Gradient Descent Finds Global Minimum

Subedi, Abhishek
Universität passau
Email: subedi01@ads.uni-passau.de

***Abstract*—Gradient descent is an algorithm to perform optimization of neural network. In a neural network the objective function will have several minimum values and one of them is the global minimum. To ensure trainability and generalizability of a neural network finding global minimum is very important. Some of widely used methods to overcome local minima and achieve global minima are over-parameterization of neural network, Gaussian initialization of weight, defining upper, lower bound of input size and hidden number of nodes per layer etc.**

Algorithms like CNN, RNN, LSTM, DQN are used to solve problems in various fields, autonomous driving is one of them. Autonomous driving make use of these and many other algorithms in all the stages : *Perception*, *Planning* & *Control*, to make the vehicle intelligent and gradient descent is extensively used to optimize these algorithms. From literature research and experiment performed on MNIST datasets with CNN, practical degree of over-parameterization was found to be the best method to find global minimum.

***Index Terms*—gradient descent, global minimum, autonomous vehicles, parametrization**

I. Introduction

Deep Learning technology is a method that consists of multiple continuous processing layers to learn meaningful characteristics of data. These approach has extensively improved the application in speech recognition, visual object recognition, object detection, robotics etc [1]. Autonomous Driving is a domain that have done tremendous progress with the advancement in the area of Deep Learning. Algorithms such as convolutional and recurrent neural networks including deep reinforcement learning helped rapid growth of the field [2].

Deep Learning model is a highly iterative process that consists of layers, inputs, objective function, weight parameters combined in a architecture. For a given architecture, the values of parameters determine how accurately the model performs the task, where loss function determines the performance of model. The target is to minimize the loss, as a result best possible parameters are identified that match predictions with reality [3]. Hence to optimize the model parameter of neural network, we use ***Gradient Descent*** [4].

The optimization problem can be convex or non-convex. Convex optimization function includes single minimum, where local minimum of a function is equal to the global minimum, making them relatively easy to solve. While Non-convex optimization involves several local minimum, among them one of it is the global minimum [5]. As gradient descent could stuck in some local minimum, our objective is to research various approaches in finding global minimum. Modern Deep Learning applications are extremely high dimensional which is an NP-hard problem, this is because unlike convex optimization, we do not possess a set of tools for solving non-convex problems [6].

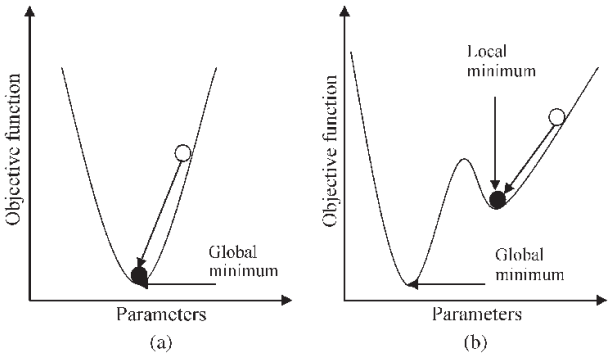

Fig. 1. Local and Global Minimum

The figure:$1(a)$ shows convex and figure:$1(b)$ depicts non-convex objective function with global and local minimum [7].

Various research provides several tools and techniques to overcome the barrier of local mimina and achieve global minima. Generally practiced methods are over-paramterizaton, gaussian weight initialization, set lower and upper bounds of input size etc. Research were performed in diverse network architecture from shallow neural network to convolutional neural network and recurrent neural network.

II. Gradient Descent Based Optimization

Deep feedforward networks, also called feedforward neural network are used to approximate function $f(.)$ that can map input X to a category Y [17]. Let $\{(x_1, y_1), ..., (x_n, y_n)\}$ be a set of n training samples, where $x_n \in \mathbb{R}^{d_x}$ is the given input, $y_n \in \mathbb{R}^{d_y}$ is the target value and θ be the parameter vector. Considering H hidden layers, the output of the neuron is given by [17]

$$f(x, \theta) = W^{(H+1)} x^{(H)} + b^{(H+1)} \in \mathbb{R}^{d_y}$$
$$H \geq 1, W^{(H+1)} \in \mathbb{R}^{d_y * d_H}, b^{(H+1)} \in \mathbb{R}^{d_y}$$

Here $W^{(H+1)}$ and $b^{(H+1)}$ are the weight matrix for the hidden layer and bias term respectively of the output layer.

The output of a layer is applied with an activation function to introduce non-linearity to neural networks. There are many activation functions in Deep Learning like ReLU, Sigmoid, TanH etc. The l-th layer is output given by [17]

$$x^{(l)} = \sigma(W^{(l)}x^{(l-1)} + b^{(l)}), l = 1, 2, ..., H,$$

where $W^{(l)}$ is the weight matrix, $b^{(l)}$ is the bias term and σ is the activation function.

From the iterative process for subsequent layers, the last layer gives the output, i.e. the predicted value $\hat{y}_i$, by the neural network. The target value of the neural network is compared with predicted value and error is calculated, given by [17]

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i, \theta), y_i)$$

where y_i is the target and $l(., y_i)$ represents a loss criterion. There are several different loss functions e.g. cross-entropy, binary cross-entropy, mean-squared error etc. given by [17]

- Mean-Squared Error (MSE)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Cross Entropy

$$CE = -\sum_{c} y_i log(\hat{y}_i)$$

A. Variants of Gradient Decent

There are three types of gradient descent, they are different as per the amount of data applied, to find the gradient of loss function. The update time and the accuracy of network also depends on the amount of data [8].

1) Batch gradient descent: Batch gradient descent (GD), calculates the gradient of the loss function w.r.t. the parameters θ for all training dataset:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta)$$

Gradient descent for all datasets are calculated to perform single update, hence it is slow and face of memory. The parameter is updated in the path of GD with η as learning rate. Batch GD can converge to global minina for convex optimization and to local minima for non-convex optimization [8].

2) Stochastic gradient descent: SGD updates parameter for each and every training data $x^{(i)}$ and label $y^{(i)}$.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$$

SGD is fast as it is performing one update a time and also can fluctuate the loss function. The fluctuation can help the algorithm to jump to a new and better local minima [8].

3) Mini-batch gradient descent: It updates parameter for every mini-batch of n training samples.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)})$$

Mini-batch can direct to stable convergence as it decreases the variance of parameter updates. In practice, mini-batch ranges between 50 to 256 and can vary with applications. This is the algorithm of choice in practice while training neural network [8].

Algorithm 1 [17] shows stochastic gradient descent (SGD) update with mini-batch of m samples.

Algorithm 1 Stochastic gradient descent (SGD) update

Input: Train sample $(x_i, y_i)_{i\in[n]}$, learning rate η, iterations T, initial parameter θ

$k \leftarrow 1$

while *stopping criterion not met* **do**

- Sample a minibatch of m examples from training set $x^{(1)}, ..., x^{(m)}$ with corresponding target $y^{(i)}$.
- Compute gradient estimate: $g \leftarrow \frac{1}{m}\nabla_\theta \sum_i l(f(x_i; \theta), y_i)$.
- Apply update: $\theta \leftarrow \theta - \eta g$.
- $k \leftarrow k + 1$.

end

III. Literature Review

In the work [9], proves SGD find *global minima* for Deep Neural Network assuming the network is over-parameterized and non-degenerate. Over-parameterized in a sense that network width is sufficiently large with *L* layers and *n* number of samples. As various study shows that even for two-layer neural network over-parameterization is widely applicable to avoid bad local minima, the corresponding research is more focused on multi-layer neural networks. Non-degenerate property of data implies every pairs of samples in the datasets are distinct in nature, since two similar data points of different labels cannot be trained to zero error. Denoting δ the minimum distance between two data points, *n* the number of samples, *L*-layer fully connected feedforward neural network and each hidden layer consisting of *m* neurons applying ReLU activation function, the result shows that,

- Until $m \geq poly(n, L, \delta^{-1})$ with random Gaussian initialization of weight, (stochastic) gradient descent finds an ε-error global minimum in ℓ_2 regression using at most $T = poly(n, L, \delta^{-1})log\frac{1}{\varepsilon}$ iterations.
- GD/SGD finds 100% accuracy on multi-label classification with training set in $T = poly(n, L, \delta^{-1})$ iterations.

This research study [10] proves, in polynomial time deep neural network which is over-parameterized can achieves training loss equal to zero with ResNet. It is dependent on a specific anatomy of the Gram matrix lead by the neural network architecture. Hence it depicts Gram matrix is stable in the training procedure and its helps achieve global minimum. It is still unknown how gradient descent achieve training loss equal to zero having random initialization. It is hugely believed over-parameterization is the reason behind it, if the neural network has sufficiently large capacity. In practice, neural network architecture are over-parameterized, eg. Wide ResNet have 100 times parameters than the number of training samples. Another mysterious phenomenon in training

networks is "deeper networks are harder to train" and to solve the problem deep residual network (ResNet) architecture was introduced. Research shows ResNet helps prevent gradient vanishing problem but for neural network with non-linear activation, the merits of using residual connections are not well understood. Considering a neural network with *H* layers, *m* width, working on *n* data points applying least-squares as loss function and soft-plus or sigmoid as activation function, the result shows that,

- Let a fully-connected network having $m = \Omega(poly(n)2^{O(H)})^1$ that is initialized randomly will converge to training loss equal to zero.
- In the second example, considering ResNet, until $m = \Omega(ploy(n, H))$, randomly initialized gradient descent converges training loss to zero at linear rate. In comparison with the previous result, reliance on the number of layers upgrades exponentially for ResNet.Hence, shows the merits of residual architecture.
- At the end, we apply same methodology to convolutinal ResNet. It depicts if $m = poly(n, p, H)$ where p is the number of patches, gradient descent attains zero loss during training.

Research have made diverse hypothesis to have better conceptual understanding of neural network training, such as over-parameterization and Gaussian input assumption. This paper [11], studies the optimization of GD for deep ReLu neural network. This research considers a network with L-hidden layer that is fully-connected with ReLU function for binary classification to achieve global minina. Having $L \geq 1$ for GD and SGD, over-parameterization and random initialization are applied to achieve the goal. Hence the result are summarized as follows:

- GD gains zero training loss, with nodes hidden in every layer is at least $\Omega(poly(n, \phi^{-1}, L))$, in $O(poly(n, \phi^{-1}, L))$ iterations. Here n is train data size, L hidden layers size. Here Gaussian random initialization is applied.
- The above result can be applied to various family of loss functions.
- SGD also gains zero training loss when size of nodes hidden in all the layer is at the minimum $\Omega(poly(n, \phi^{-1}, L))$ with in $O(poly(n, \phi^{-1}, L))$ interations.

Unlike other previous research, this study [12] as well utilize the power of over-parameterization to achieve global minima. However it shows better way to gain global minimum with mild over-parameterization of training network w.r.t size of training data. Conventional optimization algorithms are still unable to explain why GD/SGD find global minina of the objective function in an non-convex optimization. Previous study shows two-layer networks and deep neural networks can be generalized by global convergence. But, there are still space between practice and theory, as these works are based on unrealistic over-parameterization on the width of neural networks. This research provides an improved techniques, i.e. a tight lower bound, that can find convergence faster. The contribution of research focus on global convergence of SGD for deep ReLU neural network and under same setting faster global convergence for GD and SGD with milder condition on width of neural network given by

- When the number of hidden nodes per layer is $\Omega(kn^8L^{12}\phi^{-4})$ with Gaussian random initialization on each layer, gradient descent (GD) can achieve ϵ training loss with in $O(n^2L^2log(\frac{1}{\epsilon})\phi^{-1})$ iterations, where ϕ is the minimum data separation distance, L is the number of hidden layers, n is the number of training examples and k is the output dimension. Comparing the result with [8], this shows over-parameterization condition is milder by a factor of $\Omega(n^{16}\phi^{-4})$ and the iteration complication is better by a factor of $O(n^4\phi^{-1})$.
- Similar convergence for stochastic gradient descent (SGD) shows, when the number of hidden nodes per layer is $\Omega(kn^{17}L^{12}B^{-4}\phi^{-8})$ with Gaussian random initialization on each layer, SGD can achieve ϵ training loss with in $O(n^2L^2log(\frac{1}{\epsilon})\phi^{-1})$ iterations. Comparing the result with [8], this shows over-parameterization condition is better by a factor of $\Omega(n^7B^8)$ and the iteration complication is better by a factor of $O(n^2)$, where B is the minibatch size of SGD.

Previous research were focused on over-parameterization of neural network and some mild or extreme assumptions on data. This paper [13] only requires practicle applicatoin of over-parameterization. The theory defines as the number of training examples increases it is necessary to increase the number of trainable parameters linearly. This permits deep neural networks size to be in harmony with practice. With the success of deep learning, *expressivity* of shallow neural networks to mordern deep architecture have been studied. But, it has not been able to ensure the *trainability* of the neural network. Expressivity indicates that parameters in a network which are earlier have greater influence on its expressive power and states that for a neural network there is an optimal parameter vector to interpolate the given dataset. Encountering an optimal vector has been an NP-hard problem for an neural network. This paper focuses on upper bound and lower bound of the number of parameters so that trainability of the neural network can be ensured. Specifically given by

- $\Omega(n)$ parameters can efficiently make a Deep neural network trainable using gradient descent.
- This hypothesis needs total number of parameters to be in the order of n, similar to practical observations.
- Deep neural network of size $\omega(n)$ are generalizable to unseen test points with a natural dataset, but not with random dataset.

IV. Autonomous Vehicle Perspective

Before autonomous vehicles hit the market, question arise, how safe are these vehicles in terms of accuracy and performance? It is very important for determining how these systems shape the safety, public health and policies for their deployment. One way to determine the safety is put the vehicles in

TABLE I
PROBLEM SPACE AND NN ARCHITECTURE

Problem space	NN architecture	Sensor input
DARPA Challenge	6-layer CNN	Raw camera images
Steering angle, velocity control.	CNN + Fully Connected + LSTM	camera frames, steering wheel angle
Lane keeping, obstacle avoidance	DQN +RNN +CNN	TORCS simulator images
Ego-motion prediction	FCN-LSTM	Large scale video data

real life test environment and improve [14]. Technically, in improving the performance and accuracy of neural network, **gradient descent** plays an important role.

Table I [15], shows various neural network architecture and its application in autonomous vehicles.The research [9] [10] [11] [12], shows as gradient descent finds global minimum, the train-ability of neural network is ensured with high accuracy. Hence, use of gradient descent in finding global minimum is of great importance to autonomous vehicles.

Autonomous driving systems are using algorithms like DQN, CNN, RNN, LSTM, where gradient descent comes into play for improving the network. The Fig. 2, shows the core components and features of an autonomous driving with detail description as follows.

A. Perception

Perception is the capability of a system to gather information from the environment and can be categorized in two types. *Environmental perception* implements Computer vision and Sensor Fusion techniques to perceive the environment. It enables conversion of sensed data from Camera, Radar, Lidar to knowledge. *Localization* is about locating the position of vehicle in between the environment. Scene segmentation, lane line detection, road surface detection, on-road object detection, localization are the basic functionalities of perception [14].

B. Planning

Planning is known to be brain of the autonomous system. The vehicles need to successfully traverse the path from start point to the end point avoiding pedestrians, objects around them according to the trajectories instructed by the autonomous planner [14]. Basically it has three different components: Mission, Behavioural, Motion planning. *Mission Planner* executes higher goals, e.g. selection of roads, pickup or drop off task etc. The *behavioural planner* is responsible for interaction with agents that comes around the vehicle and make local strategy e.g. overtake, change lanes etc. Third is generating trajectory to achieve goals that are set to travel from start to destination, avoiding obstacles, this is handled by *motion planner* [14].

C. Control

The last step is to control the vehicle. This is dependent on the previous components i.e. Perception and Planning. It is based on executing the strategy and functionalities planned by

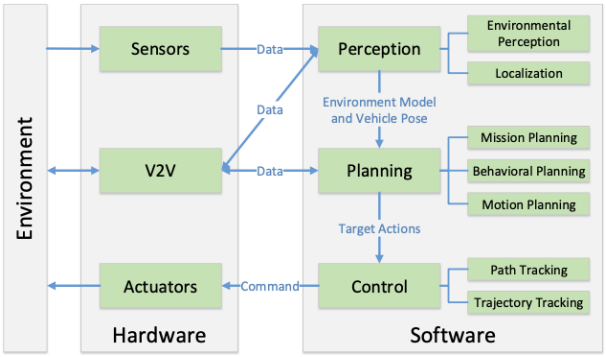

Fig. 2. A typical Autonomous System

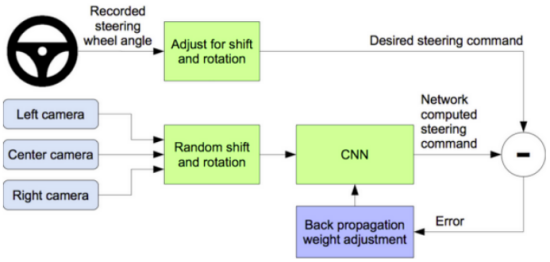

Fig. 3. Deep Driving Training

the previous components. The main activity of Control is to execute the strategy and functionalities controlling the hardware to generate intended motions [14]. *Path and Trajectory Tracking* are two aspects of control. Geometrical architecture of a strategy is considered as path to travel from start to end. Whereas the motion and it's velocity is concerned with trajectory [14].

V. DEEP DRIVING TRAINING & TESTING

Fig. 3 [16], shows an deep learning algorithm architecture for autonomous vehicle. The *"Desired steering command"* and *"Network computed steering command"* seen in the figure are the target and predicted values of the architecture. The target values are pre recorded steering wheel angle that adjust the shift and rotation of steering. Whereas predicted values are the output of network prediction using Convolutional Neural Network (CNN). Through Left, Center and Right camera of the vehicle random shift and rotation datasets are created and used as input to CNN. In comparing target and predicted values, cost function calculates loss values. The target is to minimize the loss, as a result best possible parameters are identified that match predictions with reality. Hence to optimize the model parameter, we use **gradient descent**. On the other hand research shows that gradient descent could ensure train-ability and generalizability of neural network with practical application of over-parameterization with high accuracy, i.e. gradient descent finds **global minimum** [13]. As of result of which autonomous vehicle system can achieve higher accuracy to reduce false prediction.

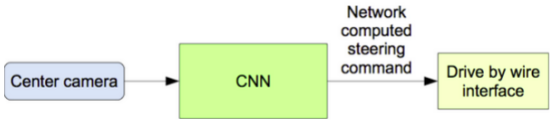

Fig. 4. Deep Driving Testing

TABLE II
NO. OF TRAINING SAMPLES & PARAMETERS

Setting	Training Samples	Validation Samples	Trainable Parameters
Under parameterized	50000	10000	2,634
Over parameterized	50000	10000	188,650
Parameter in order of n	50000	10000	48250

Fig. 4 [16], represents deep driving testing ,where the previously trained network is used for testing purpose. As shown in figure, camera gives new inputs to the CNN and it computes steering command to control the vehicles.

VI. IMPLEMENTATION

This implementation is to show the change in **trainability** and **generalizability** of the neural network due to the change in parameters of the network. Here **gradient descent** acts as a bridge and a tool to make change in trainability and generalizability using given parameters. Focus is on Convolutional Neural Network as it is widely used in autonomous vehicles for perception problems and MNIST datasets is used for the experiment. The experiment is done in three different settings as follows.

- *Under-parameterization*: Number of trainable parameters are tremendously less than the number of training samples.
- *Over-parameterization*: Number of trainable parameters are tremendously more than the number of training samples.
- *Practical degree of over-parameterization*: Total number of trainable parameters are in the order of n, where n is the number of training samples.

To make difference in the parameters, number of neurons in each layers is changed keeping the depth of network constant. Fig. 5, gives an overview of the network architecture particularly for practical degree of over-parameterization.

Table II, shows the network, data size and parameters used for training and validation.

Fig. 6, depicts training and validation accuracy for under-parameterized neural network. The number of trainable parameter is tremendously less then the training samples as in Table II. Under such condition the trainability and generalizability is assured as validation accuracy is higher than the training accuracy. But under-parameterized networks are not suitable for large datasets and large neural network.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_46 (Conv2D)           (None, 26, 26, 64)        640
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 13, 13, 64)        0
_________________________________________________________________
conv2d_47 (Conv2D)           (None, 11, 11, 64)        36928
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_48 (Conv2D)           (None, 3, 3, 16)          9232
_________________________________________________________________
flatten_16 (Flatten)         (None, 144)               0
_________________________________________________________________
dense_16 (Dense)             (None, 10)                1450
=================================================================
Total params: 48,250
Trainable params: 48,250
Non-trainable params: 0
```

Fig. 5. Network architecture for parameter ≈ n

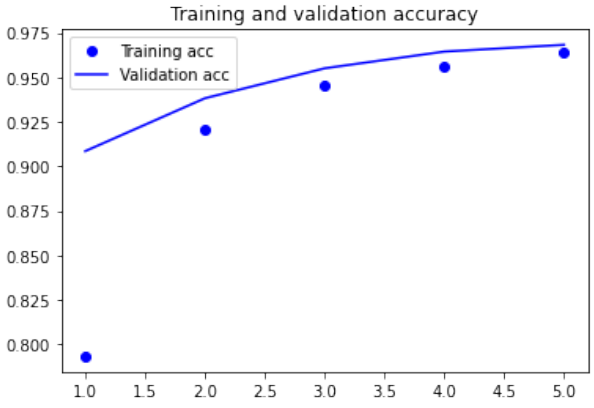

Fig. 6. Training & validation for under-parameterized network

The result of over-parameterization as in Fig. 7, shows that validation accuracy is less than training accuracy, which means the network overfits. In compared to under-parameterized network, the trainability in over-parameterized network is increased but is unable to ensure generalizability.

The third approach proves, under practical degree of over-

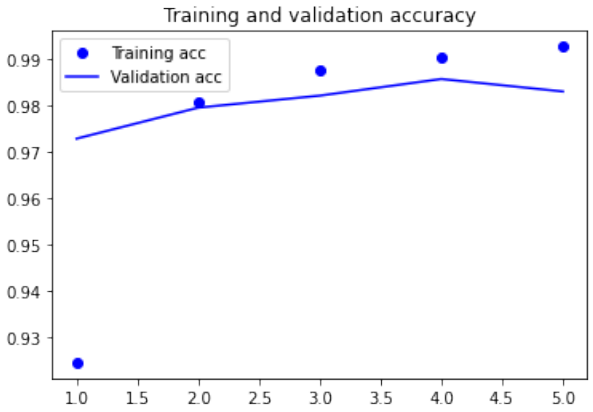

Fig. 7. Training & validation for over-parameterized network

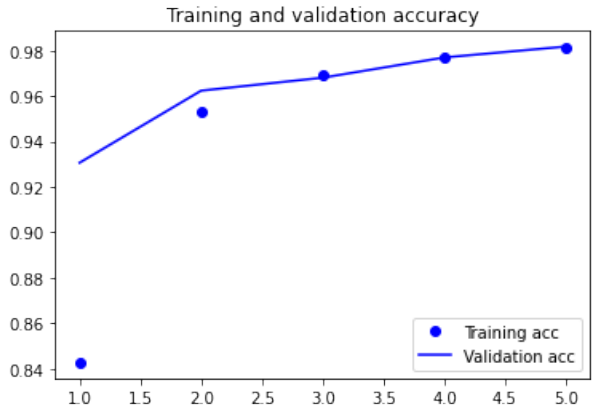

Fig. 8. Training & validation for practical degree of parameterization

parameterization, the network is able to make sure trainability and generalizability is achieved. As shown Fig. 8, training and validation accuracy are working well achieving higher accuracy and problem of network over-fitting is also solved.

VII. Analysis & Conclusion

Autonomous vehicle that are going to be an inevitable factor of future transportation, needs very high level performance and must assure reliability and safety. Mathematical algorithms are the brain of these systems and need to be optimized to improve their performance. **Gradient Descent** as one of the most popular optimization techniques in deep learning, it is widely used to increase the ability of intelligent vehicles algorithms. On the other hand, Gradient descent finding optimal solutions for these algorithms is foremost important to ensure safety, reliability and performance of the vehicles.

The literature research [13] and the experiment performed on MNIST datasets using CNN, showed a common result. To find a global minimum by gradient descent that is able to ensure trainability and generalizability of the neural network, practical degree of over-parameterization is the best approach. Though some research and experiment performed showed, over-parameterization can achieves global minimum but it is only able to ensure trainability and not generalizability.

References

[1] LeCun, Yann, B. Yoshua , and H. Geoffrey , “Deep Learning,” Nature, vol. 521.7553, pp. 436-444, 2015.

[2] G. Sorin, T. Bogdan, C. Tiberiu, M. Gigel, “A survey of deep learning techniques for autonomous driving,” Journal of Field Robotics, vol. 37.3, pp. 362-386, 2020.

[3] K. Kian, K. Daniel, M. Jiaju, ”Parameter optimization in neural networks,” 2019. [Online].Available: deeplearning, https://www.deeplearning.ai/ai-notes/optimization/. [Accessed June 07, 2020].

[4] R. Sebastian, “An overview of gradient descent optimization algorithms.,” arXiv preprint arXiv:1609.04747, 2016

[5] S. Matthew, ”Neural Network Optimization,” Jun 27, 2019. [Online].Available:https://towardsdatascience.com/neural-network-optimization-7ca72d4db3e0. [Accessed June 07, 2020].

[6] J. Prateek, K. Purushottam, “Non-convex optimization for machine learning.,” arXiv preprint arXiv:1712.07897, 2017

[7] L. Bi, O. Tsimhoni and Y. Liu, ”Using the Support Vector Regression Approach to Model Human Performance,” in IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 41, no. 3, pp. 410-417, May 2011, doi: 10.1109/TSMCA.2010.2078501.

[8] R. Sebastian, “An overview of gradient descent optimization algorithms.,” arXiv preprint arXiv:1609.04747, 2017

[9] A. Zeyuan, L. Yuanzhi, and S. Zhao, “A convergence theory for deep learning via over-parameterization.,” arXiv preprint arXiv:1811.03962, 2018

[10] S. Simon, D. Jason, L. Haochuan, W. Liwei, Z. Xiyu, “Gradient descent finds global minima of deep neural networks.,” arXiv preprint arXiv:1811.03804, 2018

[11] Z. Difan, C. Yuan, Z. Dongruo, G. Quanquan, “Stochastic Gradient Descent Optimizes Over-parameterized Deep ReLU Networks.,” arXiv preprint arXiv:1811.08888, 2018

[12] Z. Difan, G. Quanquan, “An improved analysis of training over-parameterized deep neural networks.,” Advances in Neural Information Processing Systems, 2019

[13] K. Kenji, H. Jiaoyang, “Gradient descent finds global minima for generalizable deep neural networks of practical sizes.,” arXiv preprint arXiv:1908.02419, 2019

[14] Kalra, Nidhi and Susan M. Paddock “Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?,” Transportation Research Part A-policy and Practice 94 (2016), pp. 182-193

[15] G. Sorin, T. Bogdan, C. Tiberiu, M. Gigel, “A Survey of Deep Learning Techniques for Autonomous Driving.,” arXiv preprint arXiv:1910.07738, 2019

[16] W. Narada, ”Deep Learning for Control in Robotics,” Nov 8, 2018. [Online].Available:https://www.uio.no/studier/emner/matnat/its/TEK5040/h18/lecture-slides/. [Accessed July 07, 2020].

[17] G. Ian, B. Yoshua, C. Aaron, “Deep Learning.,” MIT Press, 2016