Machine Learning (CE 40477)

Fall 2024

Ali Sharifi-Zarchi

CE Department
Sharif University of Technology

April 17, 2025

1. Batch Normalization
2. MLPs as Universal Approximators
3. References

1 Batch Normalization
- Why Batch Normalization?
- How Batch Normalization Works
- Batch Normalization During Inference Mode
- Batch Normalization Pros & Cons
- Batch Normalization in Practice
- Closing Takeaways on Batch Normalization

2 MLPs as Universal Approximators

3 References

1. Batch Normalization
   - Why Batch Normalization?
   - How Batch Normalization Works
   - Batch Normalization During Inference Mode
   - Batch Normalization Pros & Cons
   - Batch Normalization in Practice
   - Closing Takeaways on Batch Normalization
2. MLPs as Universal Approximators
3. References

Why Batch Normalization?

Problem: Internal Covariate Shift

- What does it mean, in simple terms?
- Let's say that we want to train a model and the ideal target output function that the model needs to learn is as below:

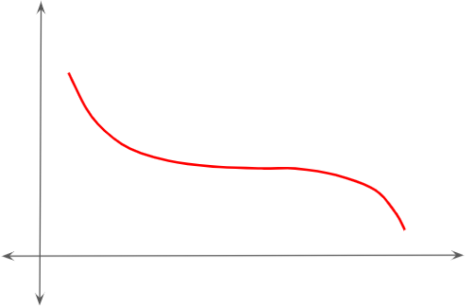


Figure adapted from Ketan Doshi. Batch Normalization: A Comprehensive Blog

Problem: Internal Covariate Shift

- Suppose that the training data values input to the model cover only a part of the range of output values. Therefore, the model can only learn a subset of the target function.

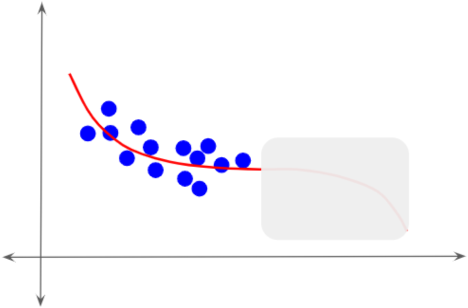


Figure adapted from Ketan Doshi. Batch Normalization: A Comprehensive Blog

Problem: Internal Covariate Shift

- The model has no idea about the rest of the target curve. It could be anything.

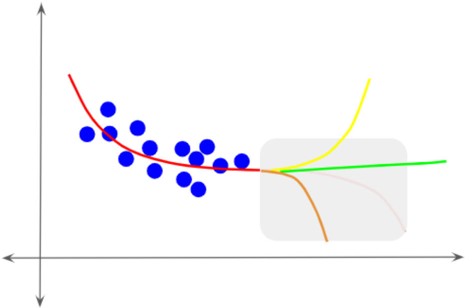


Figure adapted from Ketan Doshi. Batch Normalization: A Comprehensive Blog

Problem: Internal Covariate Shift

- Suppose we feed the model to the testing data as below.
- This has a very different distribution from the data that the model was initially trained with.
- The model cannot simply generalize its predictions for this new data.

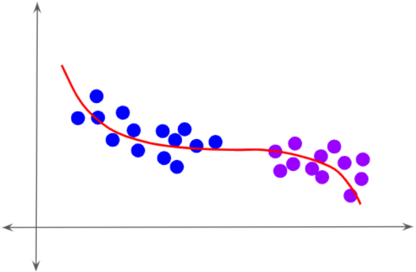


Figure adapted from Ketan Doshi. Batch Normalization: A Comprehensive Blog

Problem: Internal Covariate Shift

- Covariate Shift occurs when **the model is fed data with a different distribution than what it was previously trained with**, even though that new data still conforms to the same target function.
- For the model to figure out how to adapt to this new data, it has to re-learn some of its target output functions.
- **This slows down the training process.**

Problem: Internal Covariate Shift

What happens inside Deep Network Layers?

- Consider what happens when training a deep network: As we update the weights of earlier layers, the data distribution in the deeper layer keeps shifting.
- Deeper layers see new and varying patterns every time we update the weights in previous layers.

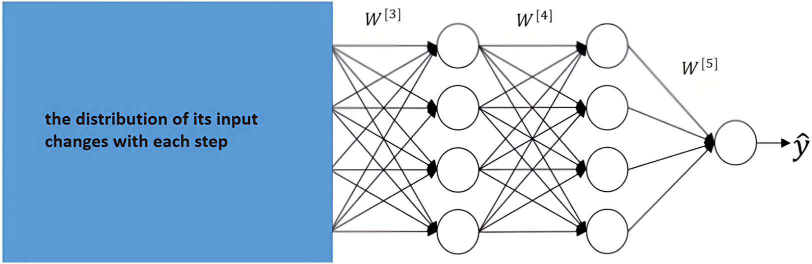


Figure adapted from Andrew Ng, Deep Learning Specialization Course

Problem: Internal Covariate Shift

What happens inside Deep Network Layers?

- The network must keep readjusting, which makes the learning process slower and more challenging.
- **In other words:** The network is constantly “re-learning” how to make predictions because the data it sees is never consistent.

Batch Normalization Solution

- **Goal:** Normalize inputs so that the mean is near 0 and the variance is close to 1.
- **How it helps:** Stabilizes learning, allowing for higher learning-rates and faster convergence.

1 Batch Normalization
Why Batch Normalization?
How Batch Normalization Works
Batch Normalization During Inference Mode
Batch Normalization Pros & Cons
Batch Normalization in Practice
Closing Takeaways on Batch Normalization

2 MLPs as Universal Approximators

3 References

How Batch Normalization Works

Process Overview

- For each mini-batch during training, batch normalization normalizes the inputs to a layer by adjusting their mean and variance.

How Batch Normalization Works

Steps in Batch Normalization

1. **Compute the Mean and Variance**
   For a given mini-batch, compute the mean μ_B and variance σ_B^2 of the inputs:

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i, \quad \sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

2. **Normalize the Inputs**
   Subtract the mean and divide by the standard deviation to get normalized activations:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

   where ϵ is a small constant added for numerical stability.

How Batch Normalization Works (Continued)

Steps in Batch Normalization

3. **Scale and Shift**

   After normalization, introduce learnable parameters γ and β that allow the model to scale and shift the normalized output:

$$y_i = \gamma \hat{x}_i + \beta$$

   This allows that the model to recover the original data distribution if necessary.

Smoothing the optimization space

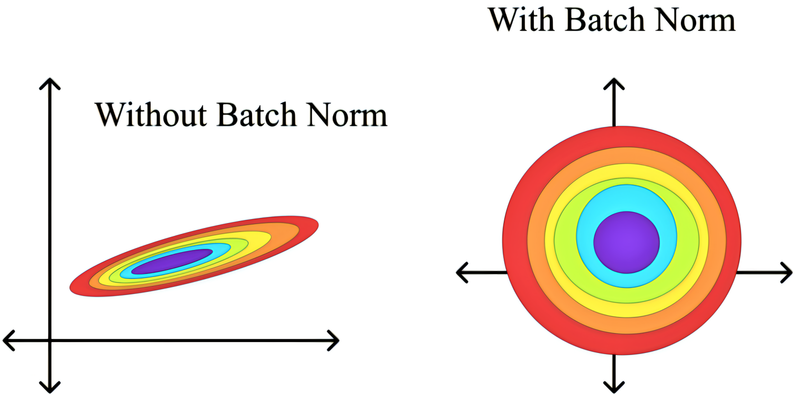
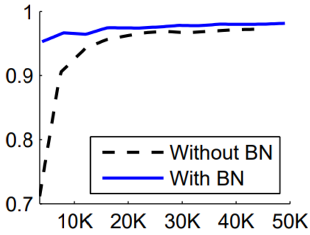


Figure adapted from Gustavo Albuquerque Lima. Ways to Improve Your Deep Learning Model - Batch Normalization and Adam: A Short Course

Effect of Batch Normalization

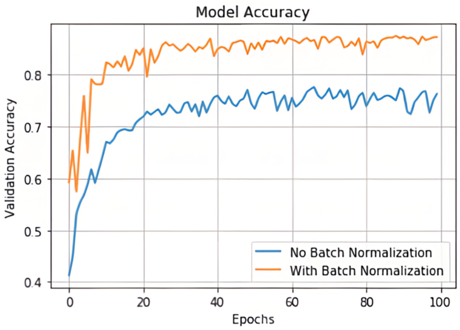
- Batch Normalization helps the network train faster and achieve higher accuracy.
- Batch Normalization **stabilizes the distribution** and reduces the internal covariate shift.



- The test accuracy of the `MNIST` network trained with and without Batch Normalization, vs the number of training steps.

Figure adapted from Johan Bjorck et al. Understanding Batch Normalization Paper, https://doi.org/10.48550/arXiv.1806.02375

Effect of Batch Normalization



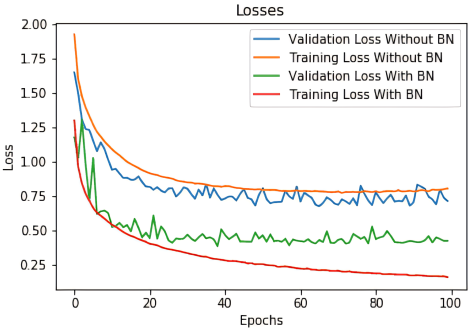


Figure adapted from Sunita Nayak, Batch Normalization in Deep Networks on LearnOpenCV

Effect of Batch Normalization on Gradients

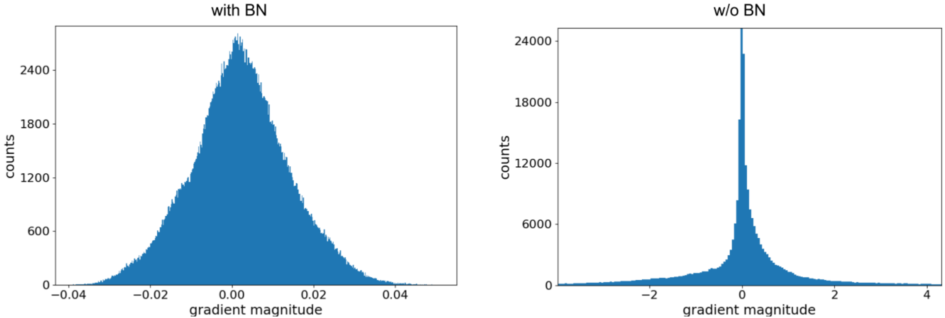


Figure adapted from Johan Bjorck et al. Understanding Batch Normalization Paper, https://doi.org/10.48550/arXiv.1806.02375

Effect of Batch Normalization on Gradients

The main benefit of batch normalization is that it reduces the dependency of gradient on the scale of input and parameters:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial y} \cdot \frac{\partial y}{\partial \hat{x}} \cdot \frac{\partial \hat{x}}{\partial x} \tag{1}$$

Where:

- $\frac{\partial y}{\partial \hat{x}} = \gamma$
- $\frac{\partial \hat{x}}{\partial x} = \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$

Thus, the gradient becomes:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\gamma}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \cdot \frac{\partial \mathcal{L}}{\partial y} \tag{2}$$

How Batch Normalization Smooth the Loss Landscape

The smoothing effect of batch normalization can be understood by observing how it constrains the gradient magnitudes. The expression shows that:

$$\frac{\partial \mathcal{L}}{\partial x} \text{ is scaled by } \frac{1}{\sqrt{\sigma_{\mathscr{B}}^2 + \epsilon}} \tag{3}$$

This consistent scaling results in a smooth loss landscape because:

- It stabilizes the gradient flow, ensuring controlled optimization step sizes.
- Reduces the risk of large oscillations or abrupt changes in the loss landscape.
- Makes the optimization process less likely to be trapped in local minima or saddle points.

1. Batch Normalization
   - Why Batch Normalization?
   - How Batch Normalization Works
   - Batch Normalization During Inference Mode
   - Batch Normalization Pros & Cons
   - Batch Normalization in Practice
   - Closing Takeaways on Batch Normalization
2. MLPs as Universal Approximators
3. References

Batch Normalization During Inference Mode

Inference Mode:

- During inference (when predicting new data), batch statistics (mean and variance) are replaced with **moving averages** collected during training.

What is a Moving Average?

- A **moving average** tracks the average of a value (like batch mean or variance) over time.
- At each training step, it updates using:

$$\mu_{\text{moving}} \leftarrow (1-\alpha)\mu_{\text{moving}} + \alpha\mu_{\text{batch}}$$

- Here, α is a momentum parameter (e.g., 0.1), controlling how much weight to give to the new batch's value.
- This "momentum" is a hyperparameter that is used only for Batch Norm moving averages and should not be confused with the momentum that is used in the Optimizer.
- These moving averages approximate the global statistics used during inference.

1 Batch Normalization
Why Batch Normalization?
How Batch Normalization Works
Batch Normalization During Inference Mode
Batch Normalization Pros & Cons
Batch Normalization in Practice
Closing Takeaways on Batch Normalization

2 MLPs as Universal Approximators

3 References

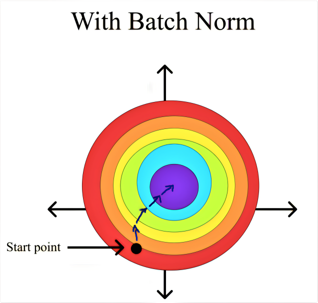
Batch Normalization Pros

Pros

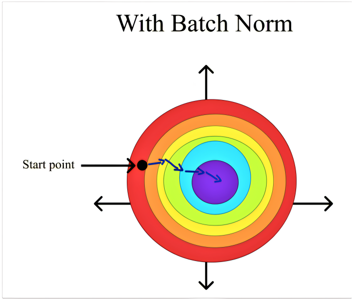
- **Faster Convergence:** Empirical results support that models with batch normalization converge faster and achieve higher accuracy, even with **higher learning rates**.
- **Reduced Sensitivity to Weight Initialization:** Mitigate the dependency on careful weight initialization.
- **Acts as Regularization:** Batch normalization can help reduce overfitting.
- **Reduces Vanishing/Exploding Gradients:** Maintains stable gradients throughout deep networks.

Batch Normalization Pros

Why Using Batch Normalization Reduces Sensitivity to Weight Initialization?



(a)



(b)

Figure adapted from Gustavo Albuquerque Lima. Ways to Improve Your Deep Learning Model - Batch Normalization and Adam: A Short Course

Batch Normalization Pros

Why Does Batch Normalization Reduce Sensitivity to Weight Initialization?

- Batch normalization smooths the optimization landscape, reducing the dependency on initial weights.
- This allows the model to converge to a minimum efficiently, **regardless of where optimization begins**.

Batch Normalization Cons

Cons

- **Batch Size Sensitivity:** Performance may depend on batch size. Very small batches potentially provide unstable statistics.
- **Computational Overhead:** Increases computational-overhead during training.
- **Behavior During Inference:** Switching from batch statistics to moving averages during inference may cause slight discrepancies.

1 Batch Normalization
Why Batch Normalization?
How Batch Normalization Works
Batch Normalization During Inference Mode
Batch Normalization Pros & Cons
Batch Normalization in Practice
Closing Takeaways on Batch Normalization

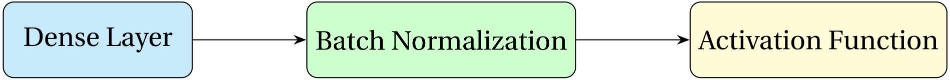
2 MLPs as Universal Approximators

3 References

Batch Normalization in Practice

Where to Apply

- **Typical Location:** Apply after the linear transformation (e.g., after a dense layer) and before the activation function.
- **Layer Placement:**



1. Batch Normalization
   - Why Batch Normalization?
   - How Batch Normalization Works
   - Batch Normalization During Inference Mode
   - Batch Normalization Pros & Cons
   - Batch Normalization in Practice
   - Closing Takeaways on Batch Normalization
2. MLPs as Universal Approximators
3. References

Batch Normalization in Practice

- **Key Point:** Batch normalization stabilizes and accelerates training while offering regularization benefits.
- **Impact on Training:** Facilitates efficient training of deeper networks with less hyperparameter tuning.

1 Batch Normalization

2 MLPs as Universal Approximators

3 References

Universal Approximation Theorem

Key Concept

- The Universal Approximation Theorem states that a feedforward neural network with:
  - A single hidden layer
  - Sufficient number of hidden neurons
  - Appropriate activation functions (e.g., sigmoid)

  Can approximate any continuous function on a **compact subset** of $\mathbb{R}^n$ to any desired accuracy.

Understanding Compact Sets

What is a Compact Set?

- In the context of the Universal Approximation Theorem, approximation is guaranteed on a **compact subset** of $\mathbb{R}^n$.
- A set is compact if it is both:
  - **Bounded**: Enclosed within a finite space.
  - **Closed**: Contains all its boundary points.
- Compact sets ensure certain mathematical properties that enable reliable function approximation by the MLP within that region.

Breaking Down Complex Functions

- Idea: Complex functions can be decomposed into multiple smaller parts, each represented by a simpler function.
- By combining a series of simpler functions (like square pulses), the target function can be closely approximated.

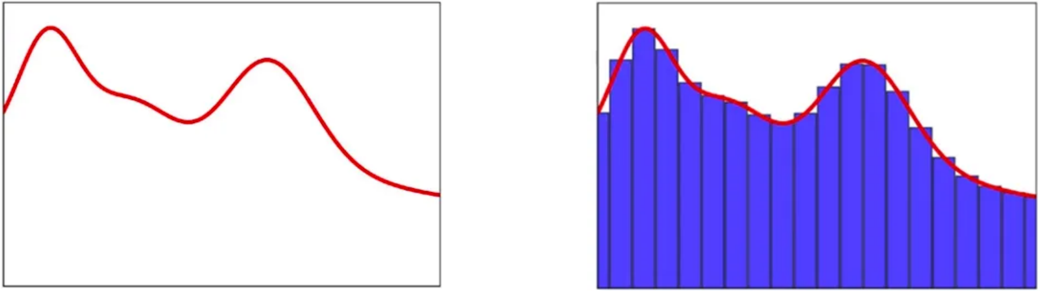


Figure adapted from Niranjan Kumar, Illustrative Proof of Universal Approximation Theorem

MLPs as Universal Approximators

- By constructing a series of these `Square Pulse` functions, we can approximate any continuous function mapping from input to output.
- A simple 3-unit MLP with a summing output unit can generate a square pulse.
- Therefore, an MLP with enough units and the right configuration is a universal approximator!

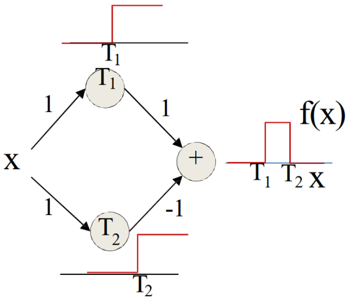


Figure adapted from Dr. Mahdieh Soleymani Baghshah, Deep Learning Course

Contributions

These slides are authored by:

- Faezeh Sarlakifar
- Sogand Salehi

1. Batch Normalization
2. MLPs as Universal Approximators
3. References

[1] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *CoRR*, vol. abs/1502.03167, 2015.

[2] A. Ng and K. Katanforoosh, *CS230 Lecture Notes.*
Stanford University, 2018.

[3] F.-F. Li and Z. Durante, *CS231n Lectures.*
Stanford University, 2024.
Updated June 3, 2024.

[4] J. Bjorck, C. P. Gomes, and B. Selman, “Understanding batch normalization,” *CoRR*, vol. abs/1806.02375, 2018.

[5] H. Li, Z. Xu, G. Taylor, and T. Goldstein, “Visualizing the loss landscape of neural nets,” *CoRR*, vol. abs/1712.09913, 2017.

[6] M. Soleymani Baghshah, “Machine learning.” Lecture slides.

Any Questions?