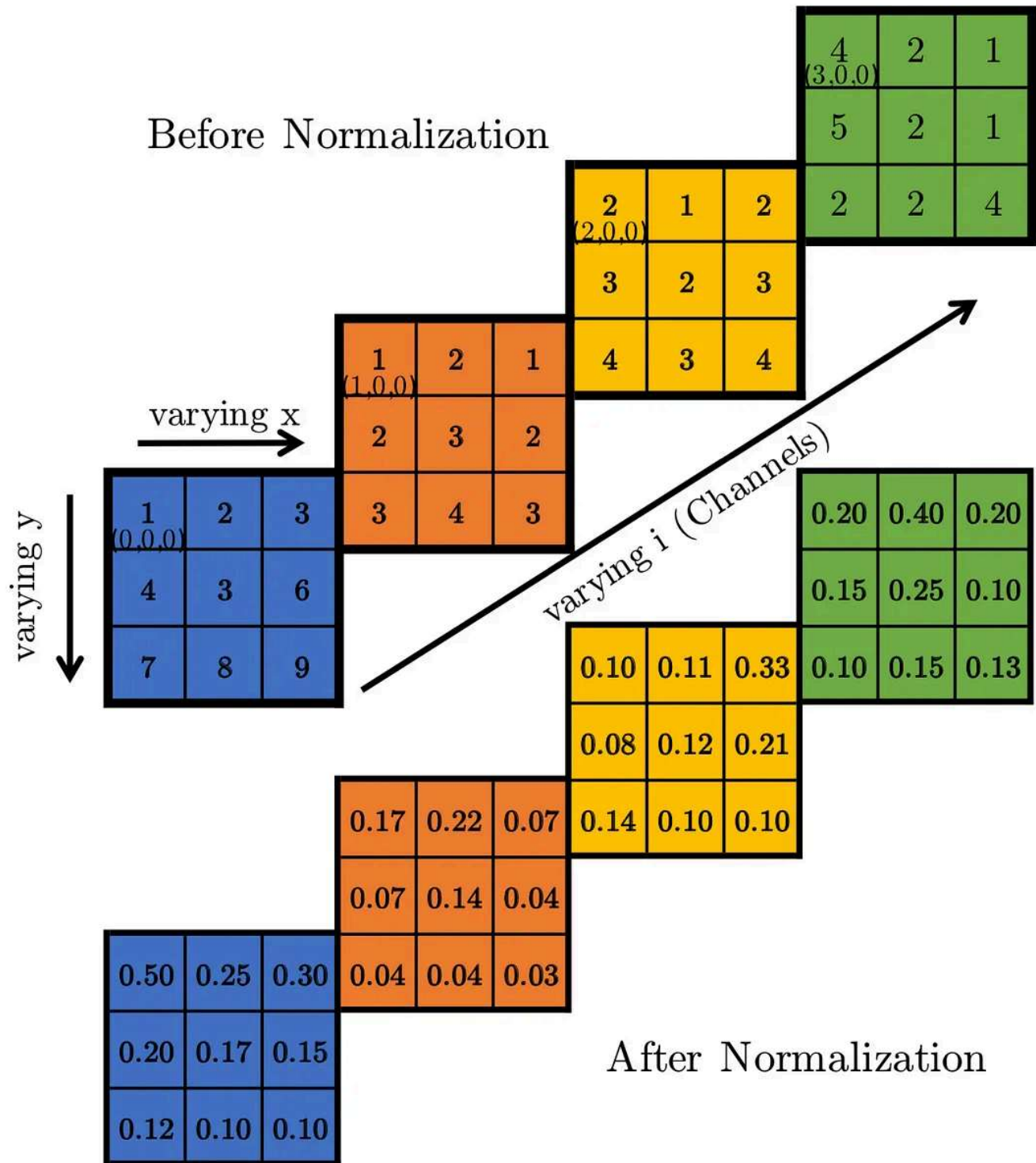
Inter-Channel LRN: This is originally what the AlexNet paper used. The neighborhood defined is **across the channel**. For each (x,y) position, the normalization is carried out in the depth dimension and is given by the following formula

$$b_{x,y}^i = a_{x,y}^i / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^j)^2 \right)^{\beta}$$

LRN used in AlexNet [2]

where i indicates the output of filter i , $a(x,y)$, $b(x,y)$ the pixel values at (x,y) position before and after normalization respectively, and N is the total number of channels. The constants (k, α, β, n) are hyper-parameters. k is used to avoid any singularities (division by zero), α is used as a normalization constant, while β is a contrasting constant. The constant n is used to define the neighborhood length i.e. how many consecutive pixel values need to be considered while carrying out the normalization. The case of $(k, \alpha, \beta, n) = (0, 1, 1, N)$ is the standard normalization). In the figure above n is taken to be 2 while $N=4$.

Let's have a look at an example of Inter-channel LRN. Consider the following figure



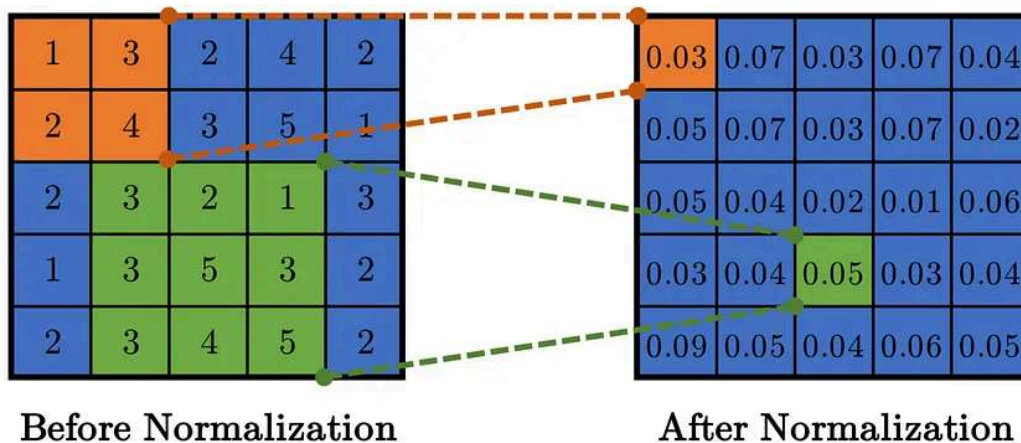
Different colors denote different channels and hence $N=4$. Let's take the hyper-parameters to be $(k, \alpha, \beta, n)=(0,1,1,2)$. The value of $n=2$ means that while calculating the normalized value at position (i,x,y) , we consider the values at the same position for the previous and next filter i.e $(i-1, x, y)$ and $(i+1, x, y)$. For $(i,x,y)=(0,0,0)$ we have $value(i,x,y)=1$, $value(i-1,x,y)$ doesn't exist

and $value(i+,x,y)=1$. Hence $normalized_value(i,x,y) = 1/(1^2+1^2) = 0.5$ and can be seen in the lower part of the figure above. The rest of the normalized values are calculated in a similar way.

Intra-Channel LRN: In Intra-channel LRN, the neighborhood is extended within the same channel only as can be seen in the figure above. The formula is given by

$$b_{x,y}^k = a_{x,y}^k / \left(k + \alpha \sum_{i=\max(0,x-n/2)}^{\min(W,x+n/2)} \sum_{j=\max(0,y-n/2)}^{\min(H,y+n/2)} (a_{i,j}^k)^2 \right)^{\beta}$$

where (W,H) are the width and height of the feature map (for example in the figure above (W,H) = (8,8)). The only difference between Inter and Intra Channel LRN is the neighborhood for normalization. In Intra-channel LRN, a 2D neighborhood is defined (as opposed to the 1D neighborhood in Inter-Channel) around the pixel under-consideration. As an example, the figure below shows the Intra-Channel normalization on a 5x5 feature map with n=2 (i.e. 2D neighborhood of size (n+1)x(n+1) centered at (x,y)).



Batch Normalization:

Batch Normalization (BN) is a **trainable layer** normally used for addressing the issues of *Internal Covariate Shift (ICF)* [1]. ICF arises due to the changing distribution of the hidden neurons/activation. Consider the following example of binary classification where we need to classify roses and no-roses



Roses vs No-roses classification. The feature map plotted on the right have different distributions for two different batch sampled from the dataset [1]

Say we have trained a neural network, and now we select two significantly different looking batches from the dataset for inference (as shown above). If we do a forward pass with these two batches and plot the feature space of a hidden layer (deep in the network) we will see a significant shift in the distribution as seen on the right-hand side of the figure above. This is called the *Covariate shift* of the input neurons. What impact does this have during training? During training, if we select batches that belong to different distributions then it slows down the training since for a given batch it tries to

learn a certain distribution, which is different for the next batch. Hence it keeps on bouncing back and forth between distributions until it converges. This *Covariate Shift* can be mitigated by making sure that the members within a batch do not belong to the same/similar distribution. This can be done by randomly selecting images for batches. Similar Covariate Shift exists for hidden neurons. Even if the batches are randomly selected, the hidden neuron can end up having a certain distribution which slows down the training. This Covariate shift for hidden layers is called Internal Covariate Shift. The problem is that we can't directly control the distribution of the hidden neurons, as we did for input neurons, because it keeps on changing as training updates the training parameters. Batch Normalization helps mitigate this issue.

In batch normalization, the output of hidden neurons is processed in the following manner before being fed to the activation function.

1. Normalize the entire batch B to be zero mean and unit variance
 - Calculate the mean of the entire mini-batch output: u_B
 - Calculate the variance of the entire mini-batch output: σ_B
 - Normalize the mini-batch by subtracting the mean and dividing with variance
2. Introduce two trainable parameters (*Gamma*: scale_variable and *Beta*: shift_variable) to scale and shift the normalized mini-batch output
3. Feed this scaled and shifted normalized mini-batch to the activation function.

The BN algorithm can be seen in the figure below.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1, \dots, x_m\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

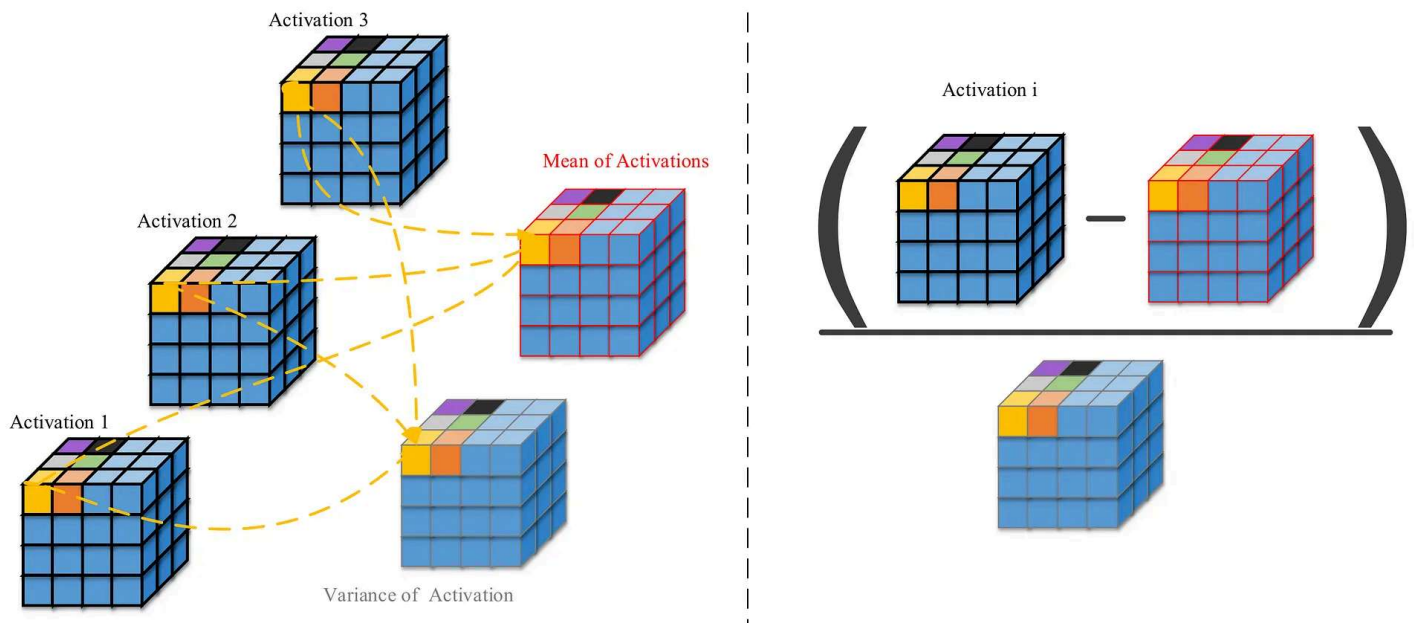
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch Normalization Algorithm [2]

The normalization is carried out for each pixel across all the activations in a batch. Consider the figure below. Let us assume we have a mini-batch of size 3. A hidden layer produces an activation of size $(C, H, W) = (4, 4, 4)$. Since the batch size is 3, we will have 3 of such activations. Now for each pixel in the activation (i.e. for each $4 \times 4 \times 4 = 64$ pixel), we will normalize it by finding the mean and variance of this pixel position in all the activations as shown in the left part of the figure below. Once the mean and variance are found, we will subtract the mean from each of the activations and divide it with the variance. The right part of the figure below depicts this. The subtraction and division are carried out point-wise. (if you are used to MATLAB, the division is dot-division `./`).



The reason for step 2 i.e. scaling and shifting is to let the training decide whether we even need the normalization or not. There are some cases when not having normalization may yield better results. So instead of selecting beforehand whether to include a normalization layer or not, BN lets the training decide it. When $\Gamma = \sigma_B$ and $\beta = \mu_B$, no normalization is carried out, and original activations are restored. A really good video tutorial on BN by Andrew Ng can be found [here](#)

Comparison:

LRN has multiple directions to perform normalization across (Inter or Intra Channel), on the other hand, BN has only one way of being carried out (for each pixel position across all the activations). The table below compares the two normalization techniques.

Comparison of the two normalization techniques in DNNs				
Norm Type	Trainable	Training Parameters	Fouses on	Regularization
LRN	No	0	Lateral Inhibition	No
BN	Yes	2 (Scale and Shift)	Internal Covariate Shift	Yes

References:

[1] <https://www.learnopencv.com/batch-normalization-in-deep-networks/>

[2] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

Bonus:

Compact cheat sheets for this topic and many other important topics in Machine Learning can be found in the link below

Cheat Sheets for Machine Learning Interview Topics

A visual cheatsheet for ML interviews (www.cheatsheets.aqeel-anwar.com)

medium.com

If this article was helpful to you, feel free to clap, share and respond to it. If want to learn more about Machine Learning and Data Science, follow me [@Aqeel Anwar](#) or connect with me on [LinkedIn](#).

Machine Learning

Artificial Intelligence

Deep Neural Networks

Deep Learning



Published in Towards Data Science

Following

801K Followers · Last published 3 days ago

Your home for data science and AI. The world's leading publication for data science, data analytics, data engineering, machine learning, and artificial intelligence professionals.



Written by Aqeel Anwar

Follow

3.7K Followers · 42 Following

Senior ML Engineer @NVIDIA | ex-Samsung | GeorgiaTech | Writer | Researcher | Traveler | www.aqeel-anwar.com | https://twitter.com/_aqeelanwar

Responses (10)



What are your thoughts?

Respond



Eduard Čuba
Sep 24, 2020



$$1/({}^{12}+{}^{12}) = 0.5$$

Thanks for the article!

A tiny error in the superscript here that confused me for a moment (should be $1/(1^2+1^2) = 0.5$)


 1 [Reply](#)



Nikita Gupta
May 31, 2020

...

Thank you for the article!


 1 [Reply](#)



Moe Shams
Feb 9, 2020

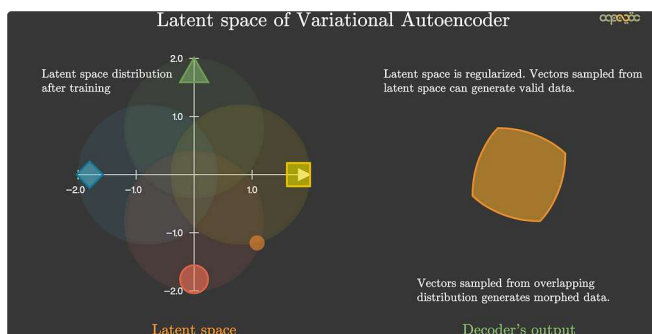
...

Thanks alot for clear description.

 1 [Reply](#)

See all responses

More from Aqeel Anwar and Towards Data Science



 In Towards Data Science by Aqeel Anwar

```
T2=(Teller/2)
Do
  X=T1*T2
  If X=Teller Then
    Antallmuligheter=Antallmuligheter+1
  Endif
  Exit If Antallmuligheter>1
  Exit If T2=1
  T2=T2-1
Loop
Exit If Antallmuligheter>2
Exit If T1=Teller
T1=T1+1
Loop
```

 In Towards Data Science by Ragnvald Larsen

Difference between AutoEncoder (AE) and Variational AutoEncoder...

How can you compress data or even generate data from random values? That is what...

Nov 3, 2021 🖱️ 566 💬 8



tds In Towards Data Science by Muhammad Ardi

Show and Tell

Implementing one of the earliest neural image caption generator models with PyTorch.

🌟 4d ago 🖱️ 26



Are Public Agencies Letting Open-Source Software Down?

Open-source software is everywhere, yet public agencies and institutions often fall...

4d ago 🖱️ 13 💬 1



Aqeel Anwar

Machine Learning Resume that got me shortlisted for Meta, Microsof...

A list of things to focus on your resume and an example CV.

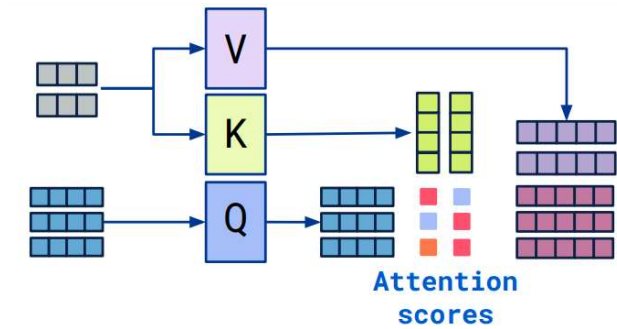
Jan 15, 2022 🖱️ 1.3K 💬 21




See all from Aqeel Anwar

See all from Towards Data Science

Recommended from Medium



 Mohd Faraaz

Implementing Self-Attention from Scratch in PyTorch

In this article we will see the step by step tutorial of the self attention mechanism whic...

Aug 31, 2024



 In Level Up Coding by Ekin Ceylan

CSS @layer — An Almighty Solution or A New Curse?

Are we sure this new feature exactly matches any problems in CSS?



Jan 22



59



1



Lists



Predictive Modeling w/ Python

20 stories · 1816 saves



Natural Language Processing

1924 stories · 1577 saves



AI Regulation

6 stories · 682 saves



Practical Guides to Machine Learning

10 stories · 2187 saves



 In Biased-Algorithms by Amit Yadav

Batch Normalization vs Layer Normalization


“You might have heard the saying, ‘Normalize to optimize.’ In the world of deep learning,...

Sep 19, 2024

 2





 In Towards Data Science by Vyacheslav Efimov

Understanding Deep Learning Optimizers: Momentum, AdaGrad...

Gain intuition behind acceleration training techniques in neural networks

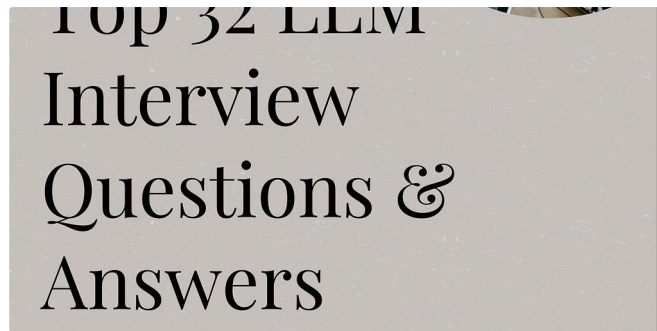
 Dec 30, 2023

 486

 4







 Narender Beniwal

LLM Interview Questions(Large Language Models): Top Interview...

Large Language Models (LLMs) are at the forefront of natural language processing...


 Oct 31, 2024

 42





7	1.239	2.167	2.833	4.255	6.346	9.04	12.02	14.07	18.48
8	1.647	2.733	3.490	5.071	7.344	10.22	13.36	15.51	20.09
9	2.088	3.325	4.168	5.899	8.343	11.39	14.68	16.92	21.67
10	2.558	3.940	4.865	6.737	9.342	12.55	15.99	18.31	23.21
11	3.053	4.575	5.578	7.584	10.341	13.70	17.28	19.68	24.72
12	3.571	5.226	6.304	8.438	11.340	14.85	18.55	21.03	26.22
13	4.107	5.892	7.042	9.299	12.340	15.98	19.81	22.36	27.69
14	4.660	6.571	7.790	10.165	13.339	17.12	21.06	23.68	29.14
15	5.229	7.261	8.547	11.037	14.339	18.25	22.31	25.00	30.58
16	5.812	7.962	9.312	11.912	15.338	19.37	23.54	26.30	32.00
17	6.408	8.672	10.085	12.792	16.338	20.49	24.77	27.59	33.41
18	7.015	9.390	10.865	13.675	17.338	21.60	25.99	28.87	34.80
19	7.633	10.117	11.651	14.562	18.338	22.72	27.20	30.14	36.19
20	8.260	10.851	12.443	15.452	19.337	23.83	28.41	31.41	37.57
22	9.542	12.338	14.041	17.240	21.337	26.04	30.81	33.92	40.29


 Rohollah

Mastering Feature Selection: Key Applications and Differences—...

Applications of Chi-Square in Feature Selection.How to use Chie-square for featur...

 Sep 6, 2024

 77





See more recommendations