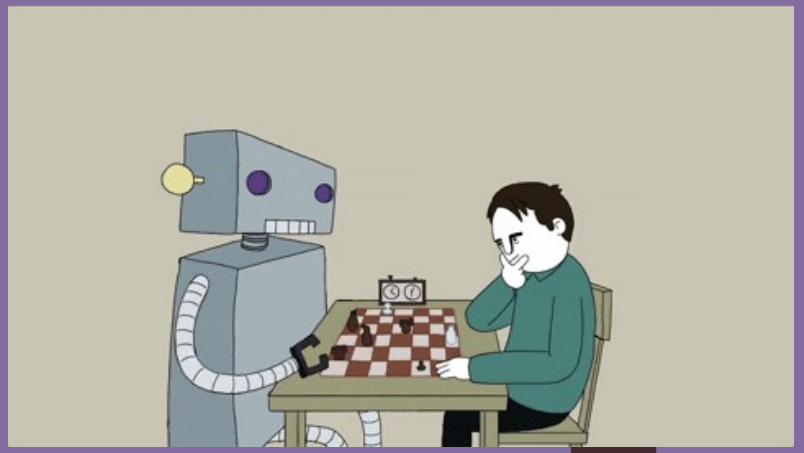
CSE4006
DEEP LEARNING



Dr K G Suma
Associate Professor
School of Computer Science and Engineering

Module No. 2
Practical Deep Networks
8 Hours

- Multilayer Perceptron
- Gradient based Learning
- Backpropagation Algorithm
- Regularization for Deep Learning
- Optimization for training deep models

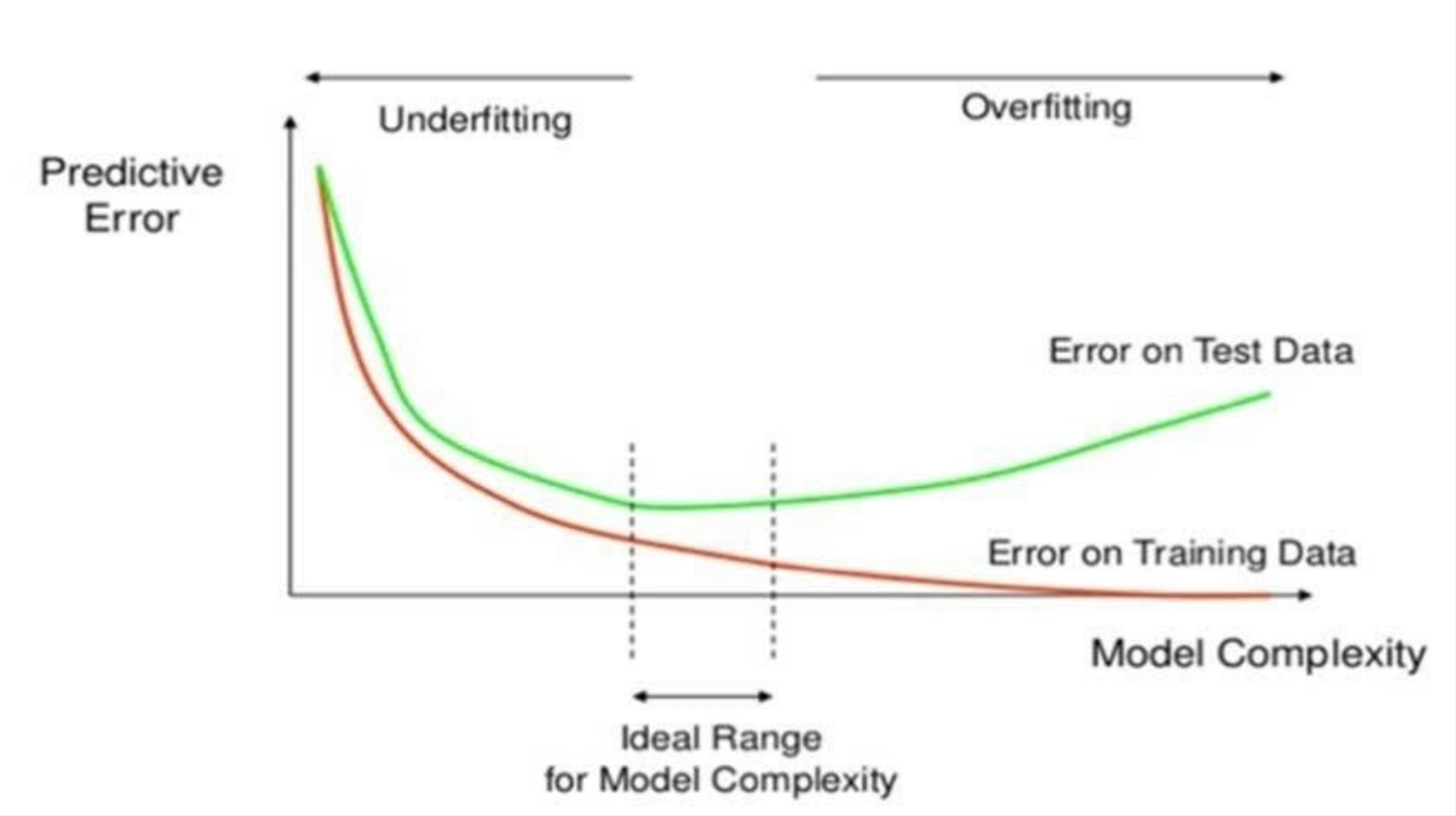
Introduction

- When a Deep learning model is provided with training samples along with corresponding labels, the model will start to recognize patterns in the data and update the model parameters accordingly. The process is known as Training.
- These parameters or weights are then used to predict the labels or outputs on another set of unseen samples that the model has not been trained on, i.e., the Testing dataset. This process is called Inference.

Correct fit vs. Overfit models

- If the model is able to perform well on the testing dataset, the model can be said to have Generalized well, i.e., correctly understood the patterns provided in the training dataset. This type of model is called a Correct fit model.
- However, if the model performs really well on the training data and doesn't perform well on the testing data, it can be concluded that the model has Memorized the patterns of training data but is not able to generalize well on unseen data. This model is called an Overfit model.

- Overfitting is a phenomenon where the machine learning model learns patterns and performs well on data that it has been trained on and does not perform well on unseen data.



- The graph shows that as the model is trained for a longer duration, the training error lessens. However, the testing error starts increasing after a specific point. This indicates that the model has started to overfit.

Methods used to handle overfitting

Overfitting is caused when the training accuracy/metric is relatively higher than validation accuracy/metric. It can be handled through the following techniques:

1. Training on more training data to better identify the patterns.
2. Data augmentation for better model generalization.
3. Early stopping, i.e., stop the model training when the validation metrics start decreasing or loss starts increasing.
4. Regularization techniques.

In these techniques, data augmentation and more training data don't change the model architecture but try to improve the performance by altering the input data. Early stopping is used to stop the model training at an appropriate time - before the model overfits, rather than addressing the issue of overfitting directly. However, regularization is a more robust technique that can be used to avoid overfitting.

Regularization for Deep Learning

- Regularization, which helps prevent overfitting and makes our models work better with new data. While developing deep learning models **we may encounter a situation where model is overfitted. To avoid such issues we use regularization techniques to improve model accuracy.**

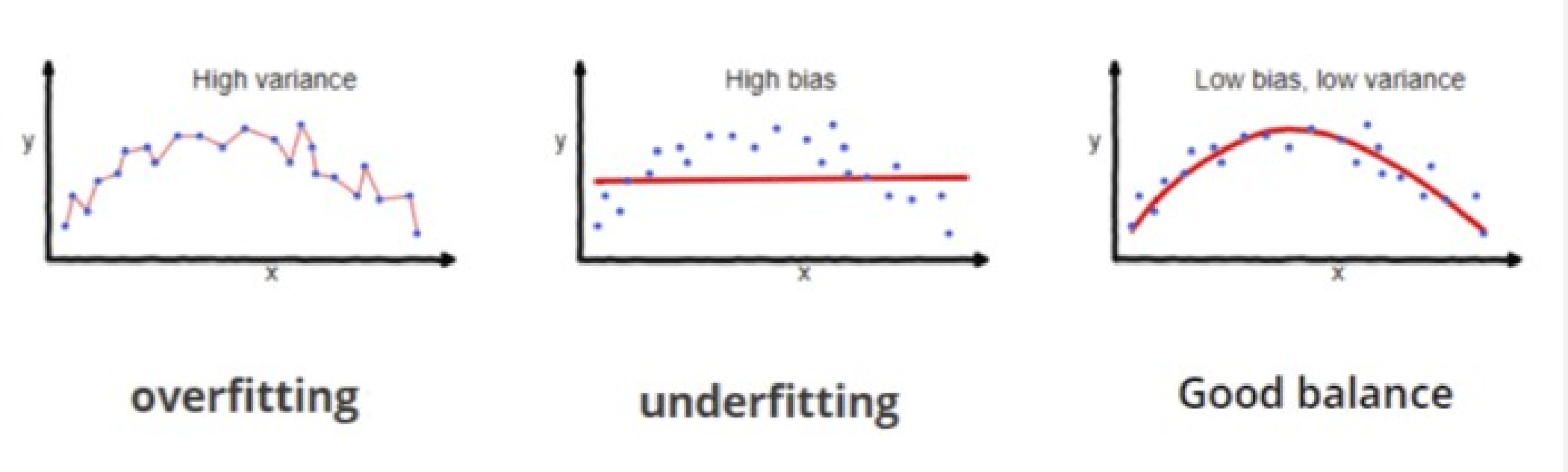
How Does Regularization helps?

- *Regularization is a technique used in machine learning to prevent overfitting. Overfitting happens when a model learns the training data too well, including the noise and outliers, which causes it to perform poorly on new data. In simple terms, regularization adds a penalty to the model for being too complex, encouraging it to stay simpler and more general. This way, it's less likely to make extreme predictions based on the noise in the data.*

Regularization for Deep Learning

The commonly used Regularization techniques are :

1. **Lasso Regularization** – (L1 Regularization)
2. **Ridge Regularization** – (L2 Regularization/ also referred as Weight decay)
3. **Elastic Net Regularization** – (L1 and L2 Regularization)



The image illustrates three scenarios in model performance:

1. **Overfitting –** The model is too complex, capturing noise and outliers leading to poor generalization.

2. **Underfitting –** The model is too simple failing to capture the underlying data patterns.

3. **Optimal Fit –** A balanced model that generalizes well achieving low bias and low variance and it can be achieved by using regularization techniques.

1. Lasso Regression

- A regression model which uses the **L1 Regularization** technique is called **LASSO (Least Absolute Shrinkage and Selection Operator)** regression.
- **Lasso Regression** adds the *"absolute value of magnitude"* of the coefficient as a penalty term to the loss function(L).
- Lasso regression also helps us achieve feature selection by penalizing the weights to approximately equal to zero if that feature does not serve any purpose in the model.

$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{i=1}^{m} |w_i|$$

where,

- **m** – Number of Features
- **n** – Number of Examples
- **y_i** – Actual Target Value
- **y_i(hat)** – Predicted Target Value

2. Ridge Regression

- A regression model that uses the **L2 regularization** technique is called **Ridge regression**.
- **Ridge regression** adds the *"squared magnitude"* of the coefficient as a penalty term to the loss function(L).

$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{i=1}^{m} w_i^2$$

Where,

- n = Number of examples (data points)
- m = Number of features (predictor variables)
- y_i = Actual target value for the iii-th example
- $\hat{y}_i$ = Predicted target value for the i-th example
- w_i = Coefficients of the features
- λ= Regularization parameter (controls the strength of regularization)

3. Elastic Net Regression

- Elastic Net Regression is a combination of both **L1 as well as L2 regularization.**
- That implies that we add the absolute norm of the weights as well as the squared measure of the weights. With the help of an extra hyperparameter that controls the ratio of the L1 and L2 regu

$$\text{Cost} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\left((1-\alpha)\sum_{i=1}^{m}|w_i| + \alpha\sum_{i=1}^{m}w_i^2\right)$$

Where,

- n = Number of examples (data points)
- m = Number of features (predictor variables)
- y_i = Actual target value for the iii-th example
- $\hat{y}_i$ = Predicted target value for the iii-th example
- wiw_iwi = Coefficients of the features
- λ = Regularization parameter that controls the strength of regularization
- α = Mixing parameter (where 0≤α≤10 \leq \alpha \leq 10≤α≤1):
  - α = 1 corresponds to Lasso (L1) regularization
  - α = 0 corresponds to Ridge (L2) regularization
  - Values between 0 and 1 provide a balance of both L1 and L2 regularization

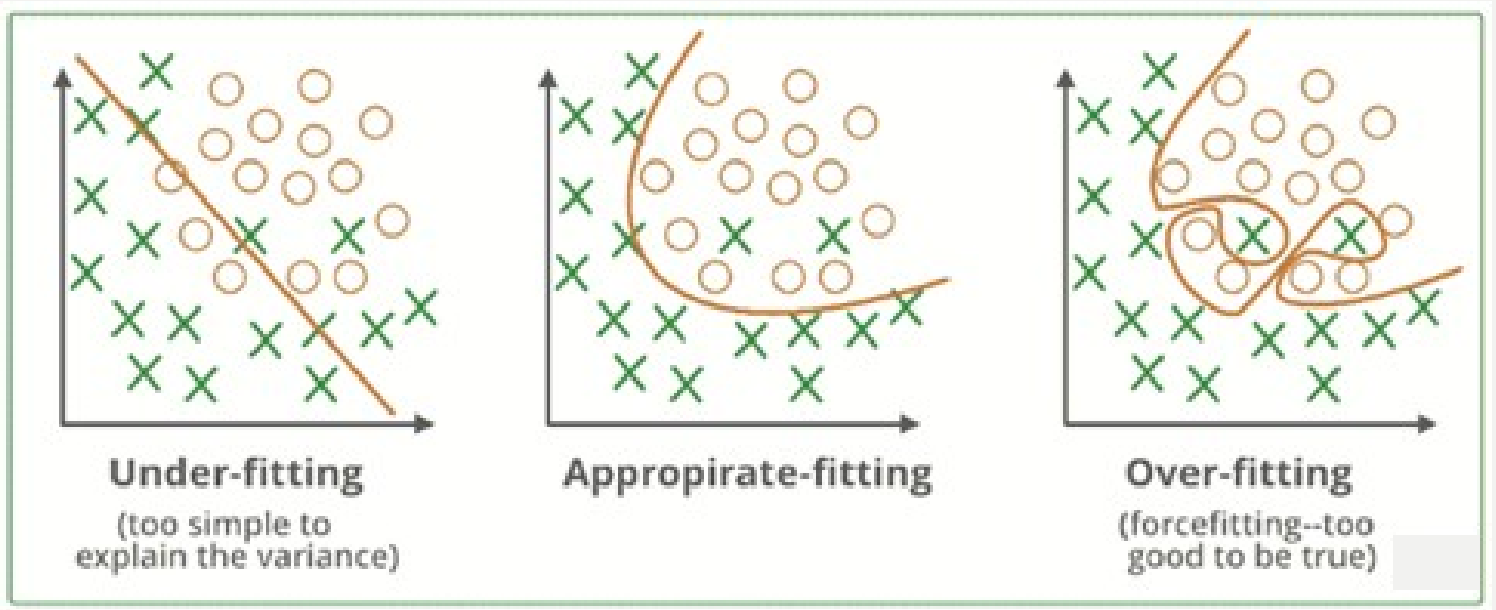
Role Of Regularization

The role of regularization is to help a machine learning model learn better by **avoiding overfitting**. It ensures the model doesn’t become too complicated and only focuses on the most important patterns in the data.

- **Complexity Control**: Regularization reduces model complexity preventing overfitting and enhancing generalization to new data.
- **Balancing Bias and Variance**: Regularization helps manage the trade-off between model bias (underfitting) and variance (overfitting) leading to better performance.
- **Feature Selection**: Methods like L1 regularization (Lasso) encourage sparse solutions automatically selecting important features while excluding irrelevant ones.
- **Handling Multicollinearity**: Regularization stabilizes models by reducing sensitivity to small changes when features are highly correlated.
- **Generalization**: Regularized models focus on underlying patterns in the data ensuring better generalization rather than memorization.

What are Overfitting and Underfitting?

- Overfitting and underfitting are terms used to describe the performance of machine learning models in relation to their ability to generalize from the training data to unseen data.



What are Overfitting and Underfitting?

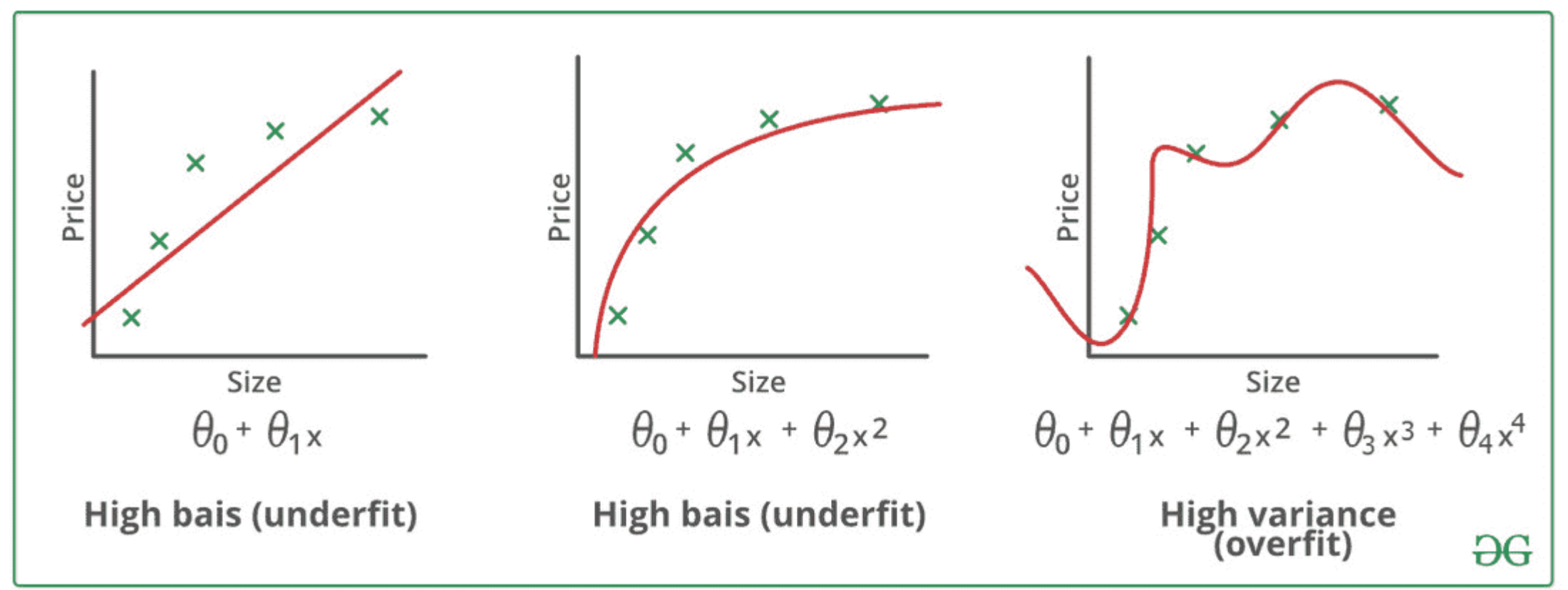
- **Overfitting** is a phenomenon that occurs when a Machine Learning model is constrained to training set and not able to perform well on unseen data. That is when our model learns the noise in the training data as well. This is the case when our model memorizes the training data instead of learning the patterns in it.
- Imagine you study only last week's weather to predict tomorrow's. Your model might predict tomorrow's weather based on irrelevant details, like a one-time rainstorm, which won't help for future predictions

- **Underfitting** on the other hand is the case when our model is not able to learn even the basic patterns available in the dataset. In the case of underfitting model is unable to perform well even on the training data hence we cannot expect it to perform well on the validation data. This is the case when we are supposed to increase the complexity of the model or add more features to the feature set.
- Now, if you only use the average temperature of the year to predict tomorrow's weather, your model is too simple and misses important patterns, like seasonal changes, leading to poor predictions

What are Bias and Variance?

- **Bias** refers to the errors which occur when we try to fit a statistical model on real-world data which does not fit perfectly well on some mathematical model. If we use a way too simplistic a model to fit the data then we are more probably face the situation of **High Bias** (underfitting) refers to the case when the model is unable to learn the patterns in the data at hand and perform poorly.

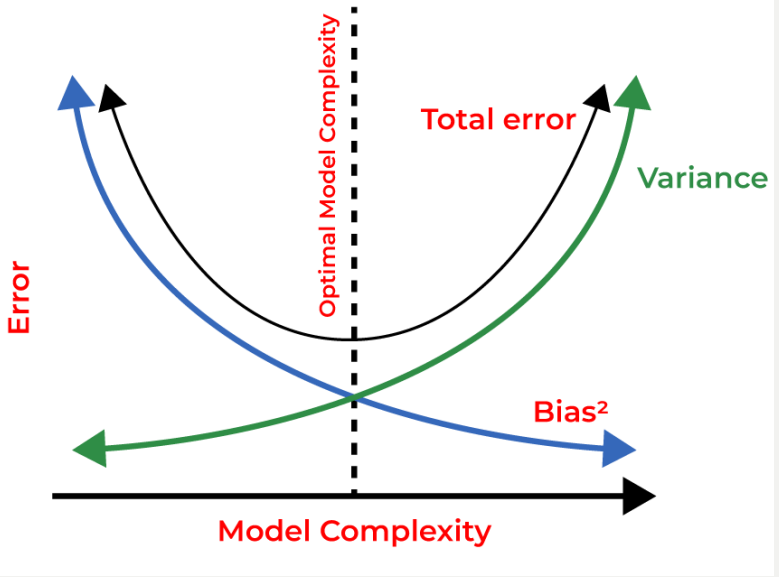
- **Variance** implies the error value that occurs when we try to make predictions by using data that is not previously seen by the model. There is a situation known as **high variance** (overfitting) that occurs when the model learns noise that is present in the data.



Finding a proper balance between the two is also known as the [Bias-Variance Tradeoff] can help us make accurate model.

Bias Variance tradeoff

- The bias-variance tradeoff is a fundamental concept in machine learning. It refers to the balance between bias and variance, which affect predictive model performance. Finding the right tradeoff is crucial for creating models that generalize well to new data.
- The bias-variance tradeoff demonstrates the inverse relationship between bias and variance. When one decreases, the other tends to increase, and vice versa.
- Finding the right balance is crucial. An overly simple model with high bias won't capture the underlying patterns, while an overly complex model with high variance will fit the noise in the data.



Benefits of Regularization

- **Prevents Overfitting:** Regularization helps models focus on underlying patterns instead of memorizing noise in the training data.
- **Improves Interpretability:** L1 (Lasso) regularization simplifies models by reducing less important feature coefficients to zero.
- **Enhances Performance:** Prevents excessive weighting of outliers or irrelevant features, improving overall model accuracy.
- **Stabilizes Models:** Reduces sensitivity to minor data changes, ensuring consistency across different data subsets.
- **Prevents Complexity:** Keeps models from becoming too complex, which is crucial for limited or noisy data.

Benefits of Regularization

- **Handles Multicollinearity:** Reduces the magnitudes of correlated coefficients, improving model stability.
- **Allows Fine-Tuning:** Hyperparameters like alpha and lambda control regularization strength, balancing bias and variance.
- **Promotes Consistency:** Ensures reliable performance across different datasets, reducing the risk of large performance shifts.

Regularization techniques like **L1 (Lasso), L2 (Ridge) and Elastic Net** play a important role in improving model performance by reducing overfitting and ensuring better generalization. By controlling complexity, selecting important features and stabilizing models these techniques help making more accurate predictions especially in large datasets.

Conclusion

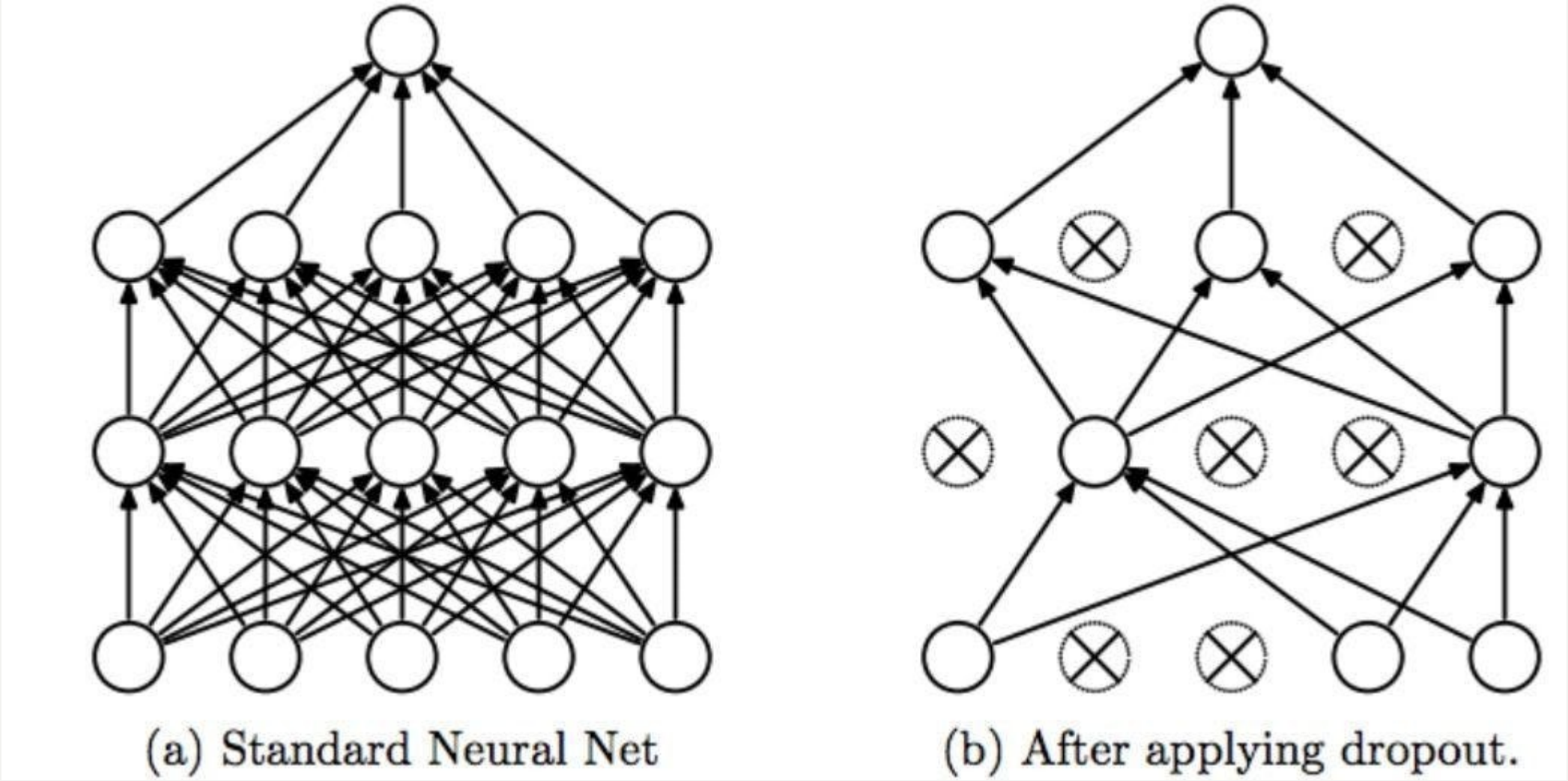
In conclusion, regularization techniques like Lasso and Ridge are essential tools in Deep learning, helping to improve model performance by addressing issues like overfitting and ensuring better generalization. By penalizing large coefficients, these methods make the model more stable, interpretable, and less sensitive to noise in the data. Whether through feature selection in Lasso or controlled complexity in Ridge, regularization enhances the model’s ability to make accurate predictions on unseen data, ultimately leading to more reliable and efficient models.

Other Regularization Techniques for Deep Learning

- Dropout Regularization
- Regularization by Early Stopping
- Batch Normalization

- Dropout Vs Weight decay (L2)

Dropout Regularization



(a) Standard Neural Net

(b) After applying dropout.

Dropout Regularization

- Dropout is a regularization technique which involves randomly ignoring or "dropping out" some layer outputs during training, used in deep neural networks to prevent overfitting.
- Dropout is implemented per-layer in various types of layers like dense fully connected, convolutional, and recurrent layers, excluding the output layer. The dropout probability specifies the chance of dropping outputs, with different probabilities for input and hidden layers that prevents any one neuron from becoming too specialized or overly dependent on the presence of specific features in the training data.

Dropout Regularization - Working

Dropout regularization leverages the concept of dropout during training in deep learning models to specifically address **overfitting,** which occurs when a model performs nicely on schooling statistics however poorly on new, unseen facts.

- During training, dropout **randomly deactivates** a chosen proportion of neurons (and their connections) within a layer. This essentially **temporarily removes** them from the network.

- The deactivated neurons are chosen **at random for each training iteration**. This randomness is crucial for preventing overfitting.

- To account for the deactivated neurons, the outputs of the **remaining active neurons are scaled up** by a factor equal to the probability of keeping a neuron active (e.g., if 50% are dropped, the remaining ones are multiplied by 2).

Dropout Implementation in Deep Learning Models

- Implementing dropout regularization in deep mastering models is a truthful procedure that can extensively enhance the generalization of neural networks.
- Dropout is typically implemented as a **separate layer** inserted after a fully connected layer in the deep learning architecture. The dropout rate (the probability of dropping a neuron) is a **hyperparameter** that needs to be tuned for optimal performance. Start with a dropout charge of 20%, adjusting upwards to 50% based totally at the model's overall performance, with 20% being a great baseline.

Advantages of Dropout Regularization in Deep Learning

- **Prevents Overfitting:** By randomly disabling neurons, the network cannot overly rely on the specific connections between them.
- **Ensemble Effect:** Dropout acts like training an **ensemble of smaller neural networks** with varying structures during each iteration. This ensemble effect improves the model's ability to generalize to unseen data.
- **Enhancing Data Representation**: Dropout methods are used to enhance data representation by introducing noise, generating additional training samples, and improving the effectiveness of the model during training.

Drawbacks of Dropout Regularization and How to Mitigate Them

1. **Longer Training Times**: Dropout increases training duration due to random dropout of units in hidden layers. To address this, consider powerful computing resources or parallelize training where possible.

2. **Optimization Complexity**: Understanding why dropout works is unclear, making optimization challenging. Experiment with dropout rates on a smaller scale before full implementation to fine-tune model performance.

3. **Hyperparameter Tuning**: Dropout adds hyperparameters like dropout chance and learning rate, requiring careful tuning. Use techniques such as grid search or random search to systematically find optimal combinations.

4. **Redundancy with Batch Normalization**: Batch normalization can sometimes replace dropout effects. Evaluate model performance with and without dropout when using batch normalization to determine its necessity.

5. **Model Complexity**: Dropout layers add complexity. Simplify the model architecture where possible, ensuring each dropout layer is justified by performance gains in validation.

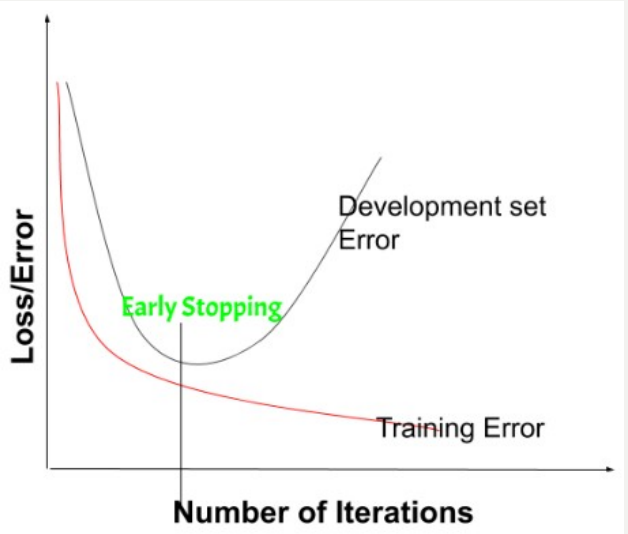
What is Early Stopping?

- In Regularization by Early Stopping, we stop training the model when the performance on the validation set is getting worse- increasing loss decreasing accuracy, or poorer scores of the scoring metric.
- By plotting the error on the training dataset and the validation dataset together, both the errors decrease with a number of iterations until the point where the model starts to overfit. After this point, the training error still decreases but the validation error increases.
- So, even if training is continued after this point, early stopping essentially returns the set of parameters that were used at this point and so is equivalent to stopping training at that point. So, the final parameters returned will enable the model to have low variance and better generalization. The model at the time the training is stopped will have a better generalization performance than the model with the least training error.

What is Early Stopping?

- Early stopping can be thought of as **implicit regularization**, contrary to regularization via weight decay. This method is also efficient since it requires less amount of training data, which is not always available. Due to this fact, early stopping requires lesser time for training compared to other regularization methods. Repeating the early stopping process many times may result in the model overfitting the validation dataset, just as similar as overfitting occurs in the case of training data.

- The number of iterations(i.e. epoch) taken to train the model can be considered a **hyperparameter**. Then the model has to find an optimum value for this hyperparameter (by hyperparameter tuning) for the best performance of the learning model.



Benefits of Early Stopping:

- Helps in reducing overfitting
- It improves generalisation
- It requires less amount of training data
- Takes less time compared to other regularisation models
- It is simple to implement

Limitations of Early Stopping:

- If the model stops too early, there might be risk of underfitting
- It may not be beneficial for all types of models
- If validation set is not chosen properly, it may not lead to the most optimal stopping

Early stopping can be best used to prevent overfitting of the model, and saving resources. It would give best results if taken care of few things like – parameter tuning, preventing the model from overfitting, and ensuring that the model learns enough from the data.

Batch Normalization

- Gradients are used to update weights during training, that can become unstable or vanish entirely, hindering the network's ability to learn effectively.
- Batch Normalization (BN) is a powerful technique that addresses these issues by stabilizing the learning process and accelerating convergence. Batch Normalization(BN) is a popular technique used in deep learning to improve the training of neural networks by normalizing the inputs of each layer.
- Batch Normalization makes the training to be more consistent, and faster, adds better performance, and avoids problems like gradient becoming too small or too large during training and ensures that the network doesn't get stuck or make big mistakes while learning. It is helpful when neural network faces issues like slow training or unstable gradients.

How Batch Normalization works?

1. During each training iteration (epoch), BN takes a mini batch of data and normalizes the activations (outputs) of a hidden layer. This normalization transforms the activations to have a mean of 0 and a standard deviation of 1.

2. While normalization helps with stability, it can also disrupt the network's learned features. To compensate, BN introduces two learnable parameters: gamma and beta. Gamma rescales the normalized activations, and beta shifts them, allowing the network to recover the information present in the original activations.

It ensures that each element or component is in the right proportion before distributing the inputs into the layers and each layer is normalized before being passed to the next layer.

Correct Batch Size:

- Reasonable sized mini-batches must be taken into consideration during training. It performs better with large batch sizes as it computes more accurate batch statistics.

- Leading it to be more stable gradients and faster convergence.

Benefits of Batch Normalization

- **Faster Convergence:** By stabilizing the gradients, BN allows you to use higher learning rates, which can significantly speed up training.

- **Reduced Internal Covariate Shift:** As the network trains, the distribution of activations within a layer can change (internal covariate shift). BN helps mitigate this by normalizing activations before subsequent layers, making the training process less sensitive to these shifts.

- **Initialization Insensitivity:** BN makes the network less reliant on the initial weight values, allowing for more robust training and potentially better performance.

The choice between using batch normalization or not depends on factors such as model architecture, dataset characteristics, and computational resources. The discussed practices for batch normalization must be taken into consider as it reflects its output in the MLP. Thus for better generalization, and faster convergence leads to takeforward the technolgies in deeper networks.

Dropout Vs Weight Decay

- **Description:** Dropout is a regularization technique used in neural networks during training. It involves randomly setting a fraction of input units to zero at each update during training, which helps prevent overfitting.
- **Purpose:** To reduce overfitting by preventing the co-adaptation of neurons and promoting robustness.
- **Implementation:** Dropout is typically implemented by randomly "dropping out" (setting to zero) a fraction (dropout rate) of neurons during each forward and backward pass.
- **Effect on Model:** It introduces a form of ensemble learning, as the network trains on different subsets of neurons in each iteration.

- **Description:** Weight decay, also known as L2 regularization, is a method used to penalize large weights in the model. It involves adding a term to the loss function proportional to the sum of the squared weights.
- **Purpose:** To prevent the model from relying too heavily on a small number of input features and to promote smoother weight distributions.
- **Implementation:** It is implemented by adding a regularization term to the loss function, which is the product of a regularization parameter (lambda) and the sum of squared weights.
- **Effect on Model:** It discourages the model from assigning too much importance to any single input feature, helping to generalize better on unseen data.