# THE APPLIED STUDY OF THE OVERPARAMETERIZED NEURAL NETWORKS

**Qiaosong Zhou**
zhouqiaosong@126.com

**Pioneer Academics**

## ABSTRACT

This paper conducts a brief overview of the fundamentals of neural networks. It studies the effect of an over-parameterized model on optimization as well as generalization. Drawing reference to many research papers, this paper compares the results from each and evaluates their usefulness and validity with empirical application of networks with different parameters.

## 1 INTRODUCTION

The main purpose of this paper is to verify the results of Du and LeeDu & Lee (2018) empirically. In their paper, they demonstrated theoretically the power of over-parameterized models in a simple but limited form. Their model is one-layer and used a quadratic activation function $\sigma(x) = x^2$.

$$f(x, W) = \sum_{j=1}^{k} a_j \sigma(\langle w_j, x \rangle) \tag{1}$$

The validity of this result is tested empirically, and extended upon other configurations of activation functions. The main result of this paper is

The theoretical conclusion of Du & Lee (2018) is

## 2 BACKGROUND RESEARCH

This section covers the fundamental theory of neural network in a nutshell.

The purpose of neural networks is to create an appropriate algorithm for accurate prediction, while the construction of the algorithm is purely conducted through optimization. Neural networks could be used in a variety of different scenarios, and can yield much better results than algorithms designed by humans.

Under the instruction of Professor Wu, I have learned much about the fundamentals and capabilities of the neural network. From the earliest basic perceptron networks, to convolutional networks and recurrent neural networks. Modern state-of-the-art neural networks often features great depths and width with large sets of training data. These networks are capable from image-recognition, natural language processing to Robotics and Artificial intelligence.

### 2.1 BASIC FEED-FORWARD NETWORK

The simplest neural networks are fully connected with a small number of hidden layers. Simple feed-forward operations could be vectorized and expressed as the following

$$z_i^l = \sum_i f(w_{ji}^{l-1} x_j^{l-1}) \tag{2}$$

where $z_i^l$ is the $i$-th element in layer $l$, $f$ the activation function, $w_{ji}^{l-1}$ the $j$-th weight in layer $l-1$, $x_j^{l-1}$ the $j$-th input in layer $l-1$.

The loss function of the simple feed-forward network, which evaluates the difference between the correct output and the prediction by the model, is expressed through the following

$$L(w) = \frac{1}{j}\sum_{j}(y-\hat{y})^2 \tag{3}$$

which is a typical sum of square error. For categorization problems, cross-entropy loss is typically used.

$$-\sum_{c=1}^{M} y_c \log p_c \tag{4}$$

Figure 1 shows the structure of a simple fully-connected neural network. The arrows points in a direction of feed-forward calculation, and back-propagation of gradients happens in the opposite direction.

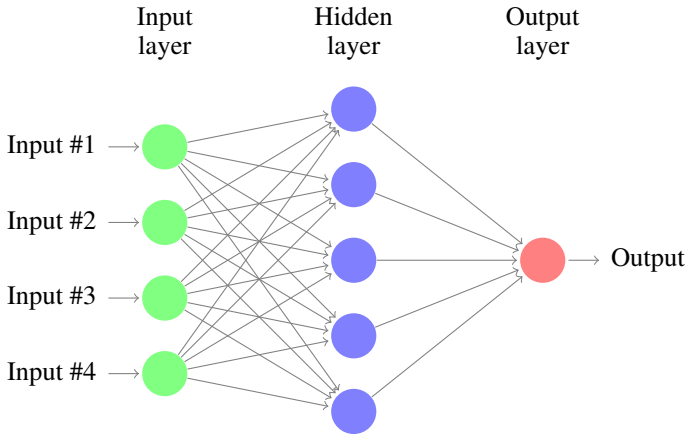



Figure 1: Structure of a simple fully-connected network

### 2.2 Backward propagation

Stochastic gradient descent is the most fundamental and often used optimization method used by machine learning algorithms. It works by calculating the analytic gradient of the loss function and approaching the local minima. For the simple network, the gradient is calculated from loss function, and propagated back through the neural network.

$$\frac{dL}{dw_{jl}^{l-1}} = \frac{dL}{d_z^l}\frac{d_z^l}{dw_{jl}^{l-1}} \tag{5}$$

### 2.3 Other types of neural networks

Convolutional Neural Networks (CNN) are often used in image processing. A pixel is quantified by RGB color, and a large number of pixels are in each image. This results in a large input vector. Convolution layers improves optimization by only training a local batch of pixels. This significantly reduce the number of weights and calculations required, while still capturing the local image features.

## 3 Different activation functions

Activation functions have an effect of introducing non-linearity to the model, and improves the optimization and learning speed of the model. The most commonly used activation function are

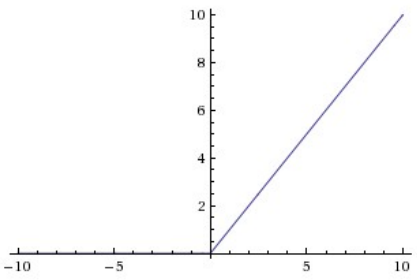
sigmoid, tanh and Relu. In the Du & Lee (2018) paper, however, their mathematical deductions of over-parameterization helps optimization only used a quadratic activation.

Sigmoid and tanh are saturating non-linear functions. Their gradient decays to 0 when $|x| \gg 1$. This slows down the learning process for weights with large norms. Relu, in contrast, is non-saturating non-linearities. The gradient does not flatten when $|x| \gg 1$. Non-saturating non-linearity has shown much greater speed in training, Although the Relu function has a 0 gradient for $x < 1$, leaky Relu function with a very small gradient for $x < 1$ can improve training results sometimes.

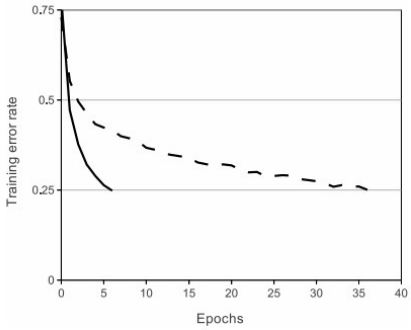
ReLUs have the desirable property of not needing input normalization to prevent saturation. From the results of Krizhevsky et al. (2017), if at least some training examples produce a positive input to a ReLU, learning will happen in that neuron.

In the empirical experiment that is conducted in this paper, quadratic activation, which is used in the Du & Lee (2018) paper to prove the two assumptions about optimization, is tested on small data set. Due to the inefficiency of the quadratic data set, the majority of the empirical analysis is conducted on Relu one-layer networks. Such inefficiency arises from the fact that the activation function becomes large very quickly. Therefore, it is prone to exploding gradient. In response to this problem, training steps have to be much smaller than those used in ReLU networks. The difference in terms of performance will be compared in this paper.

From the conclusion of Krizhevsky et al. (2017), ReLU activation indicates a 6x improvement in the speed of convergence compared to $tanh$ units.




(a) Rectified Linear Unit (Relu)




(b) Convergence of deep networks - Solid line: ReLU, Dashed: tanh

Figure 2: Results of Krizhevsky et al. (2017) for ImageNet, indicating a faster convergence with ReLU.

The assumption is that over-parameterized one-layer neural network with ReLU activation would perform much faster than one with a quadratic activation. Much better optimization of an over-paramterized network with ReLU activation compared to ones with quadratic activation could be expected.

## 4 Introduction to over-parameterized networks

Overparameterization enchances the model's ability to adapt better to a complex loss landscape. However, a complex mode has a tendency to overfit the training data. The problem of overfitting is further explored in Section 4.2.

When a neural network is overparameterized, according to Du & Lee (2018) paper, it is well-optimized. However, it is prone to overfitting, where the model fits the training data too well that it has a larger loss when applied to the prediction dataset. The Du & Lee (2018) paper covered

the theoretical proof of two conditions of overparameterization that guarantees optimization of loss function. When the conditions are satisfied, all loss functions modes would be or close to the global minima.

### 4.1 SUMMARY OF DU & LEE (2018) FINDINGS

On the Power of Over-parameterization in neural networks with Quadratic Activation paper has proven some interesting characteristics of an over-parameterized simple one-layer neural network.

By proving two properties:

**Property 1** (All local minima are global). *If $\mathbf{W}^*$ is a local minimum of $L(.)$ it is also the global minimum, i.e., $\mathbf{W}^* \subseteq \arg\min_{\mathbf{W}} L(\mathbf{W})$*

**Property 2** (All saddles are strict). *At a saddle point $\mathbf{W_s}$ there is a direction $\mathbf{U} \subseteq \mathbb{R}^{k \times d}$ such that*

$$vect(\mathbf{U}^T)\nabla^2 L(\mathbf{W_s})vect(\mathbf{U}) < 0 \tag{6}$$

For a one-layer network with quadratic activation function, it has $k$ hidden nodes, with an input size $d$ and $n$ training sets. Du & Lee (2018) considered two different types of overparameterization:

1. $k > d$
2. $\frac{k(k+1)}{2} > n$

The second requirement is often milder than the first one is practice. In both cases, if the requirement is satisfied, the loss surface has benign proerties that enable local search algorithms to find global minima, as all local minima are the same as the global minimum.

Using the theory of Rademacher complexity, which is defined as

$$\mathrm{Rad}(A) := \frac{1}{m}\mathbb{E}_\sigma \left[ \sup_{a \in A} \sum_{i=1}^{m} \sigma_i a_i \right] \tag{7}$$

The paper also proved that weight-decay helps generalization with the Rademacher theory. It found that with $l_2$ regularization, the generalization is bounded.

### 4.2 THE PROBLEM OF OVERFITTING

With the increase of parameters, an over-parameterized network is prone to the problem of overfitting. This happens when the model achieve a very small training loss by fitting the random noise in the training set, while making the compromise of worse generalization.

The result of Zhang et al. (2016) paper explores further the ability of deep and over-parameterized networks to achieve small loss even for randomly generated images. This shows the potential of overfitting that could happen. The primary finding of this paper suggests **deep neural network easily fit random labels**, while the paper seek to understand **what differs those models that generalize well and those that don't?** It questions the validity of traditional measures of generality, including VC dimension (Vapnik, 1998), Rademacher complexity (Bartlett & Mendelson, 2003), and uniform stability (Mukherjee et al., 2002; Bousquet & Elisseeff, 2002; Poggio et al., 2004).

In this paper, the empirical investigation observed little overfitting and good generalization when the number of training sets are sufficiently large. With large training sets and no explicit regularization such as weight decay, the model exhibits a small training error.

### 4.3 EVALUATION OF DU & LEE (2018) FINDINGS

The real-life application of this paper is limited, due to the poorly-optimized nature of the quadratic activation function, and the limited practicality of a one-layer neural network. In addition, it would be difficult if not impossible to extend the conclusion of this paper onto more complicated neural networks because the loss function is assumed to be smooth and convex. Non-convexity of many more

complex networks renders the results of this paper rather inapplicable in most settings. Nonetheless, the proof itself is an theoretical accomplishment that could be potentially extended to other activation functions.

The proof for the generalizability of the overparameterized neural network is undermined by the findings of Zhang et al. (2016), where the Rademacher complexity failed to indicate a tendency to overfit deep neural networks.

# 5 Empirical investigation of Over-parameterized networks

## 5.1 Design of Empirical investigation

The neural network used in this investigation is a simple one-layer neural network, with a mathematical description of Equation 1.

There are 3 parameters that influences the training of the network and whether it could be treated as over-parameterized. $k$ the number of hidden nodes, $n$ sets of training data, and $d$ size of input.

### 5.1.1 Datasets to train

1. The value of $k$ will be varied compared to $d$ with fixed $n$ in each trial. This is geared towards testing the first condition in Du & Lee (2018). Dataset 1 is used to test over-paramterized networks in regard to a fixed $d = 100$:

   $$k_1 = \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 120, 140, 160, 180, 200, 250, 300, 350, 400, 450, 500\}$$

2. The value of $k$ will be varied compared to $n$ with fixed $d$ in each trail. This is geared towards testing the second condition in Du & Lee (2018)

   $$k_2 = \{2, 3, 5, 7, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$$

### 5.1.2 Initialization of training data

The training data is initialized as a random normal distribution. The training inputs, training weights and noise are all initialized. The training output is calculated through the matrix multiplication of the input and the randomly generated weight.

### 5.1.3 Training details

Training trials are carried out with both ReLU and quadratic activation functions. All three parameters remain unchanged but the training steps are reduced for quadratic by about a factor of 100. If this is not carried out, the optimization process would encounter a large number of exploding gradient cases. The ability of quadratic activation to follow the convexity to the local minima of the loss landscape is much worse than ones with ReLU activation.

Each training trials are carried out 10 times with different initializations to find different local minima, should they exist. The initial weights in each trial is a randomly generated normal distribution. If 10 trials indicate This empirical investigation is limited by the lack of fast computing power. With enough computing power, more neural network setups with different parameters could be explored.

Specifically, datasets with following parameters are tested.

## 5.2 Training results

Despite Du & Lee (2018) having made two conditions that could guarantee optimization of over-parameterized networks, the difference is much less pronounced empirically, while different activation functions exhibit different patterns of loss optimization.

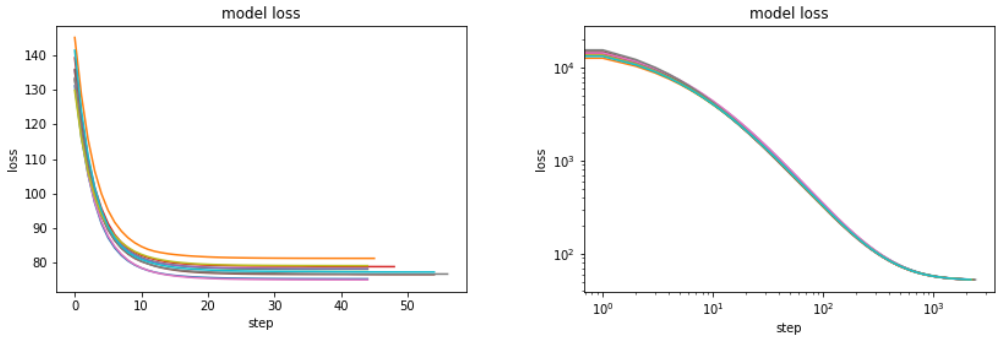
When providing a description for the training, if all 10 loss curves with randomly generated weights all descends to a same minima, then it is regarded as a global minima.

Table 1: All trials and results

| Trial | $d$ | $n$ | $k$ | $\sigma$ | **Step** | **description** |
|---|---|---|---|---|---|---|
| 1 | 1000 | 100 | $k_2$ | Quad. | 0.0001 | First convergence at k=25, total convergence at k=40 |
| 2 | 1000 | 100 | $k_2$ | Quad. | 0.00005 | First convergence at k=10, total convergence at k=20 |
| 3 | 1000 | 100 | $k_2$ | Quad. | 0.0001 | First convergence at k=10, total convergence at k=20 |
| 4 | 1000 | 100 | $k_2$ | Quad. | 0.0005 | First convergence at k=25, total k=40 |
| 5 | 100 | 10 | $k_2$ | Quad. | 0.00005 | Total convergence to single global minima. |
| 6 | 100 | 10 | $k_2$ | Quad. | 0.0005 | Total convergence to single global minima. |
| 7 | 50 | 1000 | $k_2$ | Quad. | 0.0001 | Total convergence to single global minima. |
| 8 | 50 | 100 | $k_2$ | ReLU | 0.5 | Total convergence, minima more tightly packed for larger k |
| 9 | 50 | 1000 | $k_1$ | ReLU | 0.5 | Total convergence to single global minima. |
| 10 | 10 | 100 | $k_1$ | ReLU | 0.5 | Total convergence, minima more tightly packed for larger k |
| 11 | 100 | 10 | $k_1$ | ReLU | 0.5 | Showing variance and difference, but less so for larger k |

**ReLU trains much faster than quadratic activation**

In Figure 3, two networks are trained with the same exact $k$, $n$, and $d$. ReLU networks exhibits much faster training, with $40$ steps on average to reach convergence. In contrast, quadratic network (which had a learning rate 1000 times smaller than ReLU to prevent exploding gradient) resulted in much larger initial loss, slow descent, and overall slow training. In this study, the training steps are tested to ensure no exploding gradients for quadratic activation networks.




(a) Convergence of d=100, n=10, k=100 network with ReLU activation

(b) Convergence of d=100, n=10, k=100 network with quadratic activation (log scale)

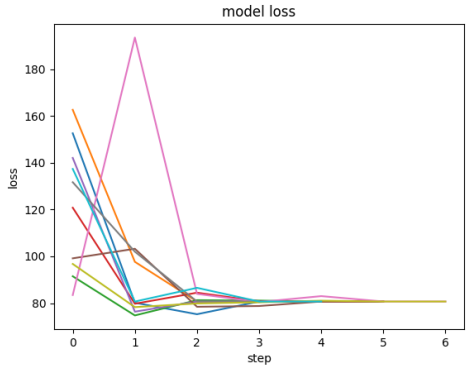
Figure 3: A comparison of steps taken by ReLU and quadratic activations to reach convergence.

**Different activation functions displays different loss properties**

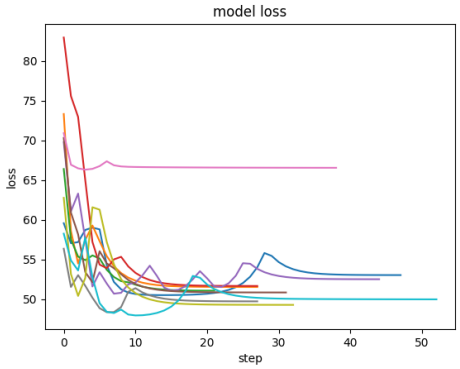
With small training sets, such as $n = 10$, ReLU exhibits an extent of variance in local minima. With larger $n$, such variance disappears and the training loss shows a efficient descent to global minimum. Interestingly, ReLU networks achieves a global minimum for very under-paramterized networks such as $k = 2$ and $k = 3$, as shown in Figure 4 . Generalization for $k = 3$ is also surprisingly good.

Loss functions with quadratic activations, in contrast, suffers from exploding gradient, but exhibits overwhelming convergence for the trials that gradients did not explode. In general, the smaller the step sizes are, the more likely that gradient will not explode. This means that exploding gradients could be eliminated with sufficiently small training step.

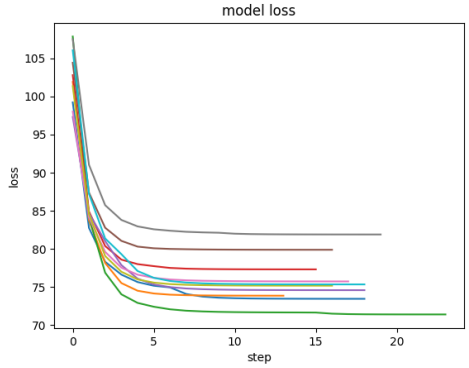
**Models with quadratic activation functions either diverges or converges. When it converges, it always reaches a global minimum.**



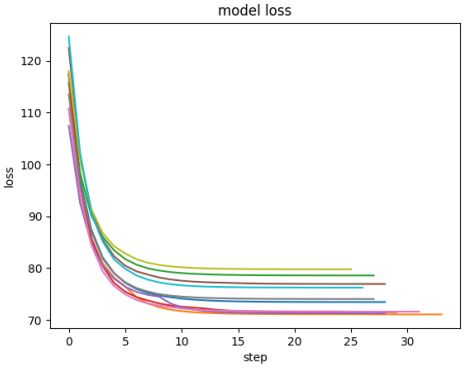

(a) Convergence of d=100, n=10, k=3 network with ReLU activation




(b) Convergence of d=100, n=10, k=5 network with ReLU activation




(c) Convergence of d=100, n=10, k=30 network with ReLU activation




(d) Convergence of d=100, n=10, k=50 network with ReLU activation

Figure 4: ReLU loss behaviour with different $k$ at typical stages

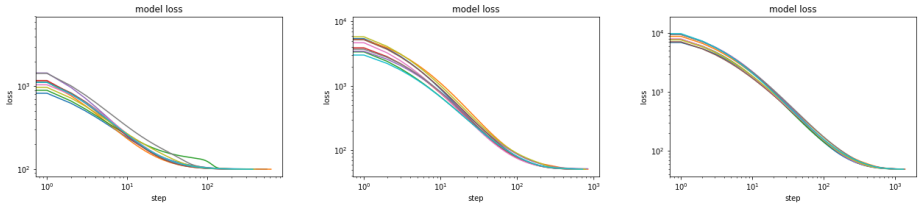
The faster training speed would ultimately mean that ReLU would be used much more than quadratic activation function. However, theoretically speaking, convergence of loss functions with quadratic activation do have the nice property of obtaining the same loss at each trial. This means that it is highly probable that it is the global minima.

**The Du & Lee (2018) results are weak when empirically tested**

1. Only ReLU training trials with small training sets reaches different local minima. Training trials with large training sets reaches a steady global minima almost regardless of the number of hidden nodes nor the type of activation. This conclusion made the first requirement of Du & Lee (2018) weak, and somewhat unnecessary.
2. When training steps are reduced to 5000 smaller than its equivalent ReLU network, the phenomenon of exploding gradient in quadratic activation networks could be eliminated. Empirical testing showed that with very small number of $k$, the model could achieve a global minimum with quadratic activation. In fact, networks with only $k = 2$ hidden nodes, as shown in 5 proved to be able to achieve global minima. Meanwhile, ReLU networks under the same circumstances could not achieve a global minimum.
3. For training trials conducted with training set $k_2$, which tests $n$ in the second requirement of Du & Lee (2018). When conducted two trials with different training steps, evidence has shown that the $k$ with which first convergence emerged, has a stronger correlation with the magnitude of training step rather than the number of hidden nodes $k$.

**Requirement 1 does not seem to affect networks with quadratic activation**

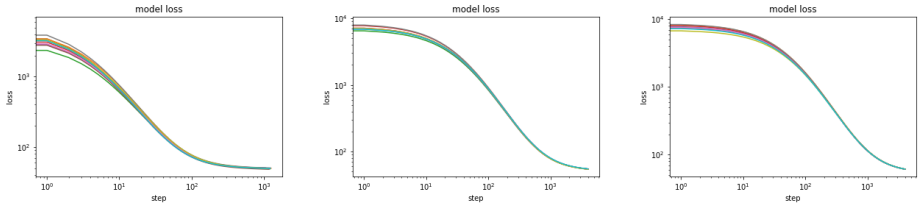
As shown by Figure 5, even with very small $k = 2$, the optimization of loss function with a quadratic activation converges to a global minimum. For ReLU networks with very small training size $n$, satisfying requirement does lead to a variance in the local minima achieved by the gradient descent optimization algorithm.




(a) d=100, n=10, k=2 Quadratic activation (b) d=100, n=10, k=5 Quadratic activation (c) d=100, n=10, k=10 Quadratic activation

Figure 5: A comparison of steps taken by ReLU and quadratic activations to reach convergence.

**Requirement 2 is does not seem to affect convergence in any trial**




(a) d=50, n=1000, k=3 Quadratic activation (b) d=50, n=1000, k=30 Quadratic activation (c) d=50, n=1000, k=50 Quadratic activation

Figure 6: A comparison of steps taken by ReLU and quadratic activations to reach convergence.

As shown in Figure 6, 3 sample cases are presented. When $k = 3$, the neural network is under-parameterized. When $k = 30$, the network is just about to satisfy $\frac{k(k+1)}{2} > n$. When $k = 50$, the network is over-parameterized, with $\frac{k(k+1)2}{2} >> n$. Empirical application shows an convergence for all conditions, even for very under-paramterized networks such as $k = 3$.

### 5.3 Generalization results

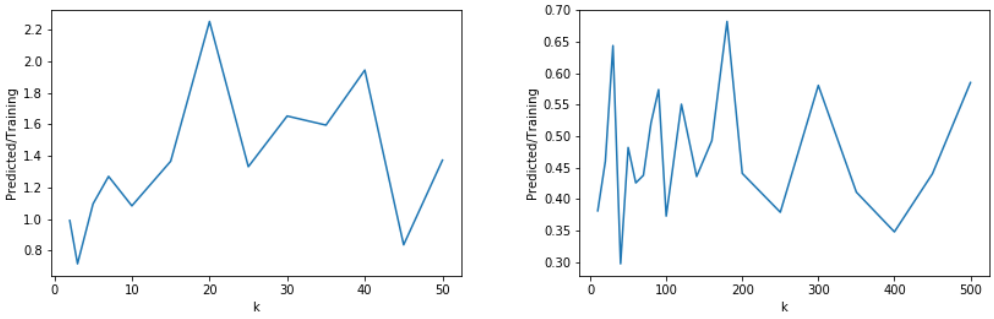
The ability of ReLU and Quadratic networks to generalize varies drastically. ReLU networks could generalize well with a small training set of $n = 10$, while quadratic networks would need $n = 1000$ to do the same.

#### 5.3.1 Quadratic Activation Function

Overall, the larger the $n$ is, the better generalization is. As a noise with normal distribution and a mean of 0 is also added to the training output, it is possible for the network to reach a predicted error that is lower than the training error.

When the training data is sufficiently large, the neural network in this empirical study could achieve a predicted error smaller than the training error without needing $l_2$ regularization for quadratic activation networks. However, for under-parameterized networks, such as when $n = 1000$ and $k = 10$, a generalization error half of training error is observed. Meanwhile, worse generalization for $k = 25$ is observed when generalization error is larger than the training error. For larger $k = 50$, the generalization error become lower than the training error again.

Therefore, the trend of generalization of such one-layer network with quadratic activation function is that the number of training set $n$ is directly correlated to a smaller generalization error.




(a) Ratio of predicted and training data vs $k$ ($k_1$)

(b) Ratio of predicted and training data vs $k$ ($k_2$)

Figure 7: Generalization for different sets of $k$ on ReLU network with $d = 100$ and $n = 10$

#### 5.3.2 ReLU Activation Function

ReLU networks share similar characteristics of generalization to quadratic networks, but its characteristics are much more unexpected. With a small $k = 3$, it a achieved a surprisingly small generalization error. As shown in Figure **??** , such a good generalization is only achieved much later.

For different sets, we can observe an unstable generalization error for different $k$. Despite that, the performance of ReLU networks are good with over-parameterized networks being able to generalize well within a ratio of 1.

## 6 Conclusions

This paper investigated the conclusion of Du & Lee (2018), and compared the different loss function properties of different neural networks.

1. The results of Krizhevsky et al. (2017) is verified. The ReLU network does train faster than quadratic networks by a factor of thousands.
2. Different activation functions result in different loss landscapes. Quadratic functions either converges to a single minima, or suffers from exploding gradient. ReLU networks' loss function converges to a single global minimum on most cases. In some cases with small $n$, different minima could be reached.
3. Requirement 1 has limited influence on ReLU networks, only affecting neural networks with small training sets. It does not have an effect on quadratic activation networks, as they achieves global minimum regardless of requirement.
4. Requirement 2 does not seem to affect any convergence, regardless of $k$, $n$, $d$, and activation function.
5. Generalization follows the trend of good generalization for over-parameterized networks. However, the trend is not stable and there are very under-parameterized networks that could generalize well.

## 7 Further Research

One potential path that further research can investigate is to investigate the effect of over-parameterization on multiple-layered networks. Such research could contribute towards describing the effect of over-parameterization on actual non-convex loss surfaces.

Another potential exploration is the properties of the generalization of the networks. I have found that very under-paramterized networks could have good generalization. The threshold for such behaviour could be investigated.

ACKNOWLEDGMENTS

Thanks to Professor YingNian Wu of UCLA, Professor of Statistics, for his excellent teaching and mentoring.

Thanks to Pioneer Academics for supporting this research project!

## REFERENCES

Simon S. Du and Jason D. Lee. On the power of over-parametrization in neural networks with quadratic activation. *CoRR*, abs/1803.01206, 2018. URL http://arxiv.org/abs/1803.01206.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Commun. ACM*, 60(6):84–90, May 2017. ISSN 0001-0782. doi: 10.1145/3065386. URL http://doi.acm.org/10.1145/3065386.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016. URL http://arxiv.org/abs/1611.03530.