

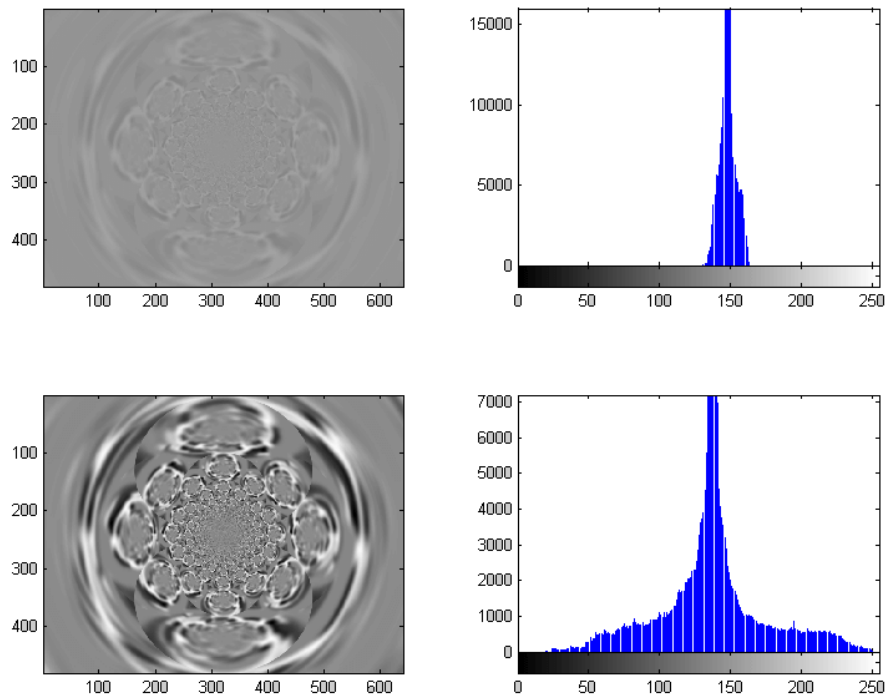
S6

Due No Due Date **Points** None **Available** after Feb 19 at 6:30am

Batch Normalization & Regularization

Normalizing input (LeCun et al 1998 “Efficient Backprop”)

Let's understand through some examples:



Why the redistribution of data is important for us?

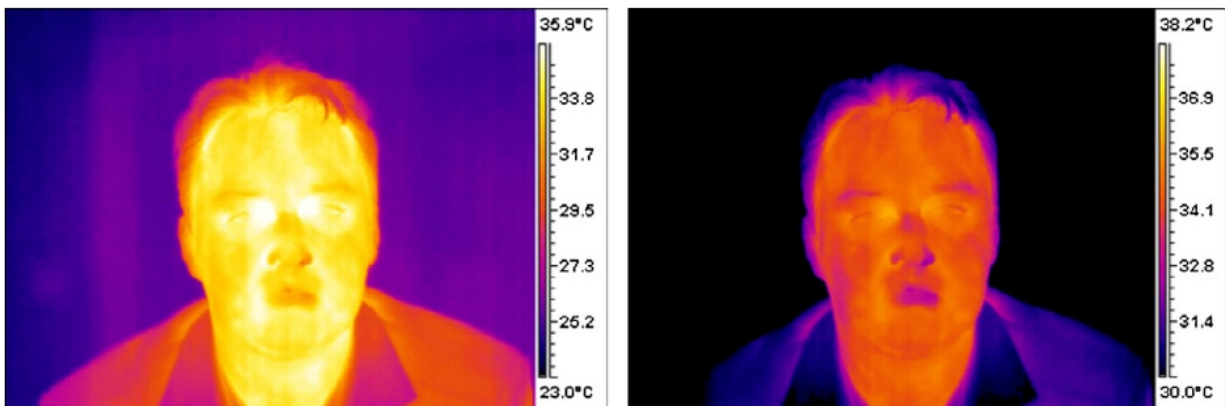


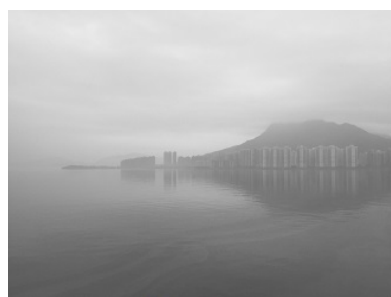
Fig. 10. Examples of thermal face images. Raw thermal image (left). Normalized thermal image (right).

Normalization is not Equalization

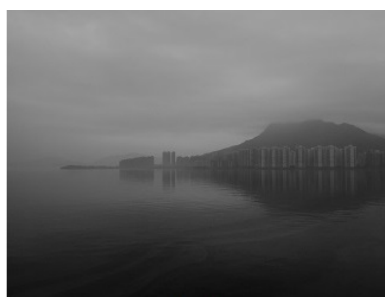
The normalize is quite simple, it looks for the maximum intensity pixel (we will use a grayscale example here) and a minimum intensity and then will determine a factor that scales the min intensity to black and the max intensity to white. This is applied to every pixel in the image which produces the final result.

The equalize will attempt to produce a histogram with equal amounts of pixels in each intensity level. This can produce unrealistic images since the intensities can be radically distorted but can also produce images very similar to normalization which preserves relative levels in which the equalization process does not.

So if you are concerned about keeping an image realistic then use normalization, but if you want a more even distribution of intensity levels then equalize can help with that. [SOURCE](http://www.roborealm.com/forum/index.php?thread_id=4350) [.\(http://www.roborealm.com/forum/index.php?thread_id=4350\)](http://www.roborealm.com/forum/index.php?thread_id=4350)



original image

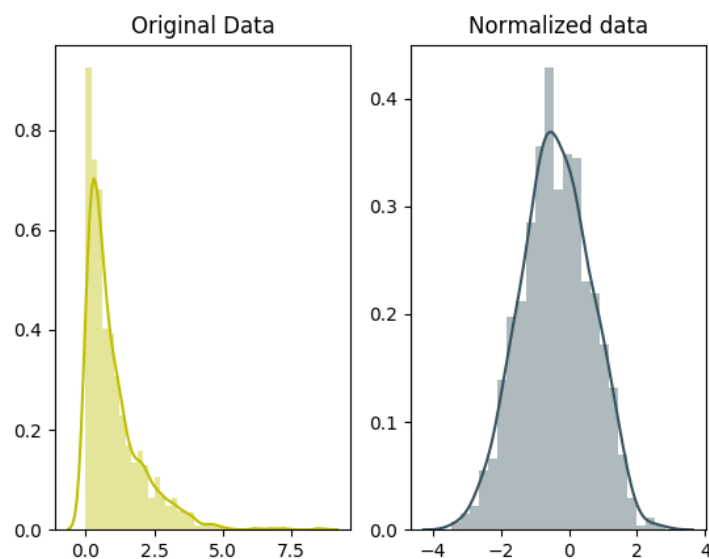
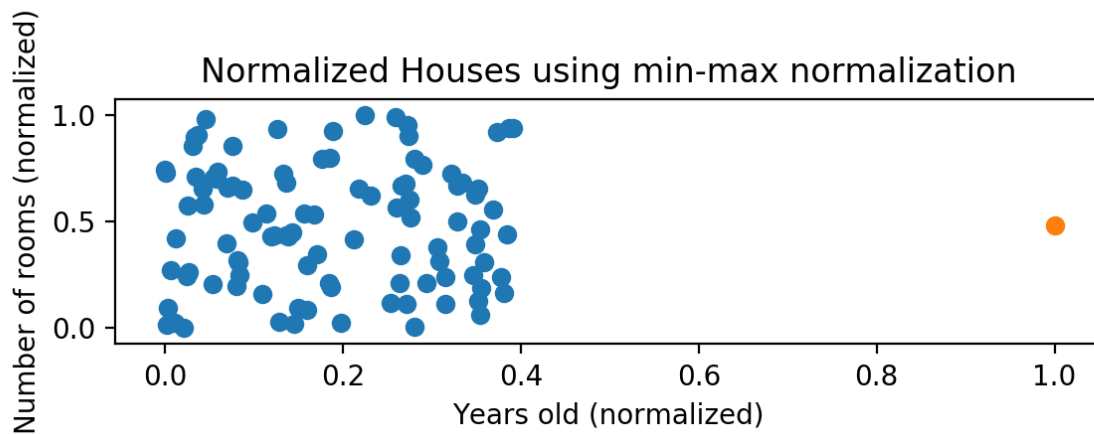
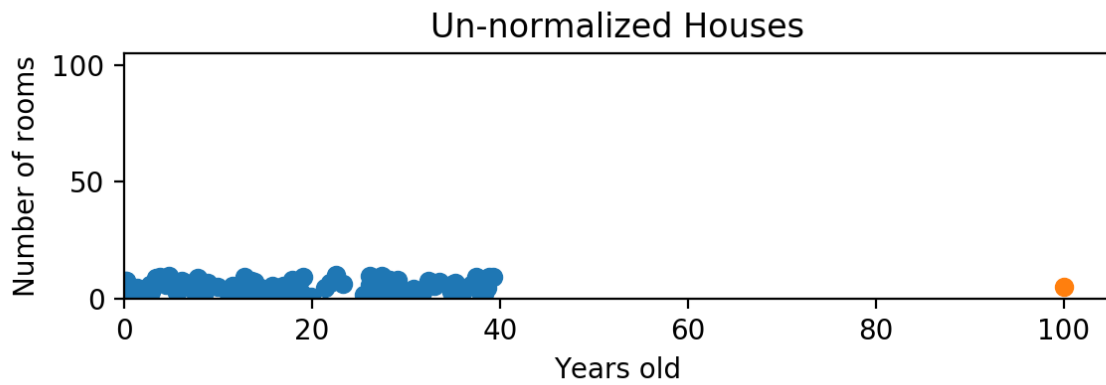


normalized image

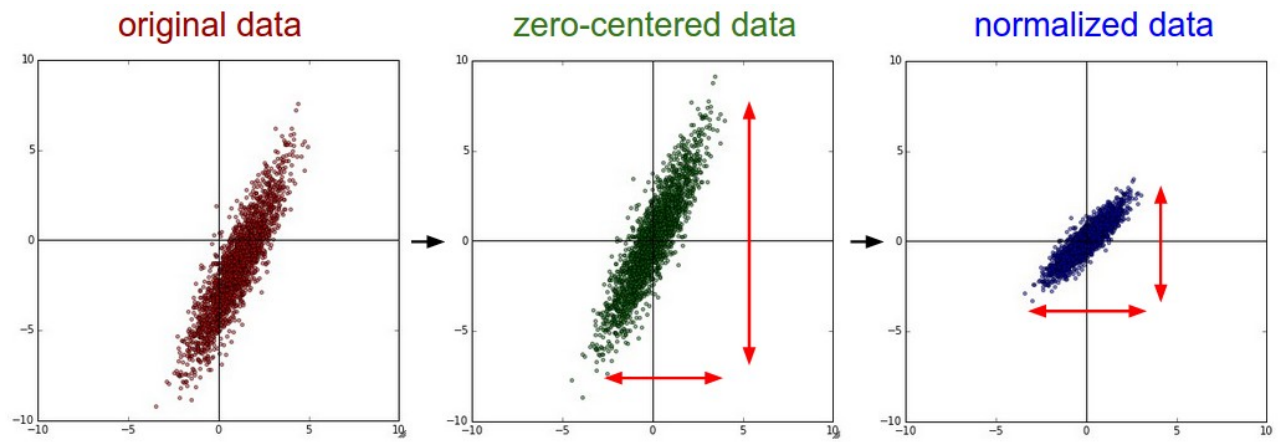


equalized image

Let's look at normalization working on other kinds of data.



How do we achieve normalization?



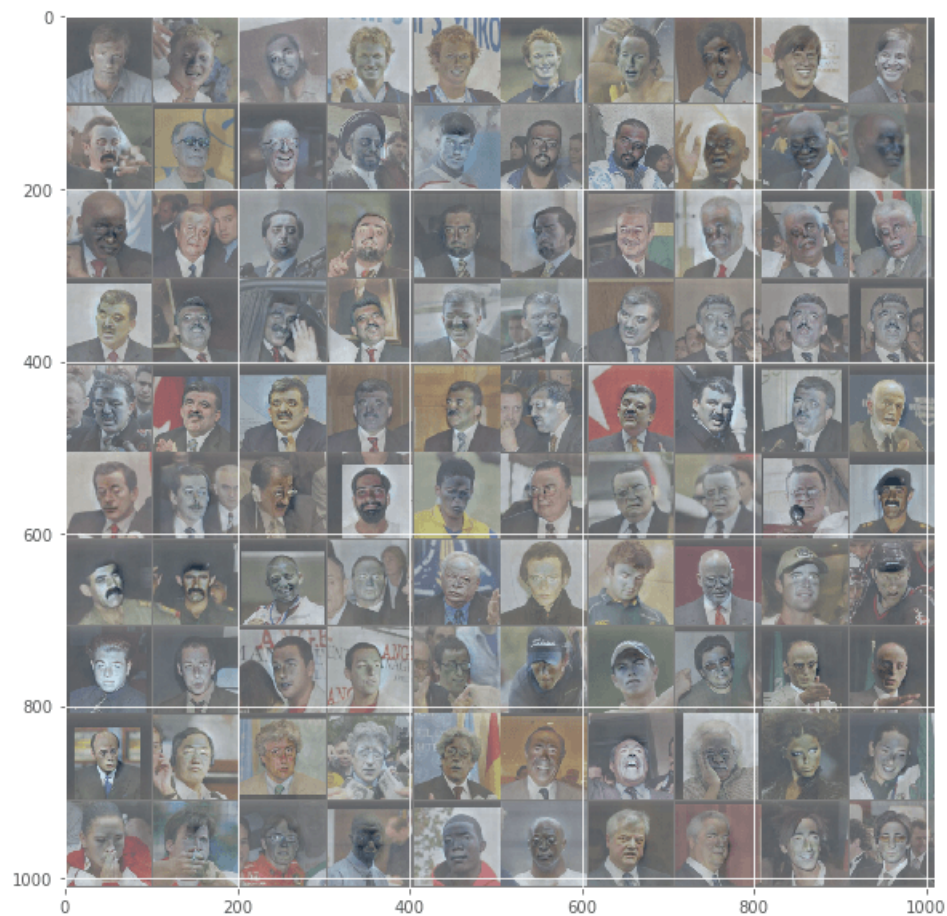
DO NOT PERFORM THIS AT HOME/WORK



Taking the mean (left) and standard deviation (right) of the batch, we get the following:

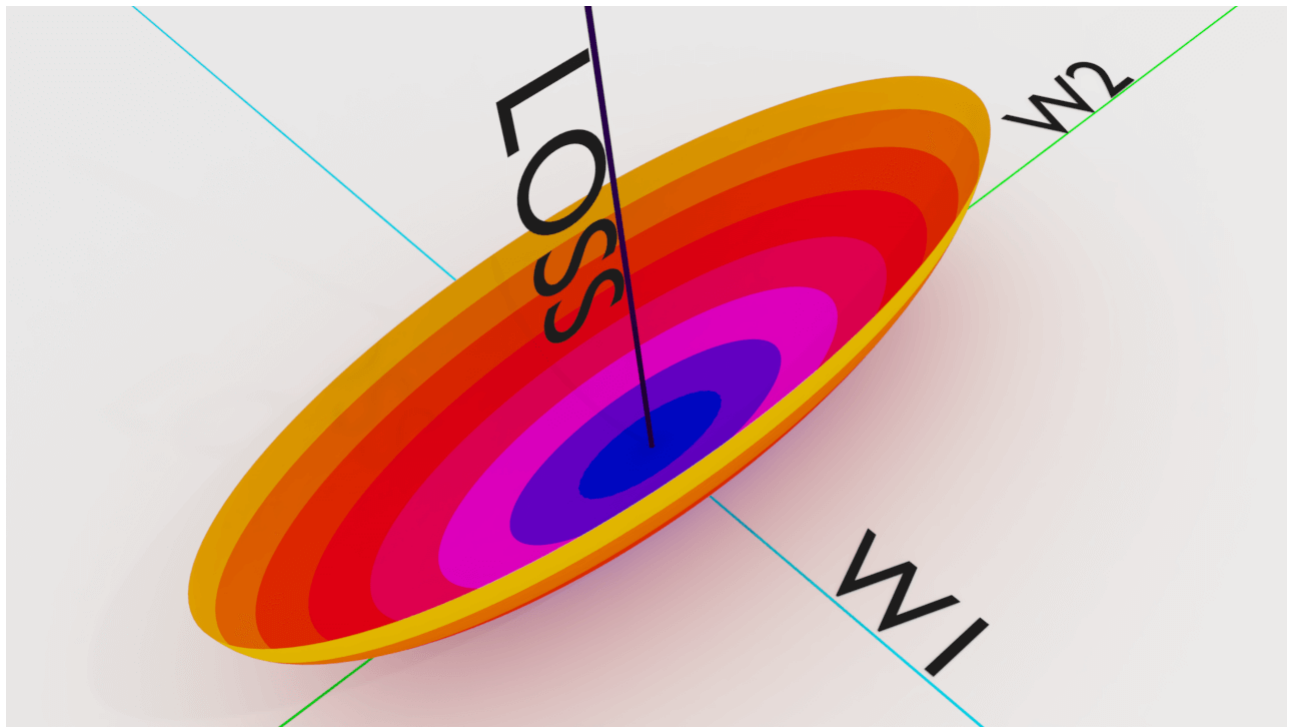


Our centered data resembles a face, but more importantly, our standard deviation image shows great variance along the borders and everywhere except the face. From our z-score formula, we can predict that the standardized values near the border and irrelevant areas will be relatively squashed more than values around the face. The resultant normalization appears as such:

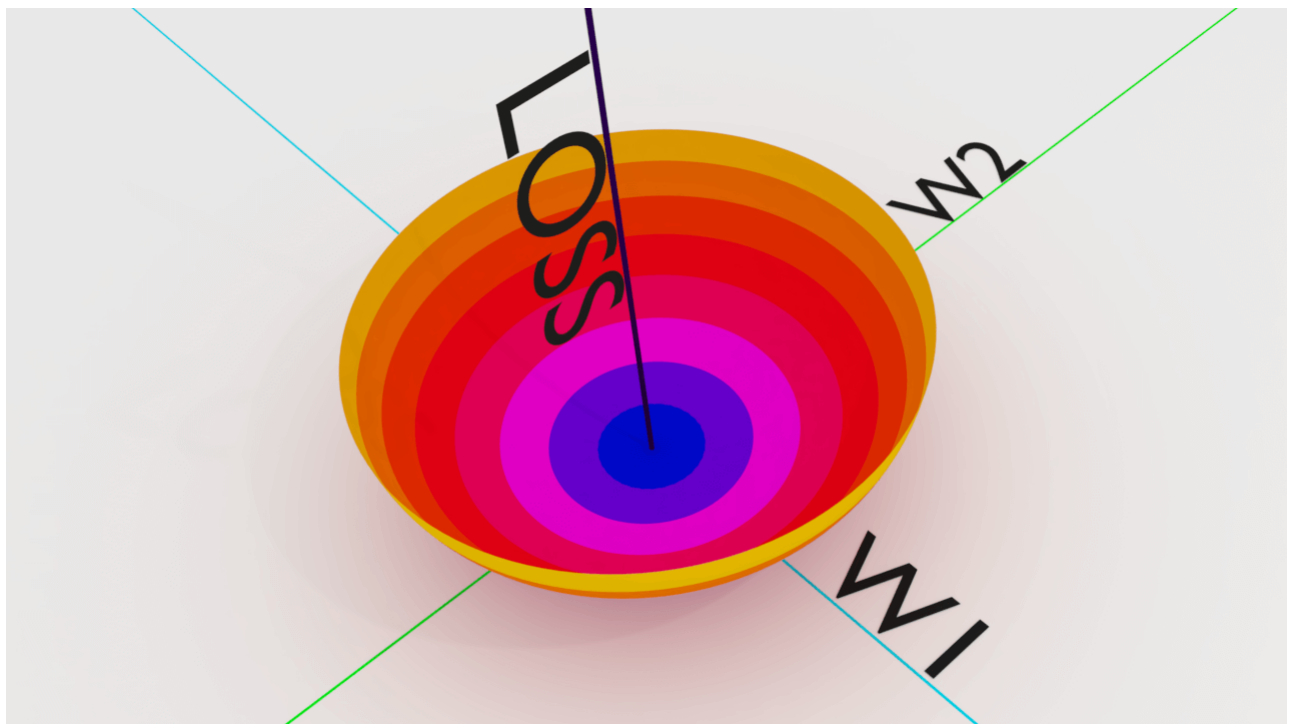


[Excellent Source \(https://datascience.stackexchange.com/questions/26881/data-preprocessing-should-we-normalise-images-pixel-wise\)](https://datascience.stackexchange.com/questions/26881/data-preprocessing-should-we-normalise-images-pixel-wise)

Let's think about our un-normalized kernels and how loss function would look like:

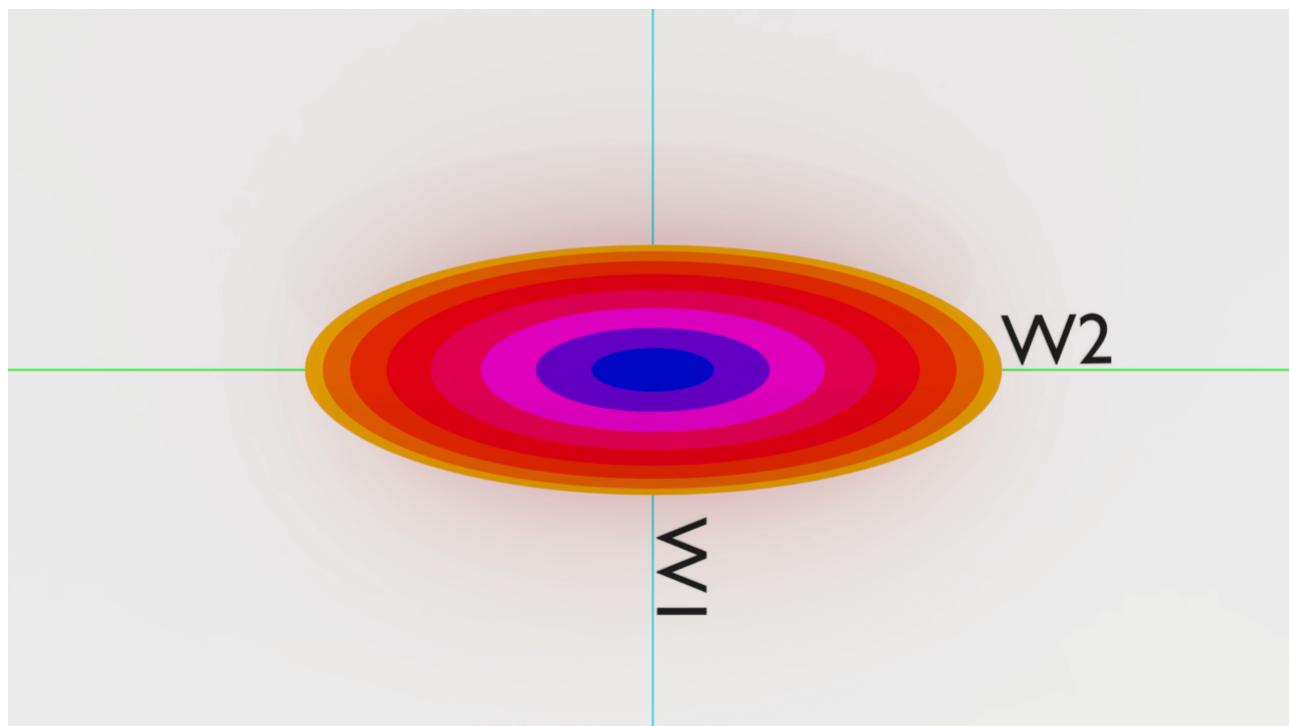


If we had normalized our kernels (indirectly channels), this is how it would look like:

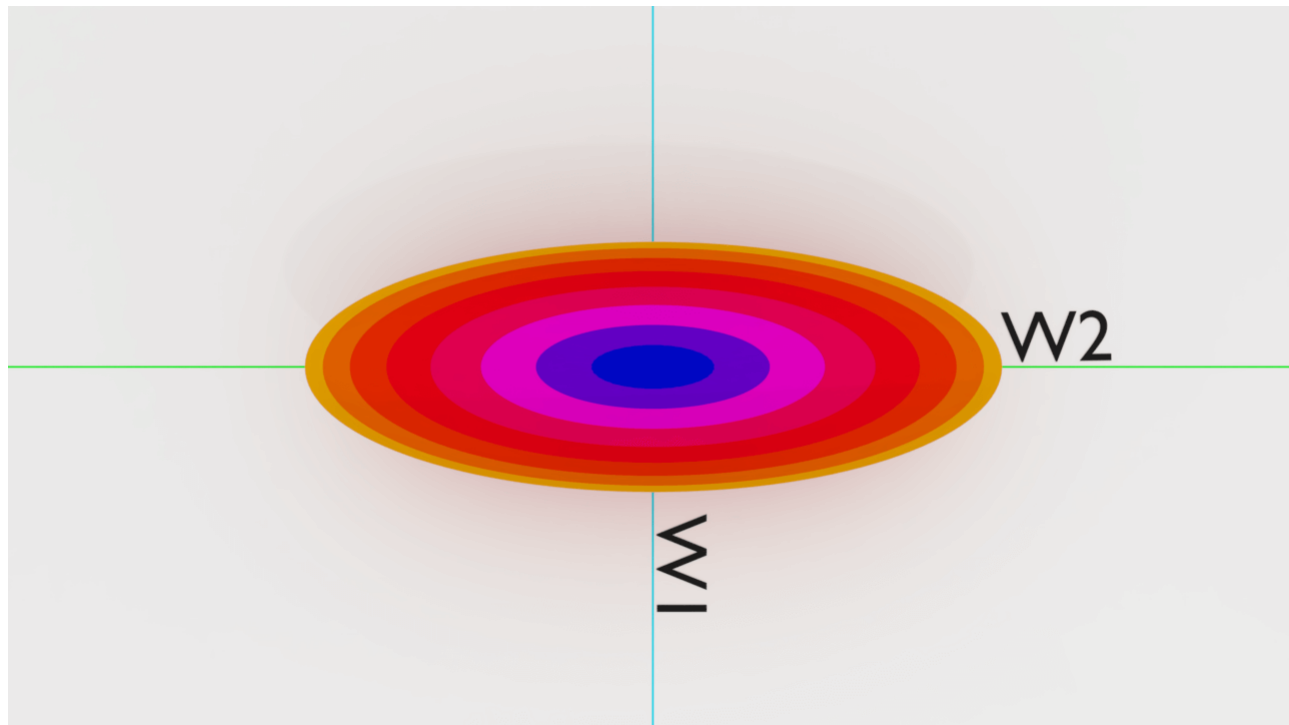


Let's look at the top view to appreciate the trouble here:

Un-normalized



Normalized



If features are found at a similar scale, then weights would be in a similar scale, and then backprop would actually make sense, ponder!

Why limit normalization to images only then?

Batch Normalization [.\(http://jmlr.org/proceedings/papers/v37/ioffe15.pdf\)](http://jmlr.org/proceedings/papers/v37/ioffe15.pdf) (BN), introduced in 2015– and is now the defacto standard for all CNNs and RNNs.

Batch Normalization solves a problem called the **Internal Covariate shift**. To understand BN we need to understand what is CS.

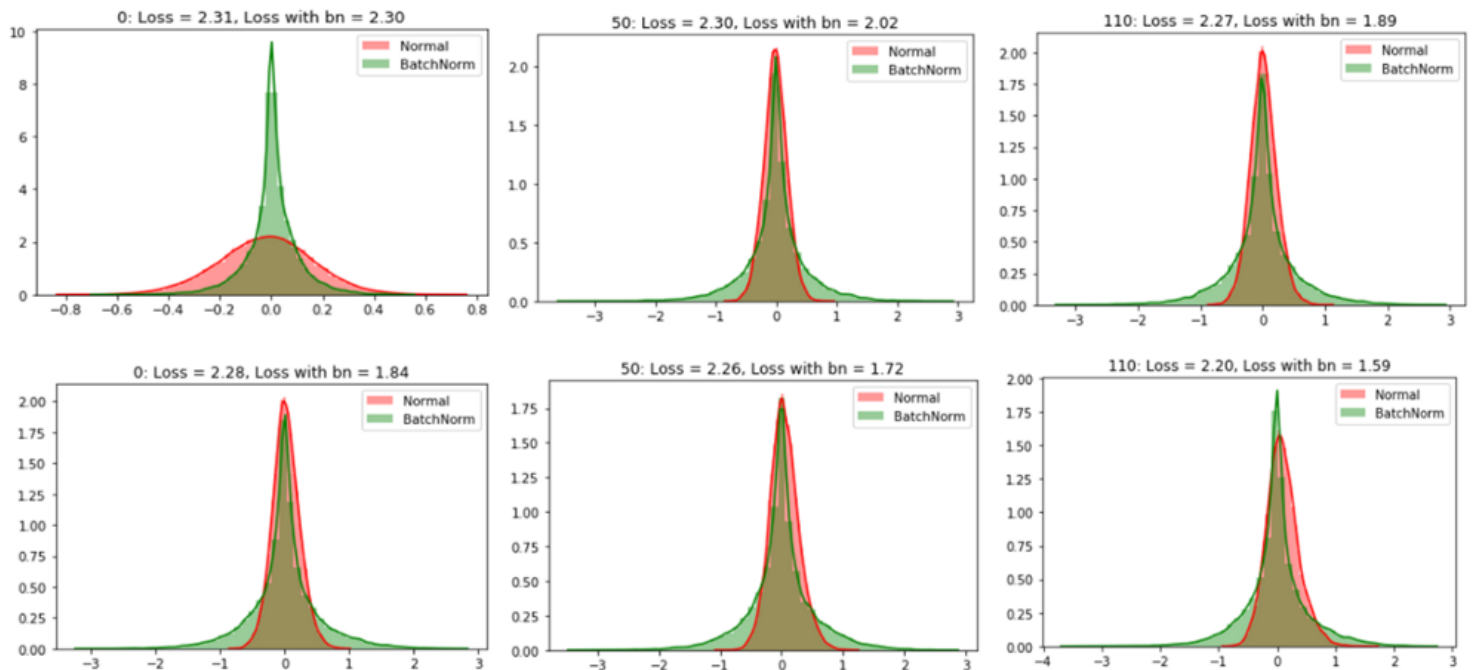
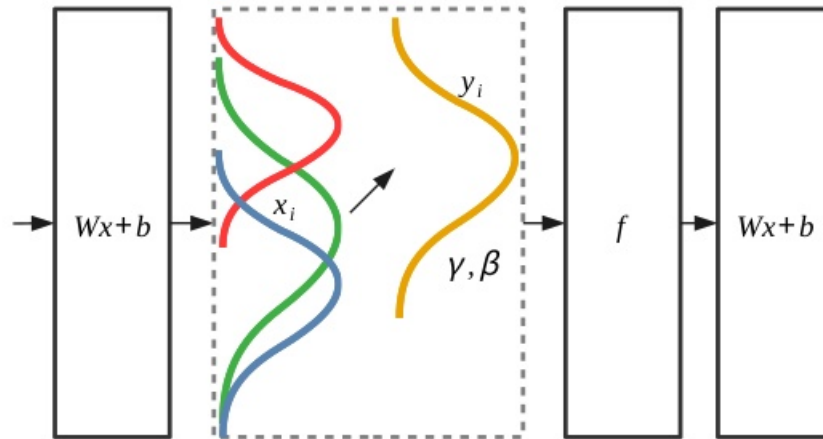
Covariate means input features. Covariate shift means that the distribution of the features is different in different parts of the training/test data.

Internal Covariate shift refers to changes within the neural network, between layers. A kernel always giving out higher activation makes next layer kernels always expect this higher activation and so on.

Imagine what would happen if One channel ranges between -1 to 1 and another between -1000 to 1000

Batch normalization

Ensure the output statistics of a layer are fixed.

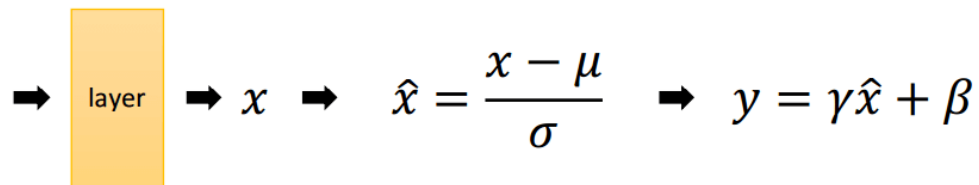


Very Deep nets can be trained faster and generalize better when the distribution of activations is kept normalized during BackProp.

We regularly see Ultra-Deep ConvNets like Inception, Highway Networks, and ResNet. And giant RNNs for speech recognition, [machine translation](https://research.googleblog.com/2016/09/a-neural-network-for-machine.html) [_](https://research.googleblog.com/2016/09/a-neural-network-for-machine.html)(<https://research.googleblog.com/2016/09/a-neural-network-for-machine.html>), etc.

Let's look at some math:

Batch Normalization (BN)



- μ : mean of x in mini-batch
- σ : std of x in mini-batch
- γ : scale
- β : shift
- μ, σ : functions of x , analogous to responses
- γ, β : parameters to be learned, analogous to weights

This is how we implement "batch" normalization:

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\dots m}\};$

Parameters to be learned: γ, β

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

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

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

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

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

Some crucial points:

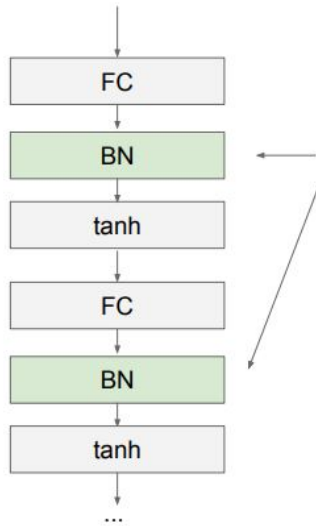
- How many gammas and betas? And what do they depend on?

BIAS GET'S SUBTRACTED OUT IN BATCH NORMALIZATION

ReLU before BN or BN before ReLU?

Batch Normalization

[Ioffe and Szegedy, 2015]



Usually inserted after Fully Connected or Convolutional layers, and before nonlinearity.

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}[x^{(k)}]}{\sqrt{\text{Var}[x^{(k)}]}}$$

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 6 - 56

April 20, 2017

Read this research paper: <http://proceedings.mlr.press/v37/ioffe15.pdf> <http://proceedings.mlr.press/v37/ioffe15.pdf>

Additional References:

Normalizing Activations in a Network (C2W3L04)



Fitting Batch Norm Into Neural Networks (C2W3L05)



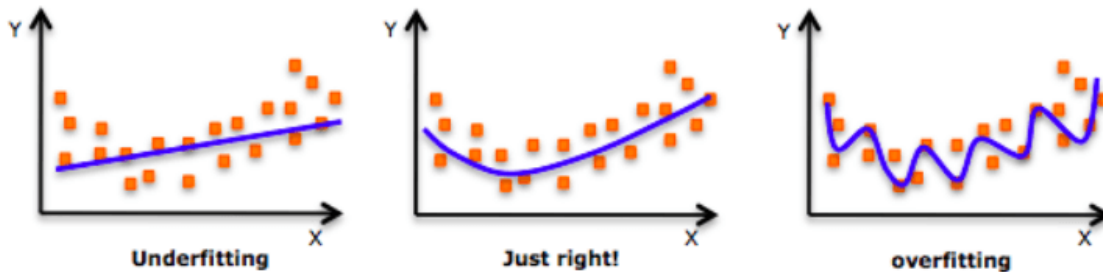
Why Does Batch Norm Work? (C2W3L06)



Batch Norm At Test Time (C2W3L07)




Regularization



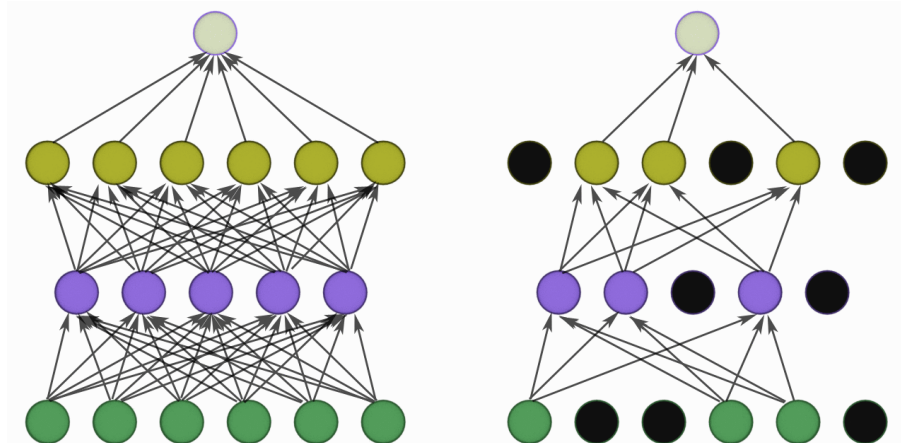
Regularization is a key component in preventing overfitting. Also, some techniques of regularization can be used to reduce model parameters while maintaining accuracy, for example, to drive some of the parameters to zero. This might be desirable for reducing the model size or driving down the cost of evaluation in a mobile environment where processor power is constrained.

Most common techniques of regularization used nowadays in the industry:

1. *Dataset augmentation* - An overfitting model (neural network or any other type of model) can perform better if the learning algorithm processes more training data.

2. *Early stopping* - Early-stopping combats overfitting interrupting the training procedure once the model's performance on a *validation* set gets worse.



3. Dropout layer - DropOut (What gets dropped? How does it help? What happens during inference?)



4. Weight penalty L1 and L2 - L1 and L2 are the most common types of regularization. These update the general cost function by adding another term known as the regularization term.

1. L1 Regularization(Lasso Regression)

L1 regularization adds an L1 penalty equal to the absolute value of the **magnitude of coefficients**. When our input features have weights closer to zero this leads to sparse L1 norm. In the Sparse solution the majority of the input features have zero weights and very few features have non zero weights.

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n |\theta_j| \right]$$

Features:

L1 penalizes the sum of the absolute value of weights.

L1 has a sparse solution

L1 generates a model that is simple and interpretable but cannot learn complex patterns

L1 is robust to outliers

2. L2 Regularization(Ridge regularization)

L2 regularization is similar to L1 regularization. But it adds a **squared magnitude of coefficient** as penalty term to the loss function. L2 will *not* yield sparse models and all coefficients are shrunk by the same factor (none are eliminated like L1 regression)

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

Features:

L2 regularization penalizes the sum of square weights.

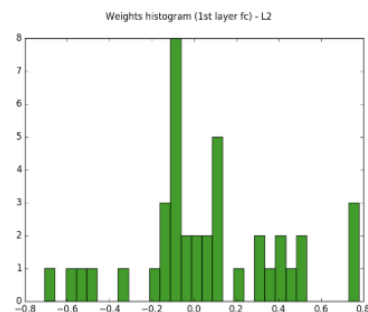
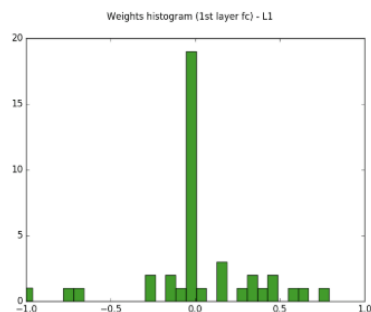
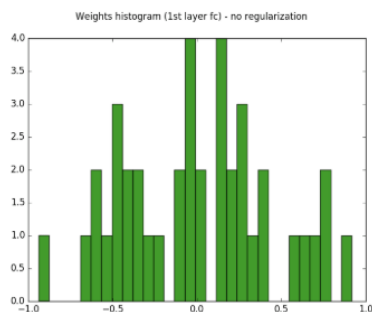
L2 has a non-sparse solution

L2 regularization is able to learn complex data patterns

L2 has no feature selection

L2 is not robust to outliers

3.



$$E = \underbrace{\frac{1}{2} * \sum (t_k - o_k)^2}_{\text{plain error}} + \underbrace{\frac{\lambda}{2} * \sum w_i^2}_{\text{weight penalty}}$$

elegant math



simple math



$$\frac{\partial E}{\partial w_{jk}} \quad \text{gradient}$$

$$\Delta w_{jk} = \underbrace{\eta}_{\text{learning rate}} * \underbrace{\left[x_j * (o_k - t_k) * o_k * (1 - o_k) \right] + [\lambda * w_{jk}]}_{\text{signal}}$$

learning
rate

signal

L2

`torch.optim.SGD` (`params`, `lr=<required parameter>`, `momentum=0`, `dampening=0`, `weight_decay=0`, `nesterov=False`)

weight_decay (*python:float, optional*) – weight decay (L2 penalty) (default: 0)

L1

```
l1_crit = nn.L1Loss(size_average=False)
```

```
reg_loss = 0
```

```
for param in model.parameters():
```

```
reg_loss += l1_crit(param)
```

```
factor = 0.0005
```

```
loss += factor * reg_loss
```

Assignment:

Your assignment 7 will demand these things:

1. Change your code in such a way that all of these are in their respective files:
 1. model
 2. training code
 3. testing code
 4. regularization techniques (dropout, L1, L2, etc)
 5. dataloader/transformations/image-augmentations
 6. misc items like finding misclassified images
2. So, while doing assignment 6, please think how would you be able to do this in the next assignment
3. Your assignment 6 is:
 1. take your 5th code
 2. run your model for 40 epochs for each:
 1. without L1/L2
 2. with L1
 3. with L2
 4. with L1 and L2
 3. draw 2 graphs to show the validation accuracy change and loss change. This graph must have proper legends and it should be clear what we are looking at.
 4. find any 25 misclassified images for L1 and L2 models. You MUST show the actual and predicted class names.
 5. make all the images available on Github Readme page, so you can upload the images for your assignment (you can upload them somewhere else as well for add image url).
 6. submit the Github link for your notebook with logs, and also upload the images in the S6-Assignment Solution.

Here are some of the questions for S6-Assignment-Solution:

1. Upload the Validation Accuracy Change Graph
2. Upload the Loss Change Graph
3. Upload the 25 misclassified images plot for L1
4. Upload the 25 misclassified images plot for L2
5. Explain your observation w.r.t. L1 and L2's performance in the regularization of your model.

Session Video - Wednesday

EVA 4 - Session 6 Wednesday



Sunday Video:

EVA 4 Session 6 - Sunday

