Tuning your Neural Network

Introduction to Deep Learning

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Agenda

1. Weight initialization
   a. Exploding gradient descent
   b. Vanishing gradient descent
2. Regularization -
   a. Batch Normalization
   b. Drop out
3. Choosing the right architecture
4. Hyperparamater optimisation

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Neural Network Constructed

Now that we have seen all the building blocks that define a neural networks, we need to understand:

- Weight initialization
- Regularization

… and finally, to build a well-functioning network

- How to baby-sit the learning process

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

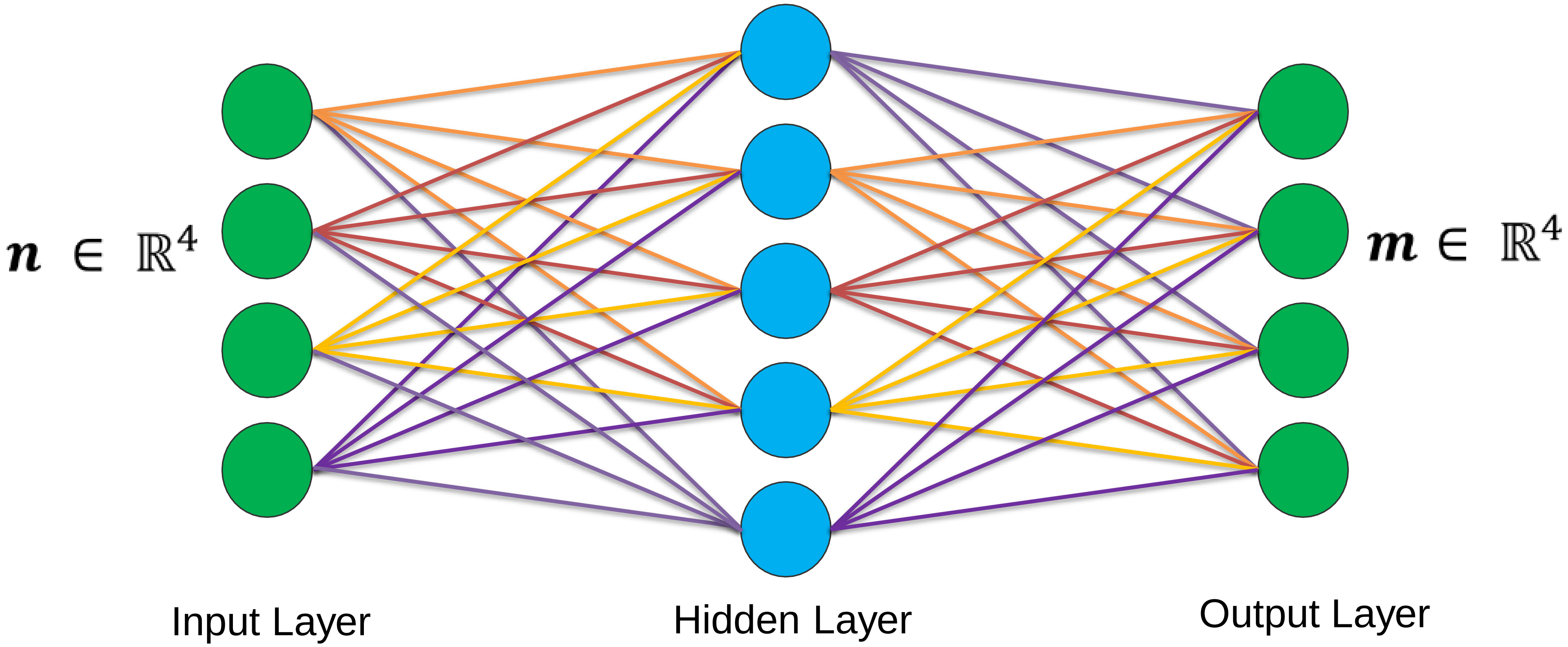
Weight Initialization

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

What happens when W=0 init is used?

Total Parameters = 49



Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Different Cases

1. Initialize all weights with 0 - Your network will not learn as all the weights are same.
2. Initialize with random numbers - Works okay for small networks (similar to our two layer MNIST classifier), but it may lead to distributions of the activations that are not homogeneous across the layers of network.

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

1. Initializing all weights to 0

- This makes your model equivalent to a linear model.
- When you set all weight to 0, the derivative with respect to loss function is the same for every w in every layer, thus, all the weights have the same values in the subsequent iteration.
- This makes the hidden units symmetric and continues for all the n iterations you run.
- Thus setting weights to zero makes your network no better than a linear model.

Refer - https://medium.com/usf-msds/deep-learning-best-practices-1-weight-initialization-14e5c0295b94

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

2. Initializing weights randomly

Initializing weights randomly, following standard normal distribution (np.random.randn(n, n-1) in Python) while working with a (deep) network can potentially lead to 2 issues — vanishing gradients or exploding gradients.

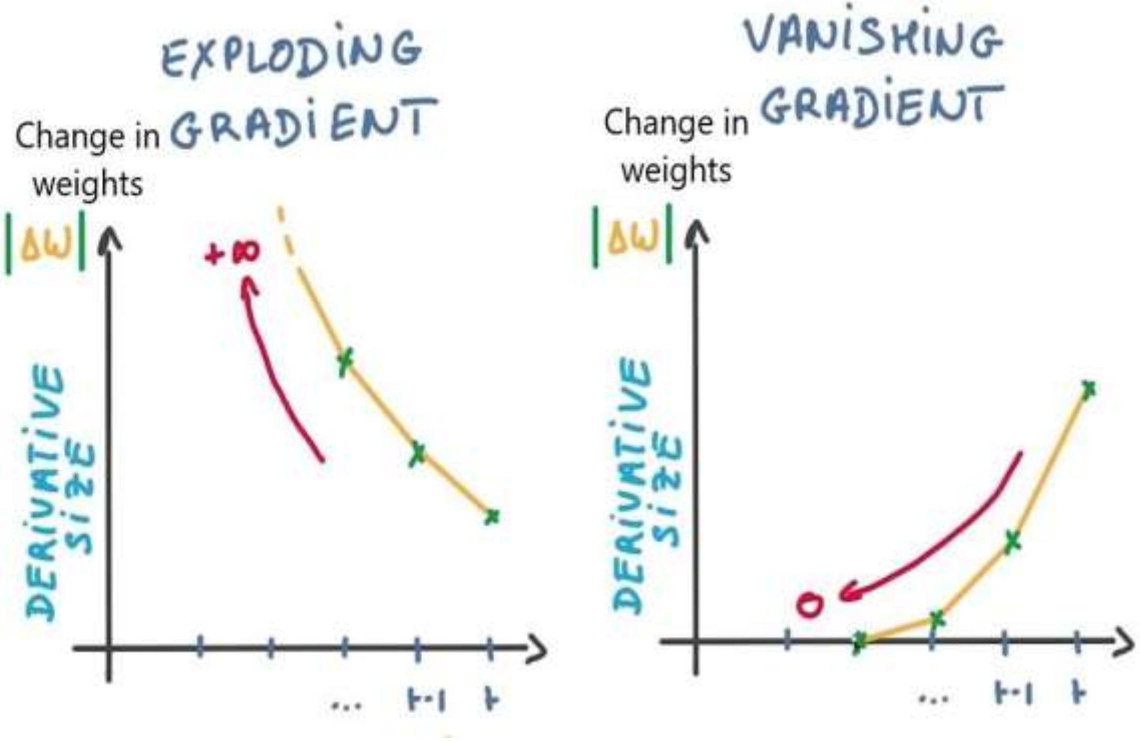
Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Vanishing Gradient Descent

a) Vanishing gradients —In case of deep networks, for any activation function, *abs(dW)* will get smaller and smaller as we go backwards with every layer during back propagation. The earlier layers are the slowest to train in such a case.

The weight update is minor and results in slower convergence. This makes the optimization of the loss function slow. In the worst case, this may completely stop the neural network from training further.



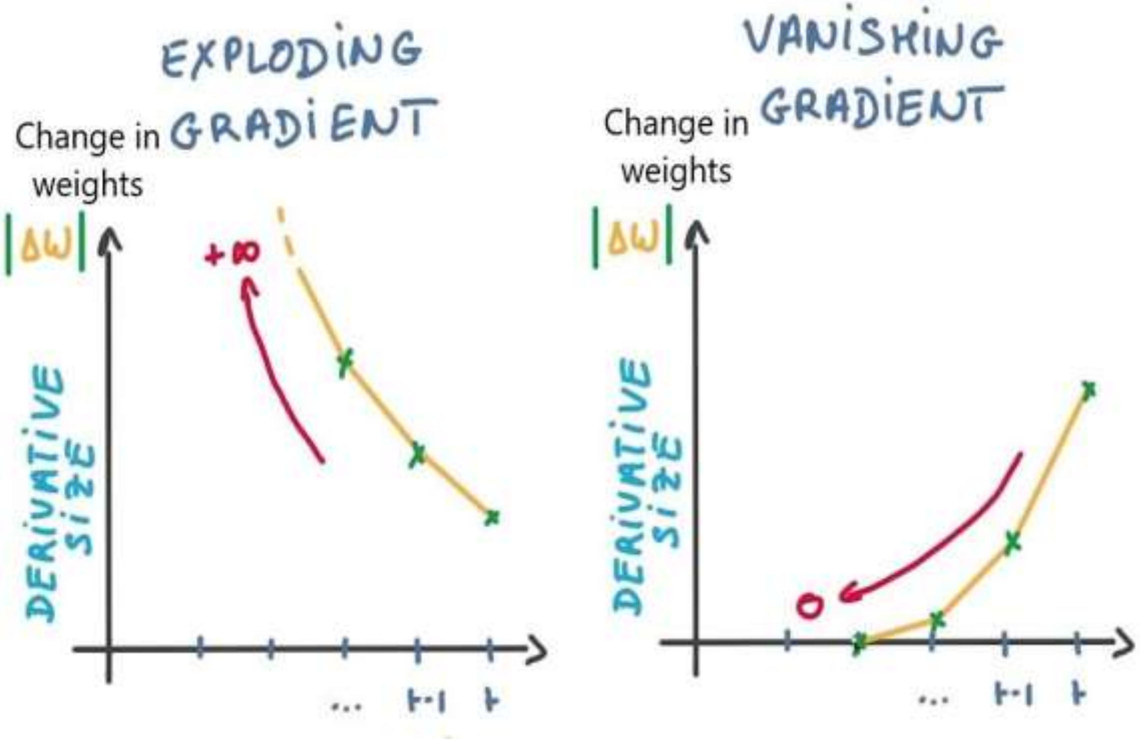
Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited
This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Exploding Gradient Descent

b) Exploding gradients —This is the exact opposite of vanishing gradients. Consider you have non-negative and large weights and small activations A (as can be the case for sigmoid(z)).

This may result in oscillating around the minima or even overshooting the optimum again and again and the model will never learn!

Another impact of exploding gradients is that huge values of the gradients may cause number overflow resulting in incorrect computations or introductions of NaN’s. This might also lead to the loss taking the value NaN.



Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Best Practices

1. Use RELU/ leaky RELU as the activation function, as it is relatively robust to the vanishing/exploding gradient issue (especially for networks that are not too deep).

2. Xavier Initialization - we want to initialize the weights with random values that are not “too small” and not “too large.”

$$\sqrt{\frac{2}{Size\ of\ Previous\ Layer}}$$

$$\sqrt{\frac{2}{Size\ of\ Previous\ Layer + Size\ of\ Current\ Layer}}$$

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Proper initialization is an active area of research…

Understanding the difficulty of training deep feedforward neural networks
by Glorot and Bengio, 2010

Exact solutions to the nonlinear dynamics of learning in deep linear neural networks
by Saxe et al, 2013

Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification
by He et al., 2015

Data-dependent Initializations of Convolutional Neural Networks
by Krähenbühl et al., 2015

All you need is a good init
by Mishkin and Matas, 2015

Centered Weight Normalization in Accelerating Training of Deep Neural Networks
by Huang et al., 2017

Adjusting for Dropout Variance in Batch Normalization and Weight Initialization:
by Hendrycks et al., 2017
…

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Neural Network Constructed

Now that we have seen all the building blocks that define a neural networks, we need to understand:

- Data preprocessing
- Data augmentation
- Weight initialization
- Regularization

… and finally, to build a well-functioning network

- How to baby-sit the learning process

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Regularization

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Batch Normalization

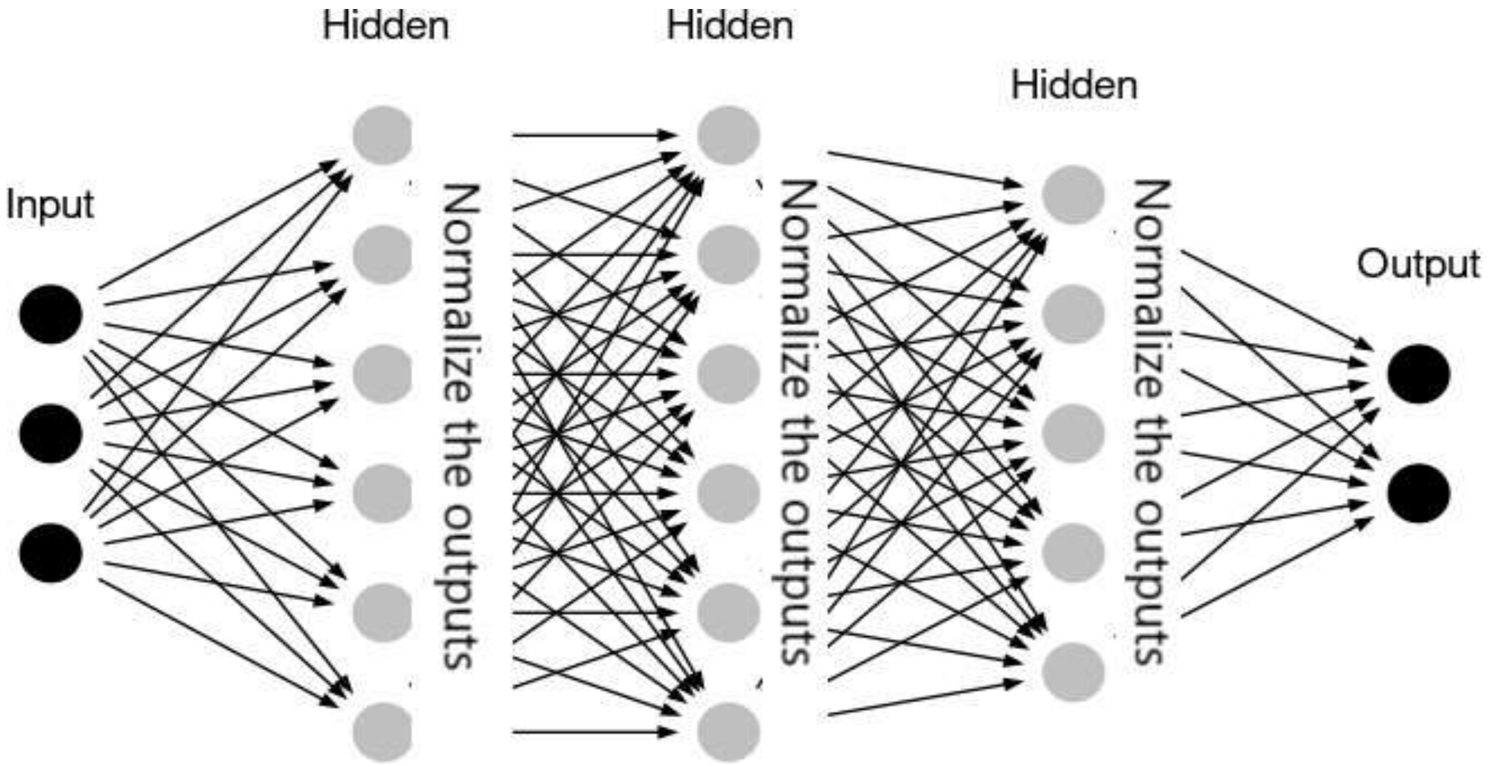
Batch normalisation is a technique for improving the performance and stability of neural networks

The idea is to normalise the inputs of each layer in such a way that they have a mean output activation of zero and standard deviation of one. This is analogous to how the inputs to networks are standardised.

Source - https://medium.com/deeper-learning/glossary-of-deep-learning-batch-normalisation-8266dcd2fa82

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited



Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Benefits of Batch Normalization

1. **Network train faster**
2. **Provides some regularisation**

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Batch Normalization

Due to this normalization “layers” between each fully connected layers, the range of input distribution of each layer stays the same, no matter the changes in the previous layer. Given x inputs from k-th neuron.

Consider a batch of activations at some layer. For making each feature dimension unit gaussian, use:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Where,
X^k = activation of layer k
$E[x^k]$= Mean

$\sqrt{\mathrm{Var}[x^{(k)}]}$ = standard deviation

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Scale and Shift

There are usually two types in which Batch Normalization can be applied:

1. Before activation function (non-linearity)
2. After non-linearity

For sigmoid and tanh activation, normalized region is more of linear than nonlinear.

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

A few issues -

For sigmoid and tanh activation, normalized region is more of linear than nonlinear.

For relu activation, half of the inputs are zeroed out.

So, some transformation has to be done to move the distribution away from 0. A **scaling factor γ** and **shifting factor β** are used to do this.

$$y^{(k)} = \gamma^{(k)} \widehat{x}^{(k)} + \beta^{(k)}$$

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Final Definition of Batch Normalization

Normalize*:*

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

And then allow the network to squash the range if it wants to:

$$y^{(k)} = \gamma^{(k)} \widehat{x}^{(k)} + \beta^{(k)}$$

Where

$\gamma^{(k)}$ An$\beta^{(k)}$ are learnable parameters

$$\gamma^{(k)} = \sqrt{\mathrm{Var}[x^{(k)}]}$$

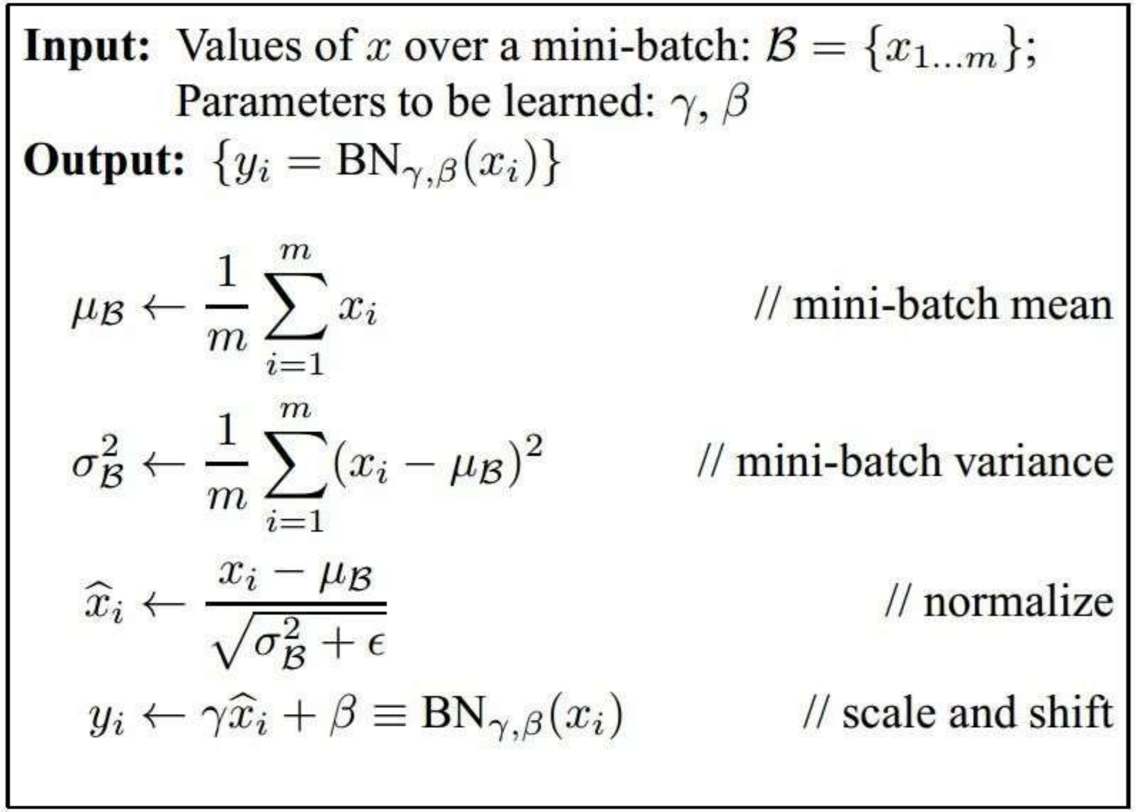
$$\beta^{(k)} = \mathrm{E}[x^{(k)}]$$

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

Batch Normalization - Mini batches

- Improves gradient flow through the network
- Allows higher learning rates
- Reduces the strong dependence on initialization
- Acts as a form of regularization in a funny way, and slightly reduces the need for dropout, maybe



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

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

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

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

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

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

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

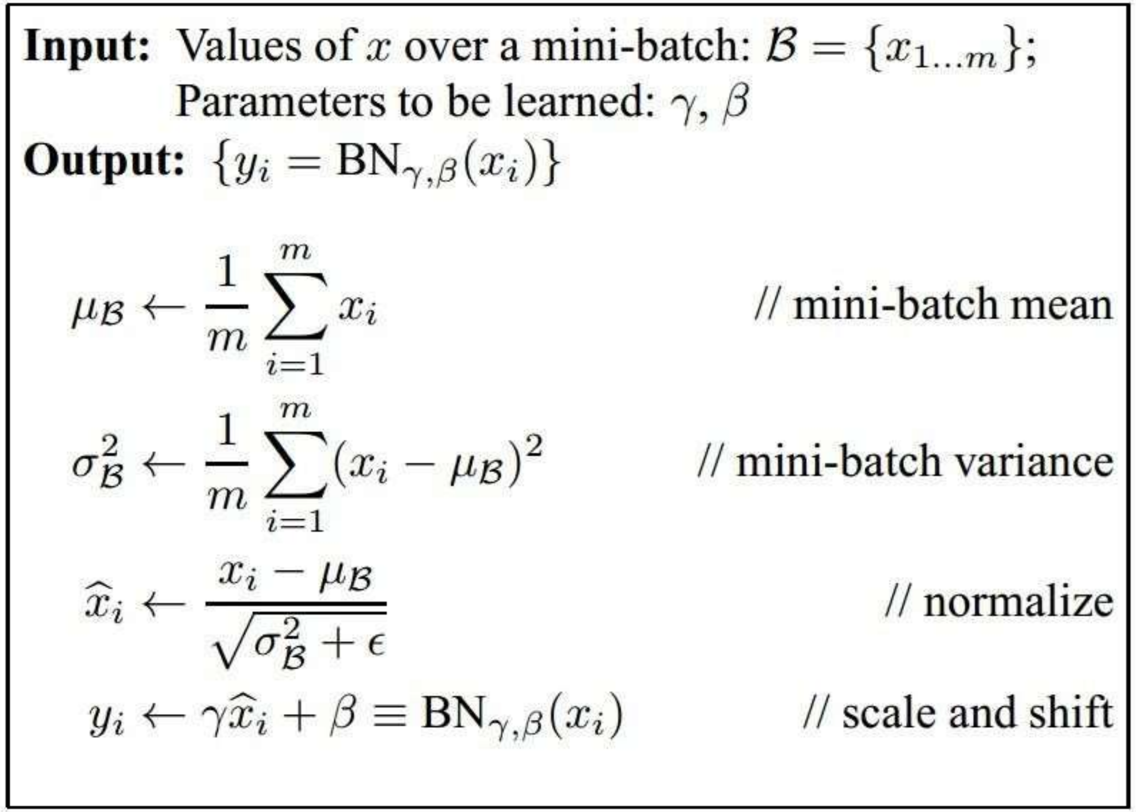
This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Batch Normalization - Mini batches

Note: During the test time BatchNorm layer functions differently:

- Mean or Std are not calculated based on the batch. Instead, during training a single fixed empirical mean if activation functions are used.

(e.g. can be estimated during training with running averages)



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

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

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

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

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

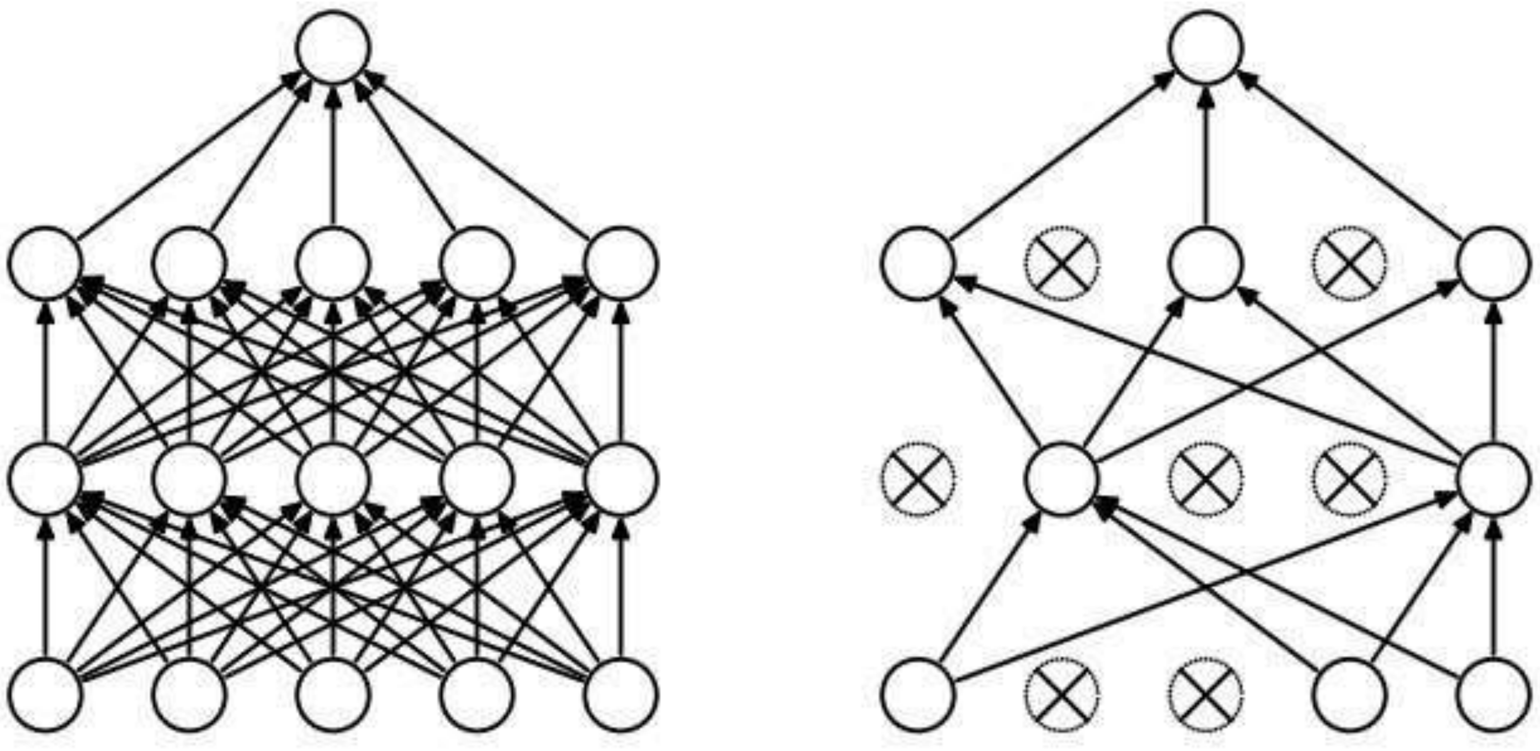
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Regularization: **Dropout**

“During training, randomly set some neurons to zero in the forward pass”



(a) Standard Neural Net

(b) After applying dropout.

Sourced from: Srivastava et al., 2014.

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

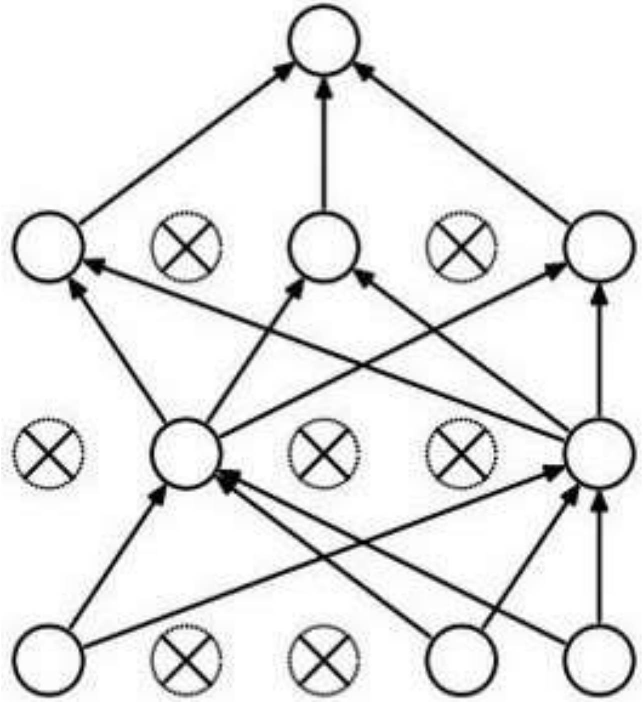
Why do we need Dropout?

The answer to these questions is "to prevent over-fitting".

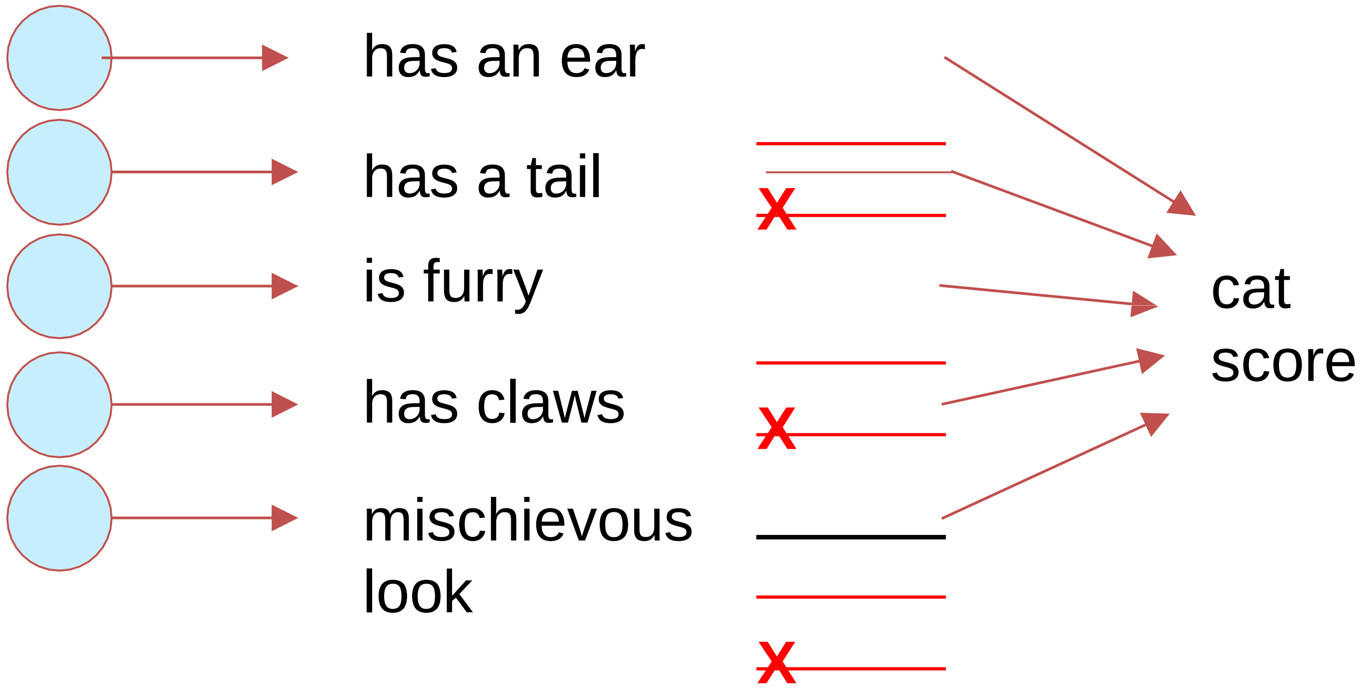
A fully connected layer occupies most of the parameters, and hence, neurons develop co-dependency amongst each other during training which curbs the individual power of each neuron leading to over-fitting of training data.

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

How could this possibly be a good idea?



Forces the network to have a redundant representation.



has an ear

has a tail

is furry

has claws

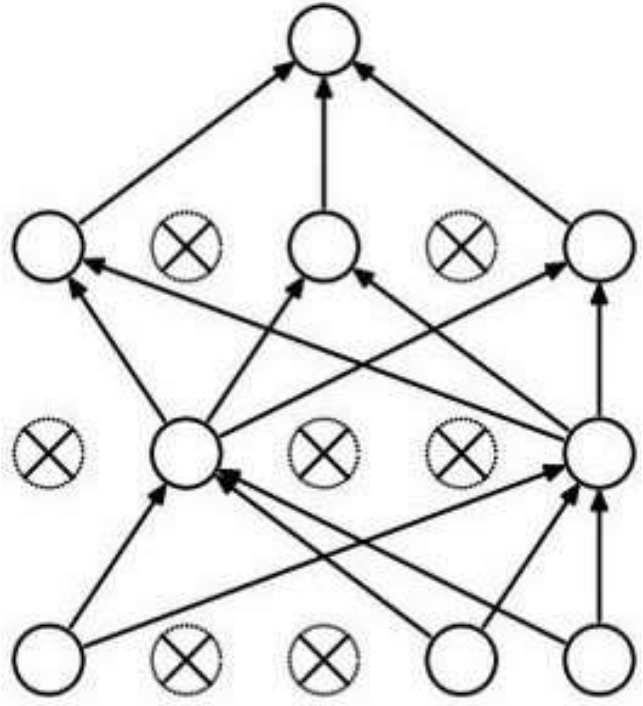
mischievous look

cat score

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

How could this possibly be a good idea?



Another interpretation:

Dropout is an approach to regularization in neural networks which helps reducing interdependent learning amongst the neurons.

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

At test time…

At test time all neurons are always **ON**

We must scale the activations so that for each neuron:

output at test time = expected output at training time

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Neural Network Constructed

Now that we have seen all the building blocks that define a neural networks, we need to understand:

- Data preprocessing
- Data augmentation
- Weight initialization
- Regularization

… and finally, to build a well-functioning network

- How to baby-sit the learning process

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Babysitting the Learning Process

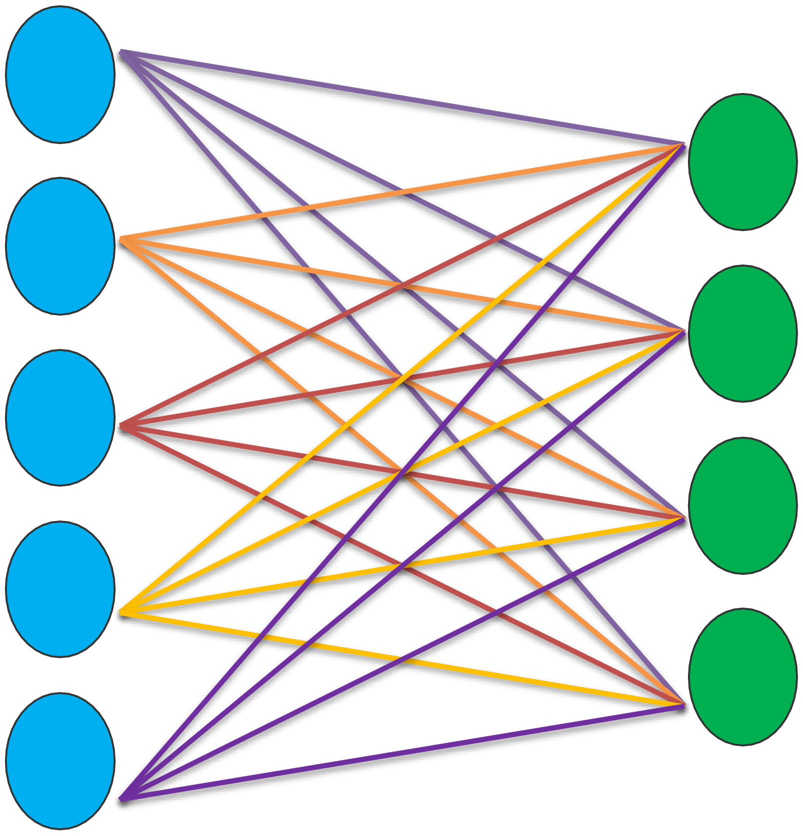
Let’s check in the Notebook

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Choose the architecture:

Say we start with single layer network:



input layer

MNIST images, **784** numbers

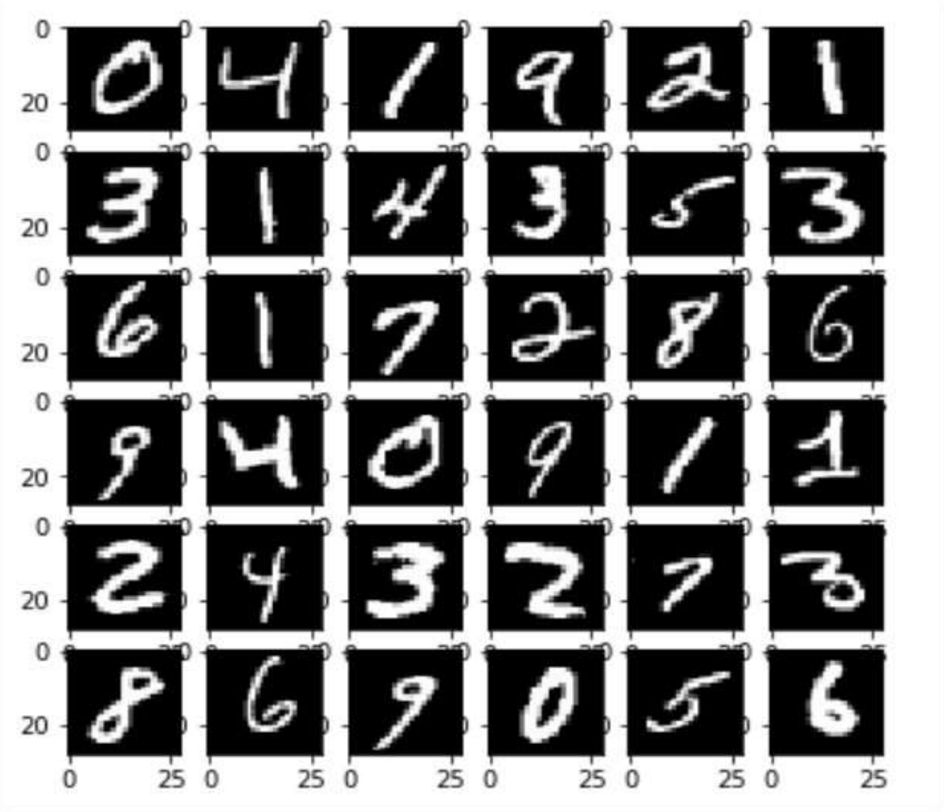
output layer

10 output neurons, one per class

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

1. Data Loading: Let us load the training and the test data and check the size of the tensors. Let us also display the first few images from the training set.



Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Hyperparameter Optimization

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Hyperparameters to play with

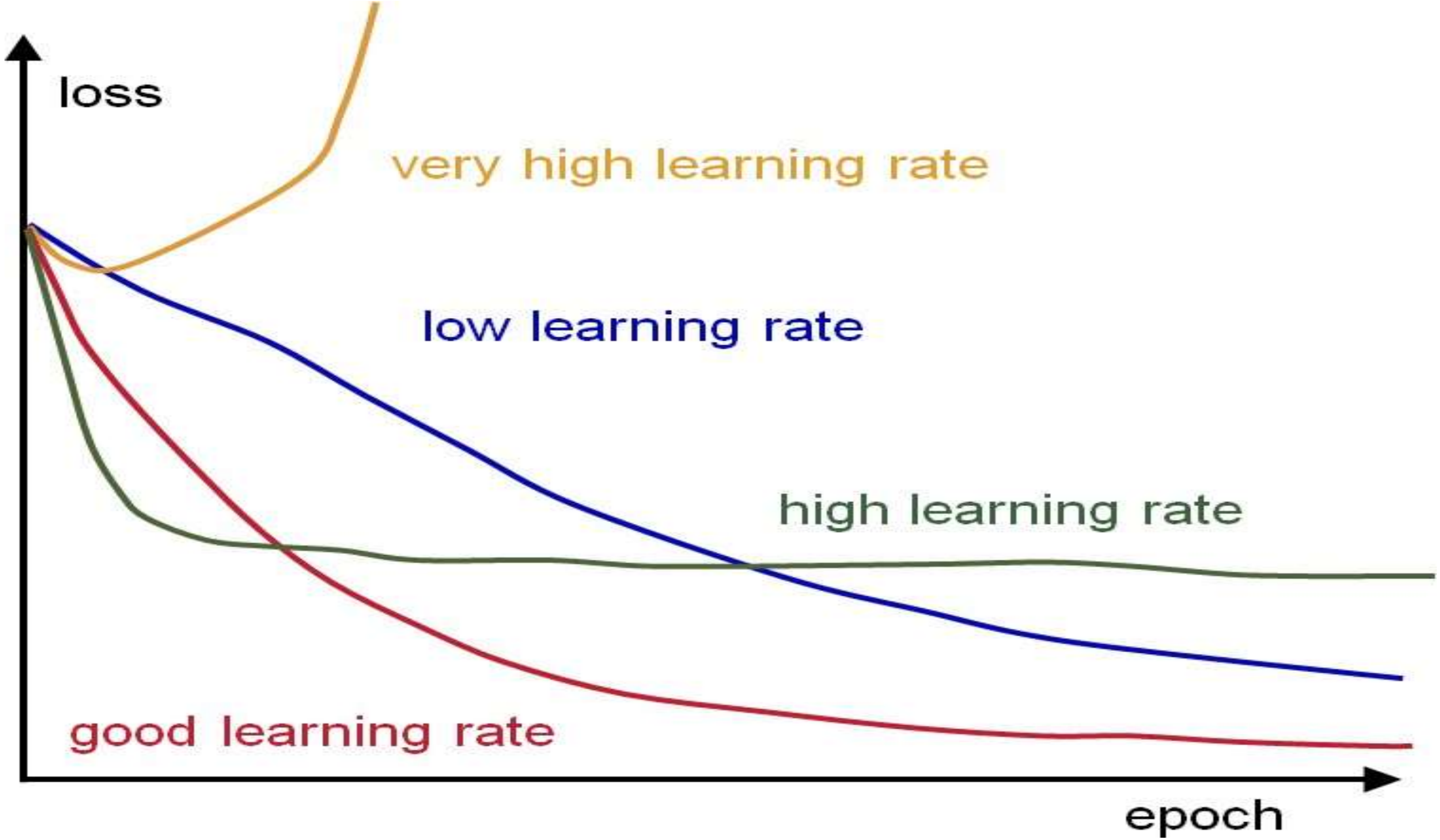
- network architecture
- learning rate, its multiplier schedule
- regularization (L2/Dropout strength)

neural networks practitioner
music = loss function

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

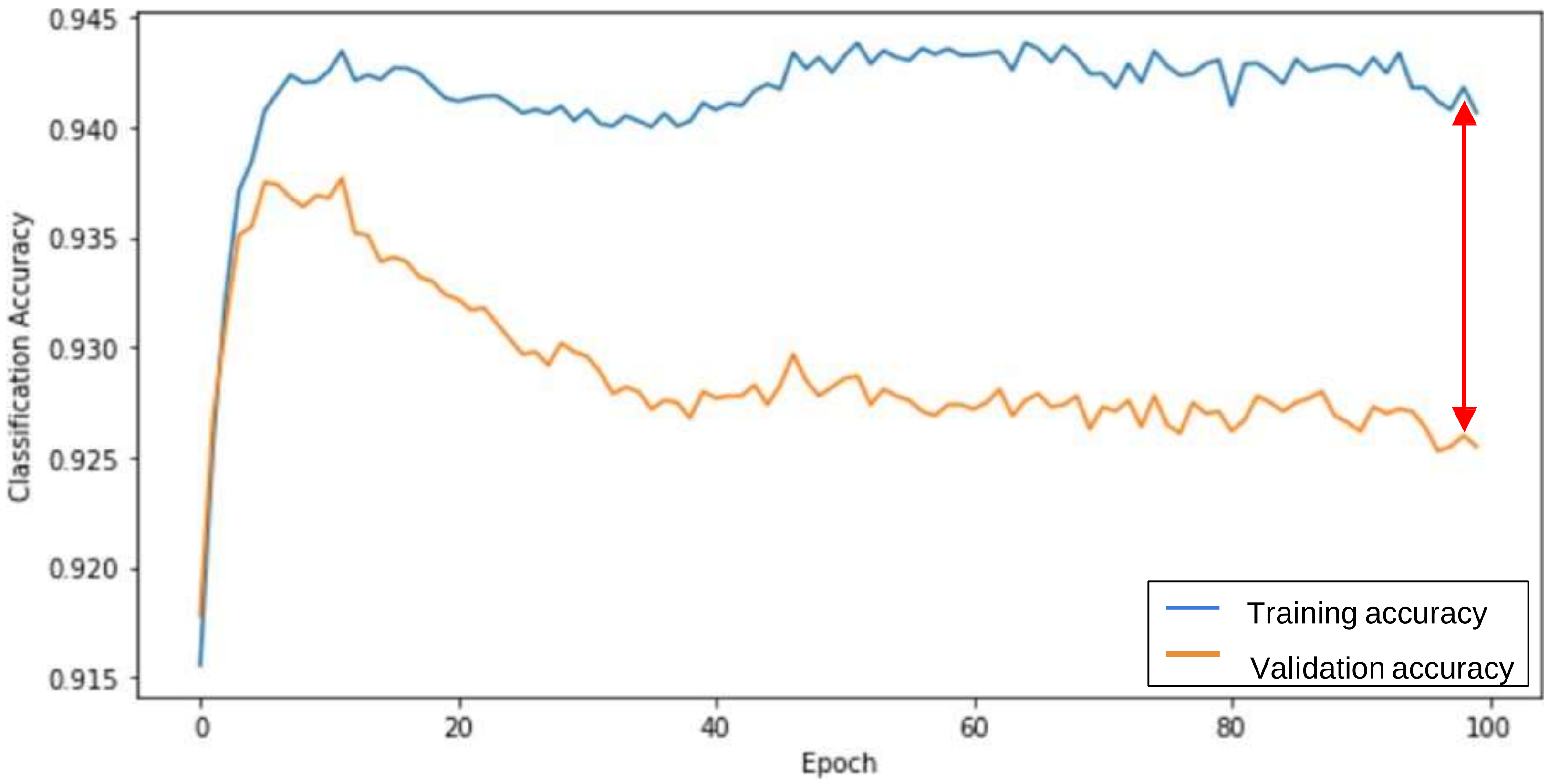
This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Monitor and visualize the loss curve



This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Monitor and visualize the accuracy:



big gap = overfitting
=> increase regularization strength?

no gap - low training and validation accuracy
=> increase model capacity?

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.

Thank You

Proprietary content. ©Great Learning. All Rights Reserved. Unauthorized use or distribution prohibited

This file is meant for personal use by rg.ravigupta91@gmail.com only.
Sharing or publishing the contents in part or full is liable for legal action.