Session 8 - Batch Normalization & Regularization

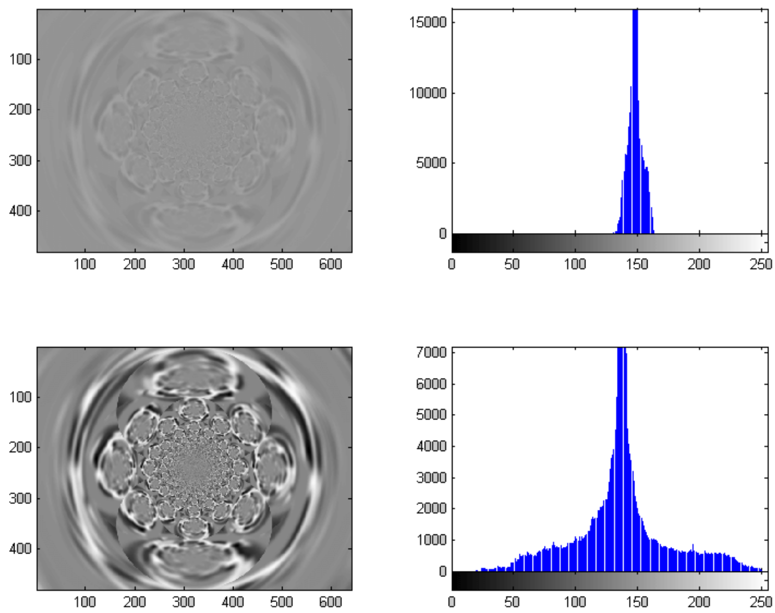
Due 17 Jun by 13:00 **Points** None
Available after 17 Jun at 13:00

SESSION 8 - BATCH NORMALIZATION & REGULARIZATION

Image Normalizing

In image processing, **normalization** is a process that changes the range of pixel intensity values.

For example, if the intensity range of the image if 50 to 180 and the desired range is 0 to 255, the process entails subtracting 50 from each pixel intensity, making the range 0 to 130. Then each pixel is multiplied by 255/130, making the range 0 to 255.



Reference: [Normalizing input (LeCun et al 1998 “Efficient Backprop”)](http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf)

(http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf)

Why redistribution of data?

Why the redistribution of data is important for us?

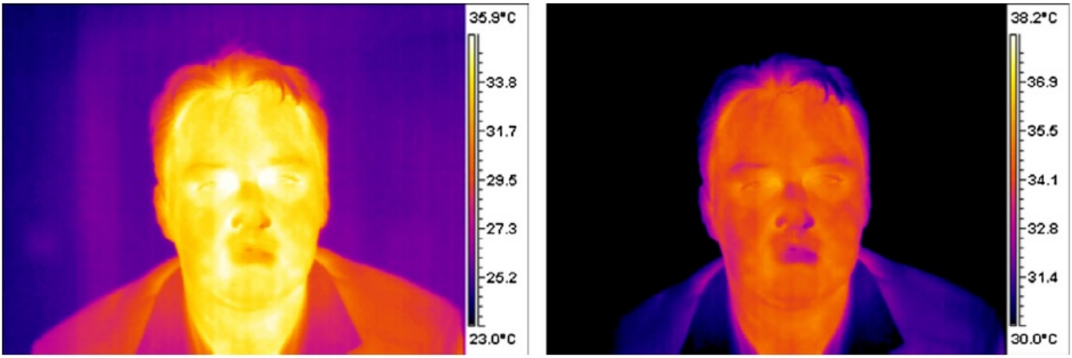
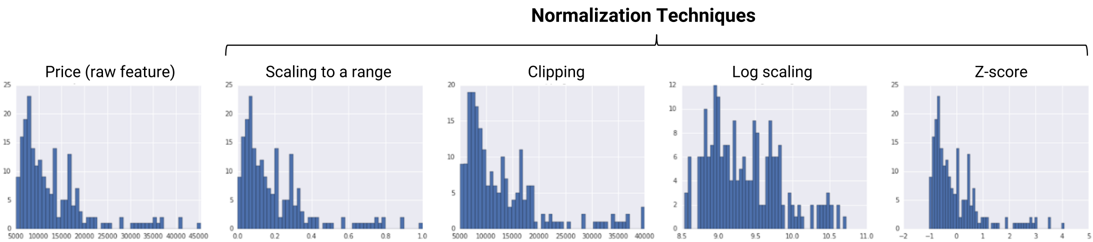
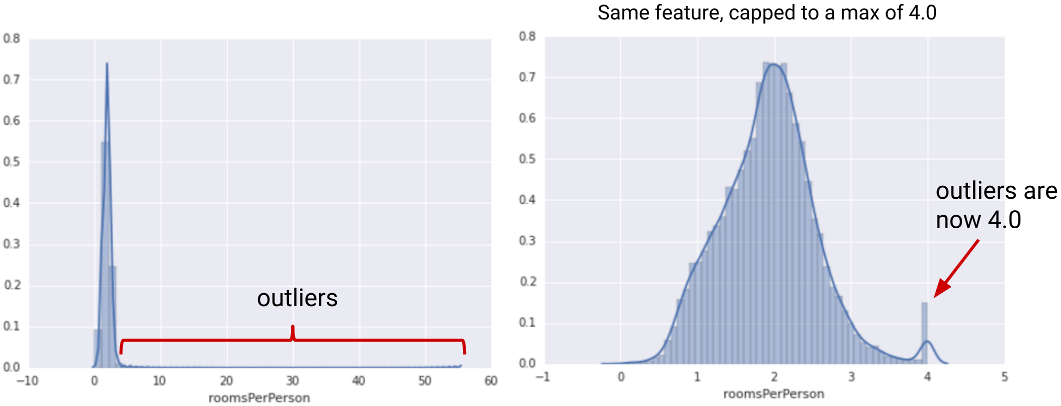


Fig. 10. Examples of thermal face images. Raw thermal image (left). Normalized thermal image (right).





[REF ⇨ (https://developers.google.com/machine-learning/data-prep/transform/normalization)](https://developers.google.com/machine-learning/data-prep/transform/normalization)

Normalization is not Equalization

The normalization 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,

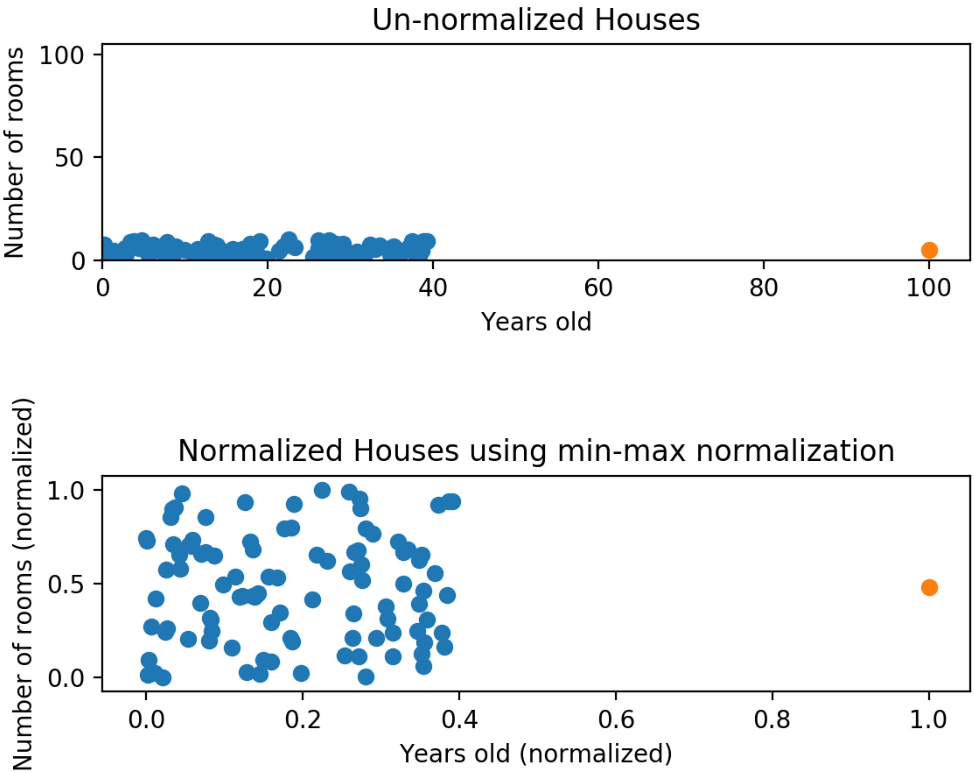
but if you want a more even distribution of intensity levels then equalize can help with that. SOURCE (http://www.roborealm.com/forum/index.php?thread_id=4350)


original image

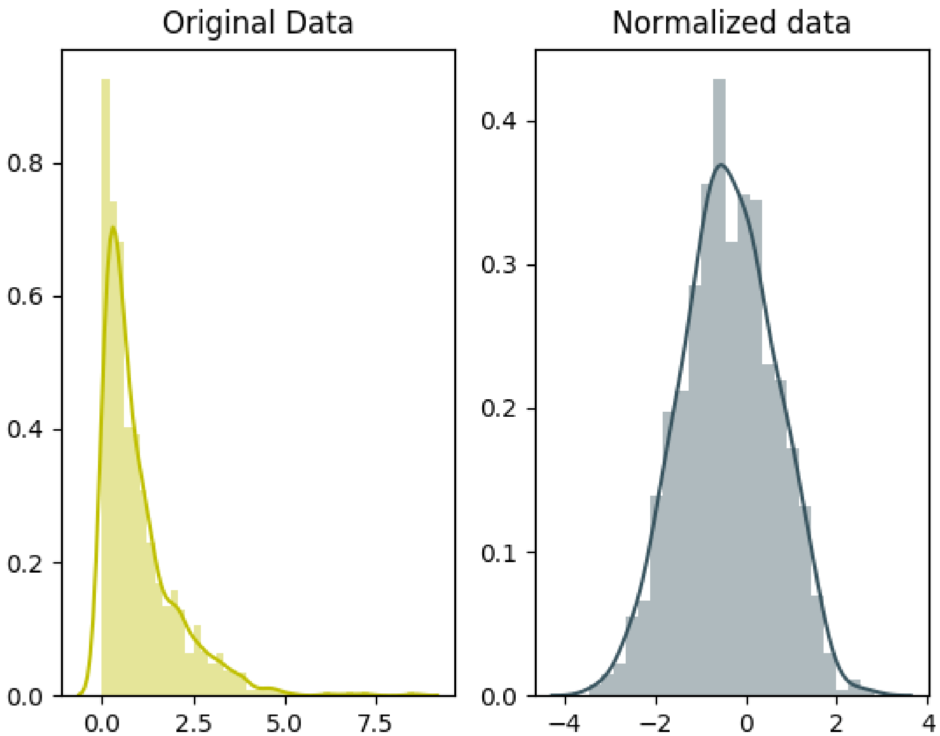

normalized image


equalized image

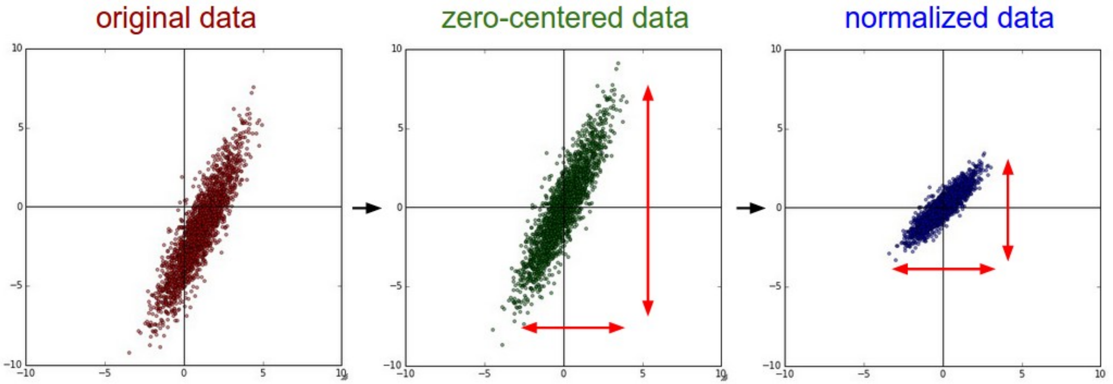
Let's look at normalization working on other kinds of data.



outlier.png

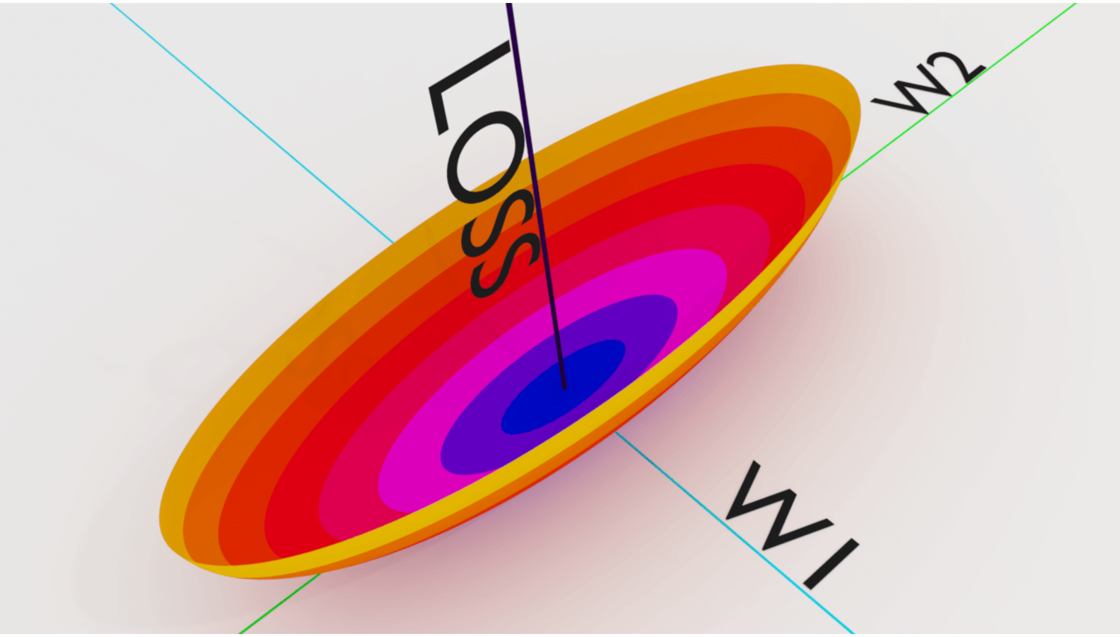


How to normalize?

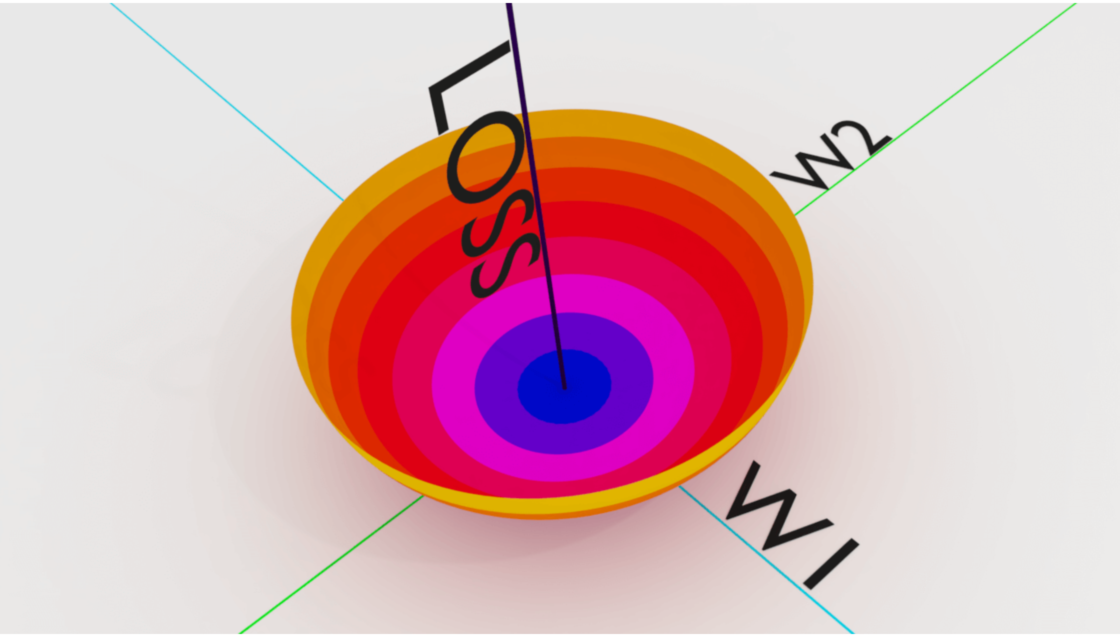


Loss & Weights with/without normalization

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

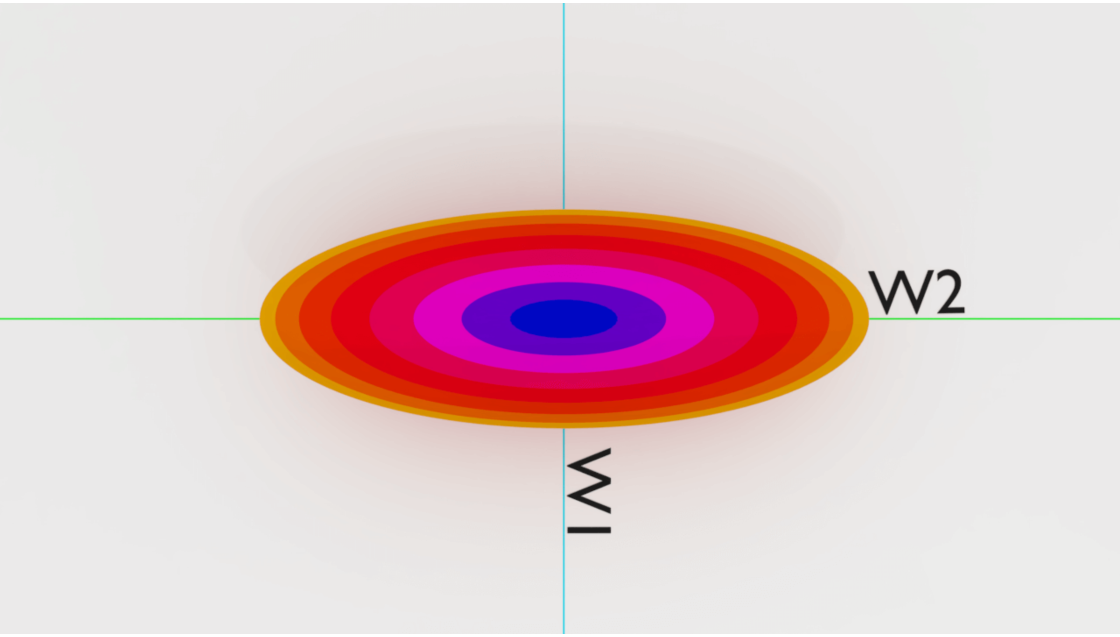


If we had normalized our kernels (indirect channels), this is how it would look like:

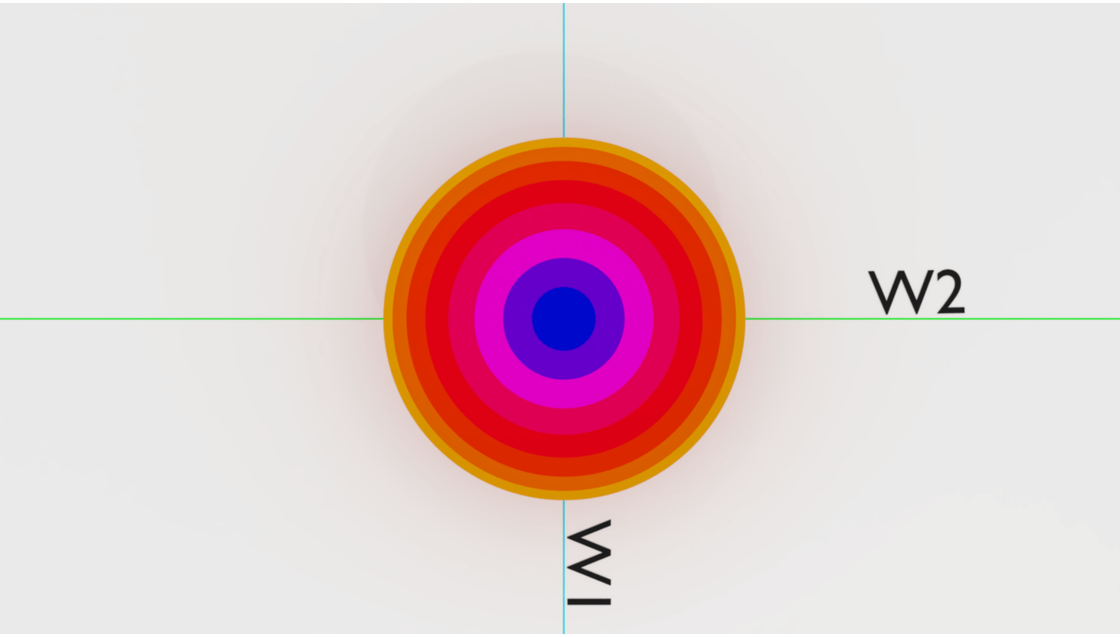


Let's look at the top view to appreciate the trouble here:

Un-normalized



Normalized



If features are found at a similar scale, then weights would be on a similar scale,

and then backprop would actually make sense, ponder!

Why limit normalization to images only then?

Batch Normalization

Batch Normalization (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.*

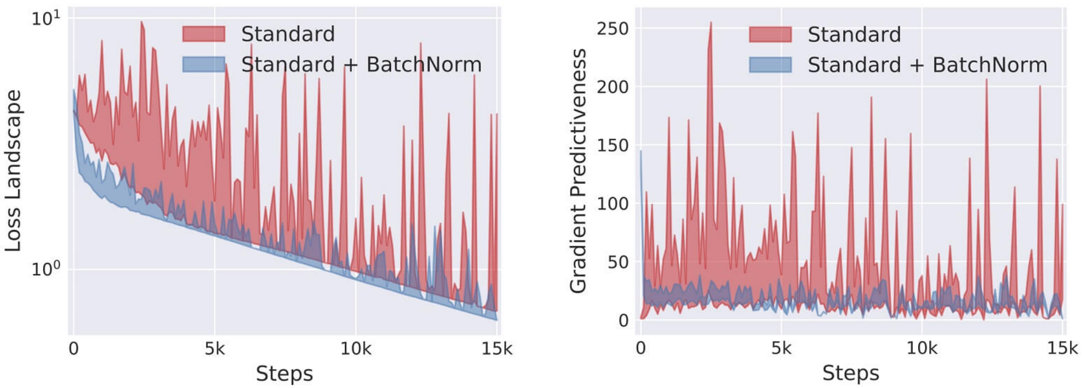
Covariate shift means that the distribution of the features is different in different parts of the training/test data.

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*.

But this has been proven wrong lately. Observe the weight distribution over a training period:

SO WHAT IS IT DOING?

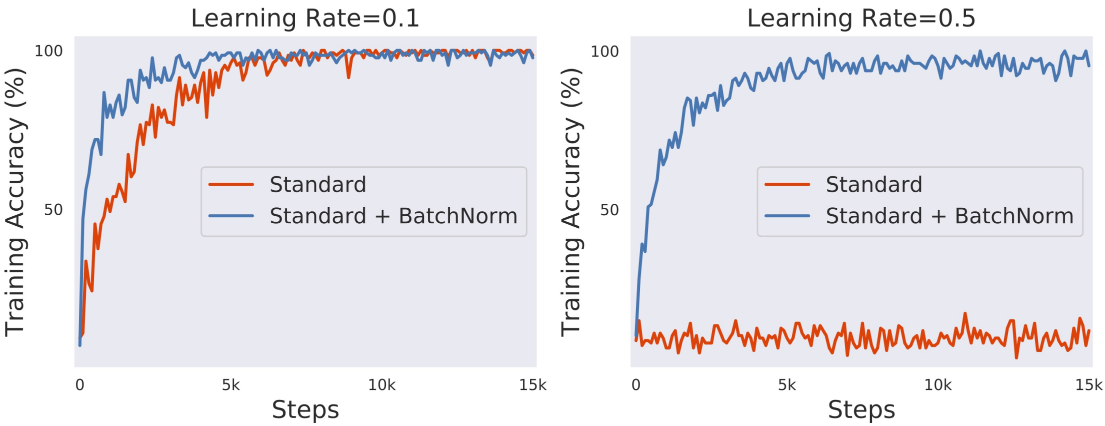
Let's look at the variation of the value of the loss and gradient predictiveness:



Imagine what would happen if one channel ranges from -1 to 1

and another between -1000 to 1000 (go back to the oval and circular loss images above)

Just see how it **elevates (https://gradientscience.org/batchnorm/)** a bit of a problem for us to get the right learning rates!



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) , etc.

Explain BN to a 5-year-old!

It reduces the dependency of the network's output on the scale of its inputs!

This reduces overfitting!

This allows us to train at higher learning rates!

RUMORS

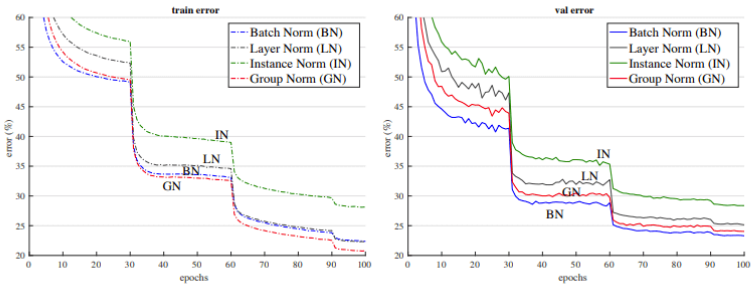


Figure 4. **Comparison of error curves** with a batch size of **32 images**/GPU. We show the ImageNet training error (left) and validation error (right) *vs*. numbers of training epochs. The model is ResNet-50.

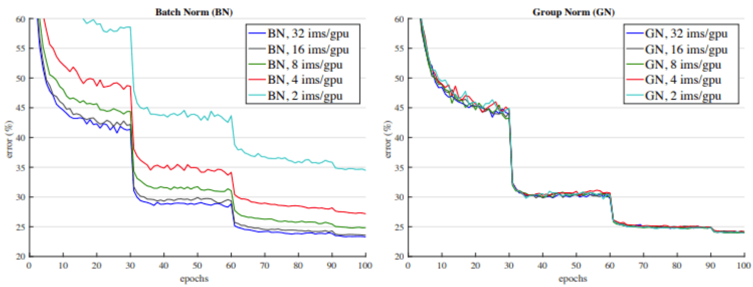


Figure 5. **Sensitivity to batch sizes**: ResNet-50's validation error of BN (left) and GN (right) trained with 32, 16, 8, 4, and 2 images/GPU.

BN Mathematics

This is how we implement "batch" normalization:

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

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

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

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

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

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

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?

[Why position doesn't matter (https://youtu.be/jhUZ800C650?t=3182)](https://youtu.be/jhUZ800C650?t=3182) ?

Batch Normalization calculates its normalization statistics over each minibatch of data separately while training but during inference a moving average of training statistics are used, simulating the expected value of the normalization statistics.

Read this research paper:

http://proceedings.mlr.press/v37/ioffe15.pdf

[(http://proceedings.mlr.press/v37/ioffe15.pdf)](http://proceedings.mlr.press/v37/ioffe15.pdf)

What are SOTAs using today?

NLP> Layer Normalization

Vision> BN

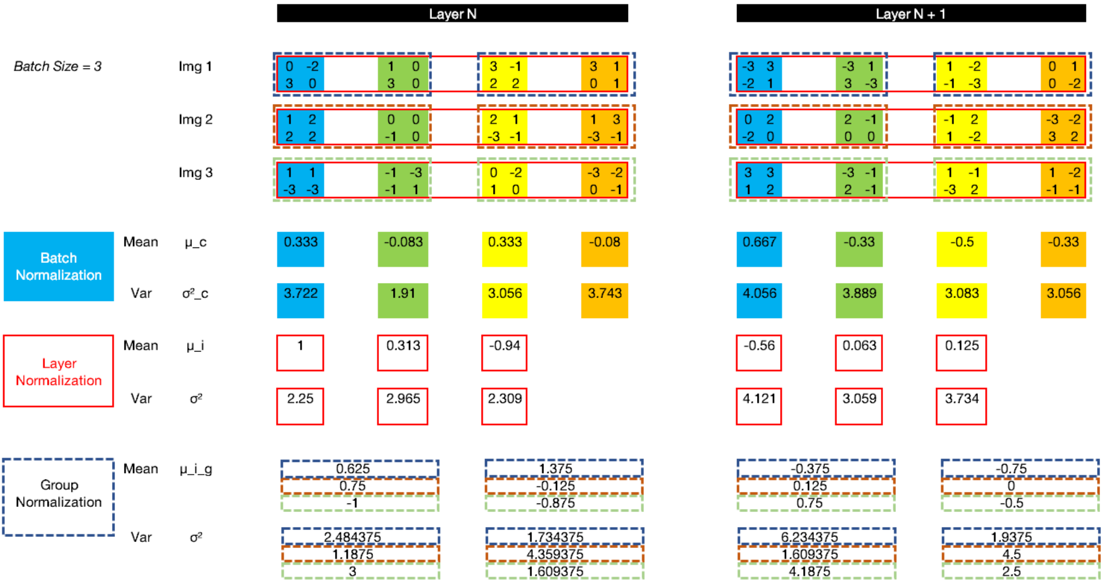
Accuracy	Paper	Date	Normalization
90.2%	Meta Pseudo Labels	1 March 2021	Batch Normalization Decay/Momentum
89.2%	NFNet-F4	11 Feb 2021	Normalization Free (but is not)

88.64%	ALIGN	11 Feb 2021	Batch Normalization
88.61%	SAM	29 April 2021	Batch Normalization
88.55%	ViT-H/14	22 Oct 2020	Group Norm | Layer Norm
78.25%	ResNet-101	10 Dec 2015	Batch Normalization

Batch Normalization Decay or Momentum

$$\mu_{mov} = \alpha * \mu_{mov} + (1 - \alpha) * \mu_B$$
$$\sigma^2_{mov} = \alpha * \sigma^2_{mov} + (1 - \alpha) * \sigma^2_B$$

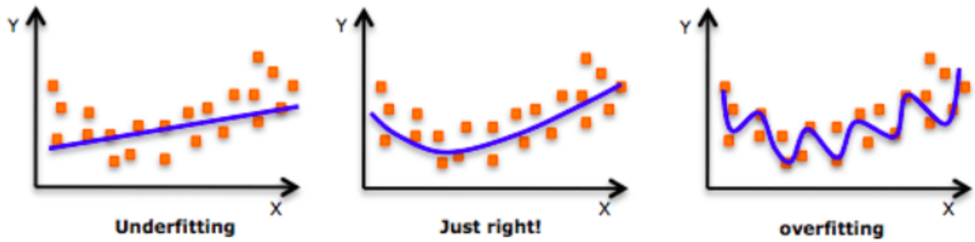
Group, Instance, and Layers Normalization



[File (https://canvas.instructure.com/courses/2734471/files/139537291?wrap=1)](https://canvas.instructure.com/courses/2734471/files/139537291?wrap=1)

(https://canvas.instructure.com/courses/2734471/files/139537291/download?download_frd=1)

Regularization



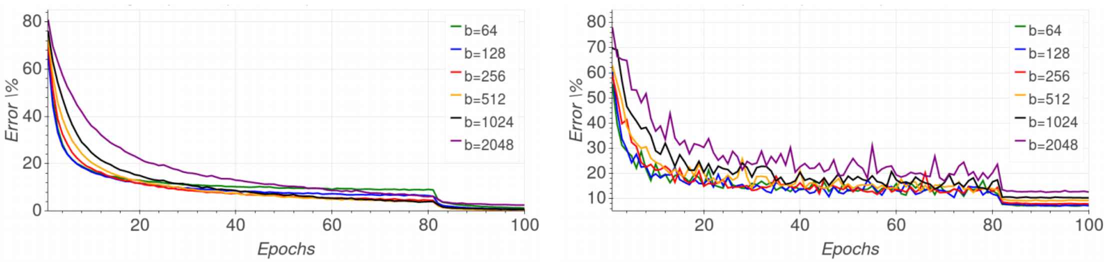
Regularization is a key component in preventing overfitting.

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.

Regularization effect of batch size



(a) Training error

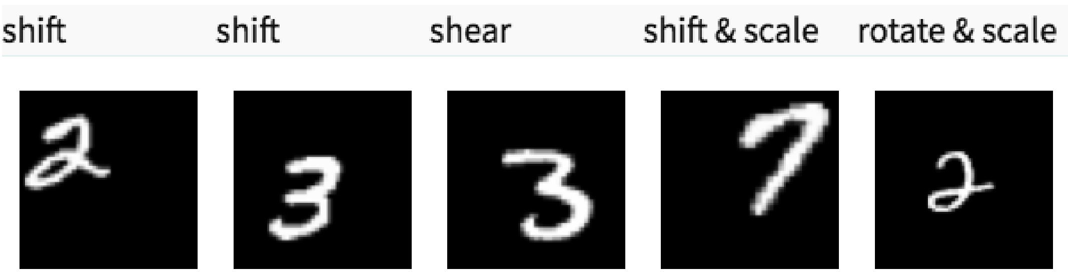
(b) Validation error

Figure 1: Impact of batch size on classification error

Most common techniques of regularization used nowadays in the industry:

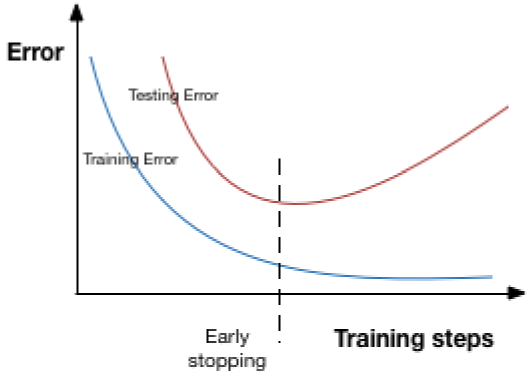
Dataset augmentation

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

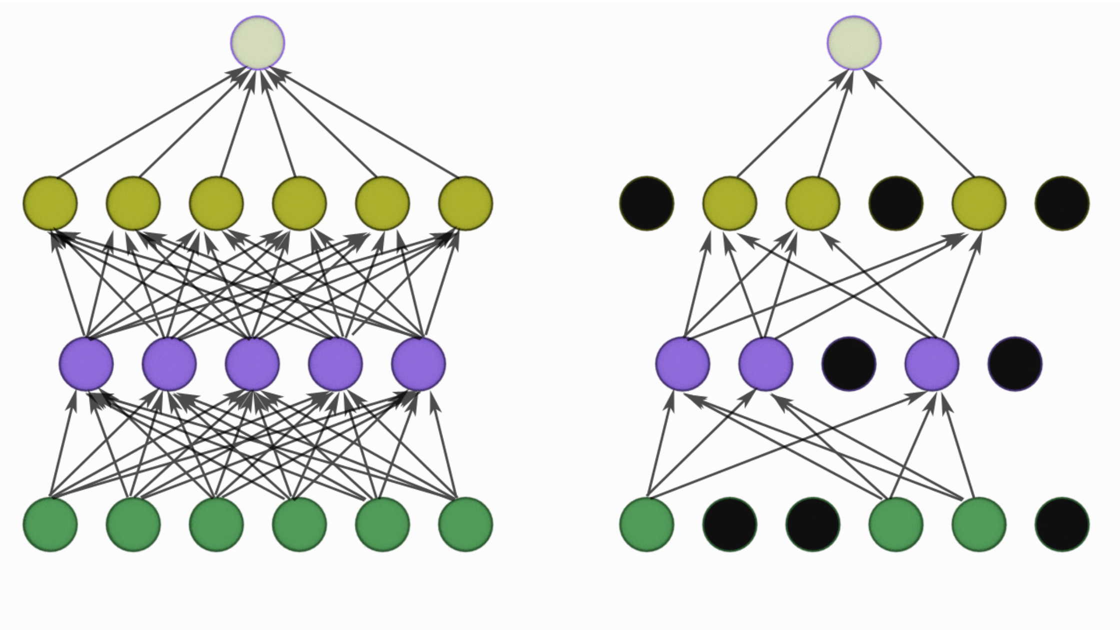


Early stopping

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



Dropout



(What gets dropped? How does it help? What happens during inference?)

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.

L1 & L2 Regularization

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 a 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

L2 Regularization(Ridge regularization)

L2 regularization is similar to L1 regularization.
But it adds a ***squared magnitude of coefficient*** as a 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

L1/L2 in Pytorch

L2

```
optimizer_sgd = torch.optim.SGD(params, lr=10e-
4, momentum=0, dampening=0, weight_decay=0, nesterov=False)

optimizer_adam = torch.optim.Adam(model.parameters(), lr=1e-4,
weight_decay=1e-5)
```

L1

```
loss = mse(pred, target)
l1 = 0
for p in net.parameters():
 l1 = l1 + p.abs().sum()
loss = loss + lambda_l1 * l1
loss.backward()
optimizer.step()
```

Assignment

Your Assignment is:

1. Change the dataset to CIFAR10
2. Make this network:
    1. C1 C2 ***c3 P1*** C3 C4 C5 ***c6 P2*** C7 C8 C9 GAP C10
    2. Keep the parameter count less than 50000
    3. Try and add one layer to another
    4. Max Epochs is 20
3. You are making 3 versions of the above code (in each case achieve above 70% accuracy):
    1. Network with Group Normalization
    2. Network with Layer Normalization
    3. Network with Batch Normalization
4. Share these details
    1. Training accuracy for 3 models
    2. Test accuracy for 3 models

2. Find 10 misclassified images for the BN model, and show them as a 5x2 image matrix in 3 separately annotated images.
3. write an explanatory README file that explains:
    1. what is your code all about,
    2. your findings for normalization techniques,
    3. add all your graphs
    4. your collection-of-misclassified-images
4. Upload your complete assignment on GitHub and share the link on LMS

VIDEO

ERA V1 Session 8 Studio

Google Meet

ERA V1 Session 8 GM