SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

Jun o -
ttps or s rd rd

# Newton design: designing CNNs with the family of Newton's methods

Zhengyang SHEN[1,2], Yibo YANG[3], Qi SHE[2], Changhu WANG[2],
Jinwen MA[1*] & Zhouchen LIN[4,5*]

[1]*School of Mathematical Sciences, Peking University, Beijing , China;*
[2]*Bytedance AI Lab, Haidian District, Beijing , China;*
[3]*JD Explore Academy, Beijing , China;*
[4]*Key Laboratory of Machine Perception, School of Intelligence Science and Technology, Peking University, Beijing , China;*
[5]*Pazhou Lab, Guangzhou p , China*

v r v s Au ust A pt J nu r u s on n

**Abstract** Nowadays, convolutional neural networks (CNNs) have led the developments of machine learning. However, most CNN architectures are obtained by manual design, which is empirical, time-consuming, and non-transparent. In this paper, we aim at offering better insight into CNN models from the perspective of optimization theory. We propose a unified framework for understanding and designing CNN architectures with the family of Newton's methods, which is referred to as Newton design. Specifically, we observe that the standard feedforward CNN model (PlainNet) solves an optimization problem via a kind of quasi-Newton method. Interestingly, residual network (ResNet) can also be derived if we use a more general quasi-Newton method to solve this problem. Based on the above observations, we solve this problem via a better method, the Newton-conjugate-gradient (Newton-CG) method, which inspires Newton-CGNet. In the network design, we translate binary-value terms in the optimization schemes to dropout layers, so dropout modules naturally appear in the derived CNN structures with specific locations, rather than being an empirical training strategy. Extensive experiments on image classification and text categorization tasks verify that Newton-CGNets perform very competitively. Particularly, Newton-CGNets surpass their counterparts ResNets by over 4% on CIFAR-10 and over 10% on CIFAR-100, respectively.

**Keywords** CNN, dropout, optimization method, network design, Newton's method

**Citation** n Z Y Y n Y B t ton s n s n n C s t t o ton s
t o s C n In ttps or s rd rd

## 1 Introduction

In recent years, convolutional neural networks (CNNs) have become the leading machine learning methods in several real-world application domains, e.g., image recognition [1–6] and text processing [7–10]. In all, the structure of a CNN model determines its performance, thus designing CNNs is a key problem. However, most CNN structures, such as ResNet [ ] and DenseNet [5], are obtained by manual design, which is empirical, time-consuming, and lacking theoretical support.

In order to reduce the requirements for human expertise and labor, researchers are increasingly interested in designing neural networks automatically. One main strategy is network architecture search (NAS) [11–15], which searches for network architectures in a given search space. However, NAS uses a search strategy, and usually requires some extra computing power for search. In addition, these architectures are inherently obtained by learning from data, and still cannot provide any theoretical insight into the neural networks either.

Besides, there exist many studies [16– 1] devoted to designing neural networks from theoretical derivation, such as optimization algorithms, which are much more transparent and interpretable compared with manual design and NAS. These studies are mainly focused on the sparse coding or compressive sensing

Corr spon n ut or t p u u n n p u u n

(CS) problems, including signal/image recovery. Mathematically, the purposes of these problems are to infer the original signal $x$ from its randomized measurements $y = \quad x$, where is a linear projection. Traditional methods for CS solve a well-defined problem, e.g., $\min_x \| \quad x - y\| \quad + \lambda\|x\|$ , where $\lambda\|x\|$ is the regularization, and employ iterative algorithms to solve it, e.g., the iterative shrinkage-thresholding algorithm (ISTA) [ ], with iteration $x_k \quad = \mathcal{T}_{\lambda t}(x_k - \quad t \quad ( \quad x_k - y))$, where $\mathcal{T}_{\lambda t}$ is the soft-thresholding operator. Noting that this iteration resembles a network layer quite well when $\mathcal{T}_{\lambda t}$ is viewed as an activation function and is made learnable. Zhang et al. [ 0] unfolded ISTA iterations and proposed ISTA-Net.

Generally, we need to point out that this CNN design methodology inspired by optimization algorithms is an important part in differential programming. A common practice is firstly using an iterative algorithm to solve a well-defined problem, and then mapping the iterations to a data flow graph, which may correspond to a deep neural network. After the network structure is obtained, the parameters can be made learnable to increase the capacity. However, this CNN design methodology is limited to the above-mentioned sparse coding or CS problems, and cannot be directly applied to more general applications, where neural networks are used to extract features, such as the image recognition task. This is mainly because it is di cult to establish a well-defined optimization problem for feature extraction like CS problems, let alone we wish the derived optimization iterations to resemble network layers in form. Some studies [ – 5] addressed this issue by viewing the forward pass of CNN as sparse coding. However, these architectures cannot help understand some common CNN architectures, like ResNets, and have a high computational cost.

Li et al. [ 6] proposed another approach: they prove that computing with a standard feedforward neural network (PlainNet), when the weights are fixed and positive semi-definite, is equivalent to minimizing an objective function using the gradient descent algorithm, and assume that a better optimization algorithm may correspond to a better neural network architecture. With this new understanding, they use faster first-order optimization algorithms to minimize this objective function and design better neural network structures. However, the assumption that weight matrices are positive semi-definite is too strong, whereas we only need to assume weight matrices to be symmetric in this work.

Specifically, we observe that the PlainNet, when the weights are fixed and symmetric, can also be viewed as a kind of quasi-Newton method solving a well-defined optimization problem. Furthermore, we find that residual network (ResNet) can be derived by using an improved quasi-Newton method to solve this problem. Then, we utilize a better method, the Newton-conjugate-gradient (Newton-CG) method, to solve the problem, and propose Newton-CGNet, which contains branch structures and dropout modules naturally. In all, our theory proposes a unified framework for understanding and designing CNNs. Since that our theory understands some existing CNNs and designs new CNNs with the family of Newton's methods, we refer to it as Newton design.

We evaluate Newton-CGNets on both image classification and text categorization tasks. As for image classification, our models achieve lower classification error rates while using comparable numbers of parameters with the counterpart ResNets. Furthermore, the results are still competitive even compared with some advanced variants of ResNet. Without data augmentation, Newton-CGNets perform better than ResNets and its variants by a large margin. For text categorization, Newton-CGNets outperform VDCNNs [9] using fewer parameters, of which two versions exactly correspond to the counterparts PlainNets and ResNets.

Our contributions are as follows:

• We propose a unified framework for understanding and designing CNNs with the family of Newton's methods, which are mainly the second-order optimization methods. ResNet can be derived from our methodology.

• With our methodology, we translate binary-value terms in the optimization schemes to dropout layers. Then the specific locations of the dropout modules are naturally determined, rather than being positioned manually.

• Newton-CGNets perform very competitively on both image recognition and text processing applications.

## 2 Related work

There have been extensive studies on the neural network design. The main design strategies include manual design, NAS, and theoretical derivation, including optimization algorithms and ordinary differential equations (ODEs).

**Manual design.** The most common design strategy is manual design. As for image recognition, AlexNet [1] and VGG [ ] achieved breakthrough results in the ImageNet classification challenge, with feedforward CNN structures. Also, many new neural network structures have been proposed, such as GoogLeNet [ ], which contains several branches. ResNet [ ] is the first ultra-deep CNN model, where skip connections are applied to avoid gradient vanishing. Moreover, Huang et al. [5] proposed DenseNet, where each layer connects to all latter layers, in order to improve the information flow. Nevertheless, manual design is empirical, imposing high demand for human skills, and always time-consuming.

**NAS.** In the early stage of neural network design, genetic algorithm [ 7, 8] based approaches were taken to find both architectures and weights. However, they perform worse than the hand-crafted ones [ 9]. Also, Domhan et al. [ 0] used Bayesian optimization for network architecture selection. First adopted in [11], reinforcement learning is the main mechanism to assign a better structure with a higher reward. Follow-up studies [1 , 1 ] focused on reducing the search space and computational cost. But they are still time-consuming. Liu et al. [1 ] proposed differentiable architecture search (DARTS) and showed remarkable efficiency improvement. However, it is still unable to offer theoretical insight into the CNN architectures. In addition, NAS uses a search strategy and usually requires some computing power, while our method does not use any search strategy and computing power.

**Derivation from optimization theory.** CNNs derived by optimization algorithms are mainly for image restoration and reconstruction. The ISTA [ ] is a popular method for CS. Most of the existing neural network based methods [16, 18] induced by ISTA have the feedforward structures. Particularly, Zhang and Ghanem [ 0] proposed ISTA-Net inspired by ISTA and FISTA-Net inspired by the fast iterative shrinkage-thresholding algorithm (FISTA). Interestingly, the acceleration in FISTA naturally leads to skip connections in the network design, and FISTA-Net outperforms ISTA-net in experiments, consistent with the performance of their related optimization methods. Besides, the alternating direction method of multipliers (ADMM) is an efficient algorithm for CS magnetic resonance imaging models. Sun et al. [19] defined the ADMM-Net over a data flow graph inspired by ADMM. In conclusion, all the studies mentioned here unfolded optimization iterations to final networks with the practice of differential programming.

Later, some studies have proposed the interpretations of deep networks as unrolling optimization algorithms. Papyan et al. [ ] showed that the forward pass of the CNN is, in fact, the thresholding pursuit serving the multi-layer convolutional sparse coding model, and Sun et al. [ ] proposed a supervised deep sparse coding network for image classification. However, it remains unclear why such low-level sparse coding is needed for the high-level classification task. Chan et al. [ 5] pointed out that for high-dimensional multi-class data, the optimal linear discriminative representation maximizes the coding rate difference between the whole dataset and the average of all the subsets, and proposed ReduNet, which is derived by using a gradient ascent scheme for optimizing the rate reduction objective. However, they should use a large batch size for training and have a high computational cost. Also, ReduNet performs much worse than common models, e.g., ResNets. The closest work to ours is [ 6], which viewed the PlainNet as the gradient descent algorithm minimizing an objective function. Then they designed better neural network structures induced by employing faster first-order optimization algorithms to solve this objective. However, the assumption on the weight matrices being positive semi-definite is too strong, and they only used first-order algorithms.

**Derivation from ODE.** The connection between neural networks and ODEs may be first observed by [ 1], where the forward propagation of ResNet can be seen as an Euler discretization of a continuous transformation. Lu et al. [ ] proposed a linear multi-step architecture (LM-architecture) which is inspired by the linear multi-step method solving ODEs. Haber and Ruthotto [ ] used this connection to analyze the stability and well-posedness of deep learning, and developed more stable network architectures. Furthermore, Chen et al. [ ] introduced a continuous neural network. Instead of specifying a discrete sequence of hidden layers, they parameterized the derivative of the hidden state using a neural network. However, it cannot induce some operations naturally, such as dropout.

**Table 1** u r o not t ons n t s p p r

| ot t on | D s r pt on | ot t on | D s r pt on |
|---|---|---|---|
| $x_k$ | output o t k t r | $W_k$ | t tr |
| $A$ | t tr | | un t on |
| $\mathbb{S}^n$ | sp o s tr tr s | $\mathbb{R}^n$ | n ns on Eu n sp |
| $\mathbb{S}^n_{++}$ | sp o pos t v n t tr s | $P\ x$ | $P'\ x$ $x$ |
| $A^{\mathrm{T}}$ | tr nspos o tr v tor | $H_k$ | ppro t nv rs H ss n tr |
| $\|\cdot\|_2$ | nor | $\nabla F\ x$ | r nt o $F\ x$ |
| $\nabla^2 F\ x$ | H ss n tr o $F\ x$ | D · | n r t on tr |
| $\mathcal{P}$ | pro t on op r tor | $\mathcal{C}$ | $\mathcal{C}$ $\{x \mid x \succeq \ \}$ |
| $I$ nt t tr | | $U$ | $U$ $I -$ D $'$ $Ax_k$ |
| $r$ | $r$ $Ax_k - x_k$ | $Q$ | $Q$ $U^{\mathrm{T}}U$ |
| $b$ | $b$ $U^{\mathrm{T}}r$ | $g_t$ | r nt |
| $d_t$ | on u t r nt | $\alpha_t, \beta_t$ | s rs |
| $N$ | nu r o CG t r t ons o s | $L$ | pt o t ton CG t |

# 3 Newton design

## 3.1 CNN as iterations of optimization

For di erential programming, people may firstly use an iterative algorithm to solve a well-defined problem. Then they map the iterations into a data flow graph that may correspond to a neural network. Finally, the parameters in iterations can be made variable and learnable. However, for the image recognition task, we do not have a well-defined optimization problem in advance. Thus, we have to translate a known CNN structure to the optimization iterations solving an optimization problem firstly, in order to get a well-defined problem. All the notations in this paper are summarized in Table 1.

The most classic CNN structure is feedforward structures, such as AlexNet [1], which establishes the dominant status of CNNs in the computer vision field. Excluding the final softmax layer, the propagation from the first layer to the last layer, i.e., the process of extracting features (see Figure 1(a)), can be expressed as

$$x_k \quad = \quad (W_k x_k), \tag{1}$$

where $x_k$ is the output of the $k$-th layer, is an activation function and we set it as an ReLU. $W_k$ is a linear transformation implemented by a convolution operation. We call model (1) PlainNet in this paper. Actually, many neural networks, which implement linear transformations using some special convolutions, can be naturally categorized into the PlainNets. For instance, VGG [ ] uses × convolutions, while MobileNet [ 5] uses depthwise and pointwise convolutions. In this work, we focus on designing CNN architectures (i.e., the patterns of stacking convolutions) from the perspective of optimization theory, rather than the specific forms of convolutions. Thus, we uniformly denote the linear transformations as $W_k$ without any distinction in the theoretical derivation.

Following [ 6], we fix the matrix $W_k$ as $A$ to simplify the analysis, and get the iteration

$$x_k \quad = \quad (Ax_k). \tag{ }$$

Furthermore, we have the following observations.

**Proposition 1.** If $A \in \mathbb{S}^n$, $x \in \mathbb{R}^n$, is an ReLU, where $\mathbb{S}^n$ denotes the space of $n$-order symmetric matrices, then the iteration $x_k \quad = \quad (Ax_k)$ solves the optimization problem

$$\min_{x \succeq} \quad F(x) \equiv \frac{1}{-} x \ \ Ax - 1 \ \ P(Ax), \tag{ }$$

via a kind of quasi-Newton method (see the explanation in the Appendix A), where $A^-$ approximates the inverse Hessian of $F(x)$. $P'(x) = \quad (x)$.

*Proof.*

$$\nabla F(x) = A[x - \quad (Ax)], \tag{ }$$

where $\nabla F(x)$ is the gradient of $F(x)$, then

$$x_k \quad = x_k - A^- \ \nabla F(x_k)$$

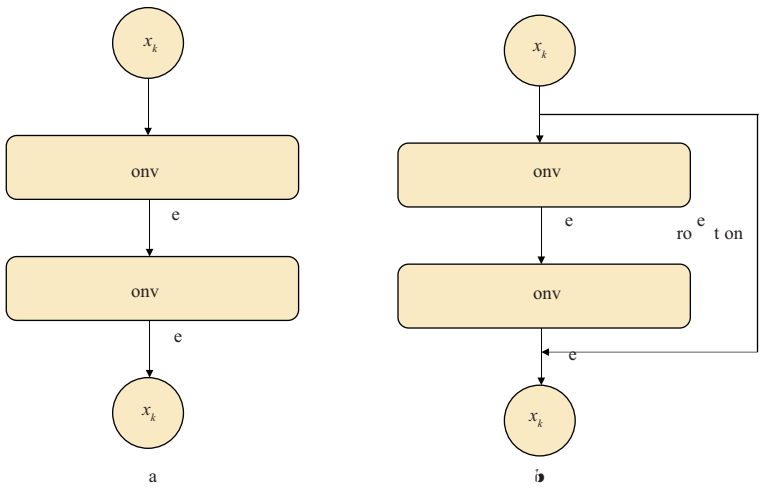



**Figure 1** Co or on n n t on ott n s t o t t o onvo ut on op r t ons s so non ott n s u B o n our p p r

$$= x_k - A^- [Ax_k - A (Ax_k)]$$
$$= (Ax_k). \tag{5}$$

Since is an ReLU, $x_k \succeq 0$ is satisfied. Thus iteration ( ) does solve the optimization problem ( ) with a kind of quasi-Newton method, where $A^-$ approximates the inverse Hessian of $F(x)$.

$F(x)$ may not be the only objective function that the iteration ( ) minimizes, but choosing $F(x)$ as ( ) seems very natural in form.

To get better insight into our theory, we analyze the gap between $A^-$ and the inverse Hessian of $F(x)$. We assume $\|A\| < 1$. Since

$$\nabla F(x) = A - A\mathrm{Diag}[ '(Ax)]A, \tag{6}$$

where $\nabla F(x)$ is the Hessian matrix of $F(x)$, then

$$\begin{aligned}[\nabla F(x)]^- &= [I - \mathrm{Diag}[ '(Ax)]A]^- A^- \\ &= \sum_n (\mathrm{Diag}[ '(Ax)]A)^n A^- \\ &= A^- + \sum_n (\mathrm{Diag}[ '(Ax)]A)^n A^- .\end{aligned} \tag{7}$$

The Neumann series can be expanded because $\|\mathrm{Diag}[ '(Ax)]A\| \leqslant \|A\| < 1$.

Obviously, the remaining term $\sum_n^\infty (\mathrm{Diag}[ '(Ax)]A)^n A^-$ cannot be neglected. On the other hand, the above quasi-Newton method only has a linear convergence rate (see the proof in the Appendix B), whereas a good quasi-Newton method may achieve a quadratic convergence rate. Thus $A^-$ is not a good enough approximation for the inverse Hessian. Instead, we can approximate the inverse Hessian by matrices $H_k$ that change over iteration (e.g., $H_k = A^- + \sum_n^m (\mathrm{Diag}[ '(Ax_k)]A)^n A^-$ , where $m$ is a given integer). As a result, the iteration scheme becomes

$$\begin{aligned}x_k &= \mathcal{P}_{\mathcal{C}}[x_k - H_k \nabla F(x_k)] \\ &= [x_k + H_k A( (Ax_k) - x_k)] \\ &= [(I - H_k A)x_k + H_k A (Ax_k)],\end{aligned} \tag{8}$$

where $\mathcal{P}$ is a projection operator and $\mathcal{C} = \{x | x \succeq 0\}$.

We can obtain the computation structure shown in Figure 1(b), which corresponds to the following iteration:

$$x_k = W_s^k x_k + W^k W^k x_k . \tag{9}$$

Eq. (9) is obtained by making the coe cient matrices in (8) learnable and variable. The structure in Figure 1(b) is non-bottleneck ResNet [ ] .

So far, we have translated the PlainNet (1) to an optimization method solving the problem ( ), building a bridge linking CNN models and optimization theory together. We use a more general quasi-Newton method to solve it, and derive ResNet. From this new understanding, we are able to design more promising and transparent CNN structures with optimization theory: optimize ( ) with better optimization methods and then inspire better CNN structures . Our theory explains and designs CNNs with the family of Newton's methods, so we call it Newton design.

### 3.2 Newton-CG method

Observing the problem ( ), we notice that the first term of $F(x)$, $x \ Ax/$ , is a quadratic term. Particularly, the Newton's method is very suitable to solve a quadratic problem, which only takes one iteration to obtain the solution. Thus we speculate that Newton's method would optimize ( ) better, and it will be verified in Subsection .1. . The iteration scheme of the Newton's method is as follows:

$$\begin{aligned} x_k &= \mathcal{P}_{\mathcal{C}}\{x_k - [\nabla \ F(x_k)]^- \ \nabla F(x_k)\} \\ &= \ \{x_k + [I - \mathrm{Diag}[\ '(Ax_k)]\, A]^- \ [\ (Ax_k) - x_k]\}. \end{aligned} \tag{10}$$

Noting that it is di cult to compute the inverse $[I - \mathrm{Diag}[\ '(Ax_k)]A]^-$ directly, we adopt the conjugate gradient (CG) method to compute it indirectly. We denote $U = I - \mathrm{Diag}[\ '(Ax_k)]A$ and $r = \ (Ax_k) - x_k$. Then we just need to compute

$$y = U^- \ r, \tag{11}$$

and $y$ is the solution of the optimization problem

$$\min_y \ h(y), \tag{1 }$$

where

$$h(y) = \frac{1}{-}(Uy - r) \ (Uy - r) = \frac{1}{-}y \ U \ Uy - r \ Uy + \frac{1}{-}r \ r. \tag{1 }$$

Again, we denote $Q = U \ U$ and $b = U \ r$, and the problem can be rewritten as

$$\min_y \ h(y) \equiv \frac{1}{-}y \ Qy - b \ y + \frac{1}{-}r \ r. \tag{1 }$$

We use the CG method to solve the problem. The procedure is shown in Algorithm 1, where $g_t$ and $d_t$ denote the gradient and the conjugate gradient, respectively.

**Algorithm 1** o v n t opt , t on pro v t CG t o

**Require:** p r t rs o t pro Q n b t nu r o t r t ons $N$
**Ensure:** so ut on o pro $y^\star$
t t n t po nt $y_0$
$g_0 \ \nabla h\ y_0 \ Qy_0 - b$
s t $d_0 \ -g_0$
**for** $t \ , \ , \ldots, N - $ **do**
$\alpha_t \ -\frac{g_t^{\mathrm{T}} d_t}{d_t^{\mathrm{T}} Q d_t}$
$y_{t+1} \ y_t \ \alpha_t d_t$
$g_{t+1} \ \nabla h\ y_{t+1} \ Qy_{t+1} - b$
$\beta_t \ \frac{g_{t+1}^{\mathrm{T}} Q d_t}{d_t^{\mathrm{T}} Q d_t}$
$d_{t+1} \ -g_{t+1} \ \beta_t d_t$
**end for**
**return** $y_N$

Theoretically, the CG method needs at most $n$ iterations to get the solution, where $n$ is the dimension of the matrix $Q$. However, $n$ is always very large, thus we always iterate $N$ times ($N < n$) to approximate the solution (see line ).

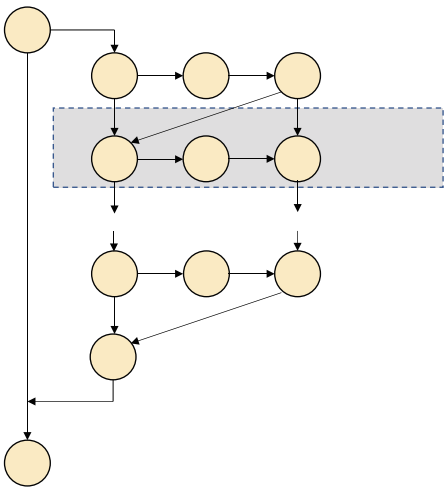
A t ou W s str t s n nt t pro t on n ost stu s t or n or s o t t tr t n t s rn n r pro t on so or s n t s or n r or
t propos t pot ss t t tt r opt , t on or t orr spon to tt r n ur n t or r t tur

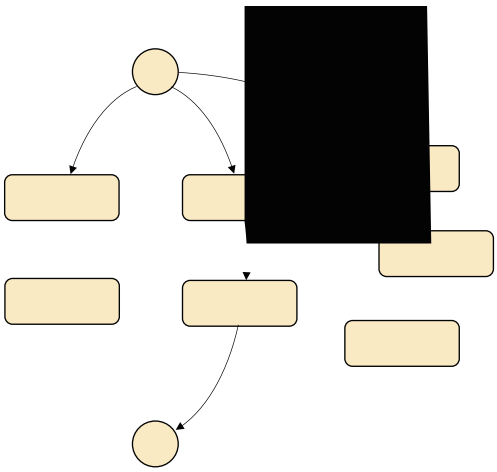
**Table 2** It r t on nu rs o us n qu s ton t o s n ton CG t o s to so v t pro

| t o | tt n | It r t on |
|---|---|---|
| | *m* | |
| u s ton t o | *m* | |
| | *m* | |
| | *N* | p |
| | *N* | |
| ton CG t o | *N* | p |
| | *N* | |
| | *N* | |

### 3.3 Numerical experiments

Before unfolding Newton-CG iterations to a CNN architecture, we show the numerical performance of





rt

**Algorithm 2** or r, prop t, on o ton CG B o ro $x_k$ to $x_{k+1}$

**Require:** , nput $x_k$ t nu r o CG B o s $N$ t onvo ut, on rn s $W_t^{(1)}, W_t^{(2)}, W_t^{(3)},$ n, $W_t^{(4)}, \leqslant t \leqslant N -$ t
s rs $\alpha_0, \alpha_1, \ldots, \alpha_{N-1}, \beta_0, \beta_1, \ldots, \beta_{N-2}$
$y_0$ $x_k$
,

also added dropout modules to Wide ResNet at similar locations. However, their strategy is empirical.

**Architectures.** In order to faciliate the analysis and comparison, we design the architecture of Newton-CGNets based on ResNets. As shown in Figure , the Newton-CGNet contains branch structures. Thus with the same depth, it contains twice more convolution kernels than the ResNet. Naturally, we modify two Residual Blocks onv onv × to one CG Block conv1; conv ; onv onv (corresponding to the topology in Figure (b)).

Generally, we use $L$-$\{N\ , N\ , N\ \}$ to denote the Newton-CGNet architecture which contains Newton-CG Blocks with $L$-layer depth totally. And each Newton-CG Block contains $N$ , $N$ , and $N$ CG Blocks, respectively.

**Training details.** All the models are trained using SGD and a Nesterov momentum [ ] of 0.9 without dampening. During the training phase, we find that our models can converge stably when we adopt common settings used in existing CNNs. Specifically, we adopt the weight initialization method in [ ] for convolutional layer and use Xavier initialization [ 5] for the fully connected layer. On CIFAR and SVHN we train our models using batch size 1 8 for 00 and 0 epochs, weight decay of $5 \times 10^{\text{ }}$ and $10^{\text{ }}$ , respectively. The initial learning rate is set to 0.1 and is divided by 10 at 50% and 75% of the total number of training epochs. We add an ReLU after conv of each CG Block to supplement some nonlinearity . We adopt batch normalization (BN) [ 6] after each convolution kernel. Following [ ], we perform a linear projection to match the dimensions for addition operation whenever in need, with a $1 \times 1$ convolution kernel. We use 0. dropout rate on C10+ and C100+, 0. dropout rate on SVHN, and 0. dropout rate on C10 and C100, respectively. Since the dropout module is an integral part of the Newton-CGNet, rather than merely a training strategy, we can adopt dropout fairly. All the learnable scalars are initialized as 1.0. We report the median of 5 runs.

### .1. *Newton-CGNets vs. ResNets*

As we have shown in Table , compared with quasi-Newton methods, Newton-CG methods perform worse than quasi-Newton methods when the interior CG methods iterate only a few times and perform better when the CG methods iterate enough times. Naturally, it is interesting to explore how their derived CNNs perform. In fact, the number of the CG Blocks in each Newton-CG Block relates to the number of the CG iterations, thus we explore the performance of Newton-CGNets via changing the number of the CG Blocks.

We now evaluate our models on C10+. We take $L$-$\{N\ , N\ , N\ \}$ as 10-$\{1, 1, \ \}$, 16-$\{\ ,\ ,\ \}$, -$\{\ ,\ ,\ \}$, 8-$\{\ ,\ , 5\}$, 56-$\{9, 9, 9\}$, and 8 -$\{1\ , 1\ , 1\ \}$, and get Newton-CGNet-10, 16, , 8, 56, and 8 , respectively. On one TITAN Xp GPU, these models take 9, 15, 1, 6, 5 , and 75 s for training for one epoch, 1, , .5, , 7, and 11 s for inference, respectively. For fair comparison, we compare the performance of Newton-CGNets with its counterpart ResNets using comparable numbers of parameters and also run ResNets for 00 epochs. The results are listed in Table . The error rates resulted from ResNets are slightly better than that reported in [ ], due to more training epochs.

We plot the results in Figure (a) and observe that when using a few CG Blocks ($N \leqslant 5$), Newton-CGNets perform worse than its counterpart ResNets. And when we use more CG Blocks ($N \geqslant 9$), Newton-CGNets perform better. To be specific, with comparable numbers of parameters, Newton-CGNet-56 and 8 surpass ResNet-110 and 16 by 0. 8% and 0.67%, respectively. This phenomenon is very similar to that shown in Table . To conclude, as for this image recognition task, iterative algorithms (quasi-Newton methods and Newton-CG methods) and their derived CNN models (ResNets and Newton-CGNets) show the similar pattern: the better an iterative algorithm approximates the inverse Hessian, the better the iterative algorithm solves the optimization problem, also, the better its derived CNN model performs.

In addition, we investigate the sensitivity of some important hyperparameters for model training, including learning rate and weight decay, based on Newton-CGNet-56. As shown in Figure , when the learning rate is 0.1, our model performs well when the weight decay is between $10^{\text{ }}$ and $10^{-\ }$. When the weight decay is $5 \times 10^{\text{ }}$ , our model performs well when the learning rate is between 0. and 0.0 . To conclude, our model can perform stably when the weight decay and learning rate are around $5 \times 10^{\text{ }}$ and 0.1, respectively.

s o t on s n ss r us ton CG B o s n r n r o ot r s t o t on s nor s poss n or r to nt n t r v s

**Table 3** st rror r t s on C us n s ts n ton CG ts n o p uns

| t o | D pt | r s | Error | t o | D pt | r s | Error |
|---|---|---|---|---|---|---|---|
| | | | | | | | p |
| | | | | | | | |
| s t | | | | ton CG t | | | |
| | p | p | | | p | | p p |
| | | | p | | | | p p |
| | | | p p | | | | |

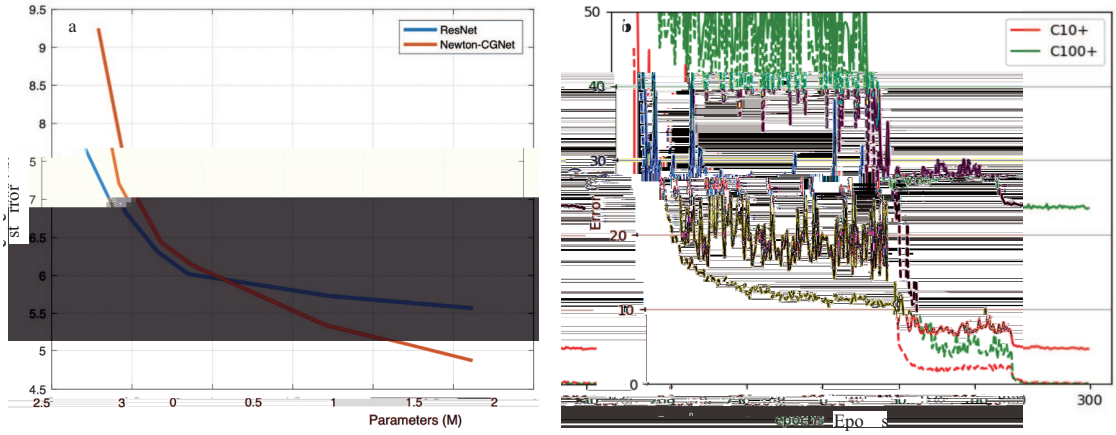



**Figure 3** Co or on n ton CG ts p r or ors t n s ts us n CG B o s n outp r or s ts us n or CG B o s tr n n urv s on C n C D s n s not tr n n rrors n so n s not t st rrors

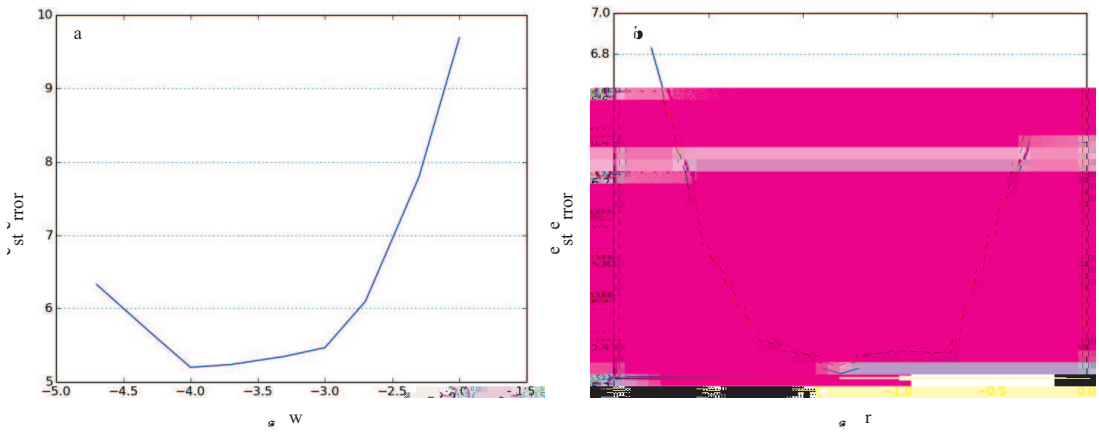



**Figure 4** Co or on n st rror r t s o ton CG t p t t rn n r t o . n r nt t t t st rror r t s o ton CG t p t t t o p n r nt rn n r t s rs

Actually, since that the performance of a CNN model is dependent on multiple factors, such as datasets and training strategies, it is di cult to accurately predict how the derived CNN models perform. However, Newton design provides an optimization perspective to help analyze and explain the performance of the derived CNN models qualitatively, and then guide us to use the derived CNNs more e ciently, e.g., using enough CG Blocks. By contrast, we cannot analyze the CNN models obtained by manual design or NAS in this way.

### .1. *Newton-CGNets vs. competitive models*

Using enough CG Blocks, we compare our derived Newton-CGNets with some more competitive models on three datasets, C10, C100, and SVHN, respectively. The test error rates are listed in Table .

**Newton-CGNets vs. advanced variants of ResNets.** On the dataset with data augmentation, Newton-CGNet-8 results in .90% on C10+ and .87% on C100+, outperforming its counterpart ResNet-16 by 0.67% on C10+ and .7 % on C100+, respectively. And the results are at least compara-

**Table 4** st rror r t s on CIFA n H t s ts n o p uns [a)]

| t o | D pt | r s | C | C | C | C | H |
|---|---|---|---|---|---|---|---|
| s t | | | | p | | | |
| | | | | p p | p | | p |
| s t t sto st pt | | | | p | | p | p |
| s t t ropout | | | - | p | - | | |
| s t pr t v t on | | | | p | p p | | - |
| o t p | p | | | | | | |
| Fr t t t ropout rop p t | | | | **4.60** | | | |
| HB t | | | | p | | | - |
| u t | | p | - | | - | - | - |
| | p | | | p p | | p | p |
| ton CG t ours | | | **6.80** | | **27.06** | | **1.57** |
| | | p | - | | - | **23.44** | - |

st r su ts r t n o

ble with all the listed variants of ResNet, including Wide ResNet [ 0], ResNet with stochastic depth [ 9] or pre-activation [ ]. We furthermore increase the number of CG Blocks and obtain a CNN architecture over 100-layer deep. Concretely, we take $L$-$\{N, N, N\}$ as 110-$\{18, 18, 18\}$ and get Newton-CGNet-110. The behaviors of Newton-CGNet-110 are shown in Figure (b), indicating that this model can be optimized without di culty.

On the dataset without data augmentation, our models perform even better. To be specific, Newton-CGNet-8 results in 6.80% on C10, 7.06% on C100, and 1.57% on SVHN, significantly surpassing its counterpart ResNet-16 by .88% on C10, 11.59% on C100, and 0.18% on SVHN, with comparable numbers of parameters, respectively. In addition, our models outperform all the listed variants of ResNet significantly.

**Newton-CGNets vs. MobileNets.** Inherently, the MobileNet [ 5] can be naturally categorized into PlainNets, where the linear transformation is implemented using much more e cient depth-wise separable convolutions. Considering that the reported MobileNet architecture uses 5 downsampling layers (convolutions of stride ) to process the input size of $\times$ , whereas the images in CIFAR and SVHN are only of size $\times$ , directly employing that setting will result in very low resolution ($1 \times 1$) after the last convolution. So we remove the first three dowmsampling and preserve the last two, in consistent with the setting of Newton-CGNets for fair comparison. As shown in Table , Newton-CGNet-8 perform better than MobileNet on all tasks using fewer parameters ( .6 M vs. . 6M), even though we only employ conventional convolutions, which shows great superiority of our architecture.

**Newton-CGNets vs. FractalNets.** FractalNet [ 7] employs fractal architecture instead of residual connections to build DNNs, and achieve much more competitive results. Compared with it, Newton-CGNets perform comparably when using data augmentation ( .66% vs. .60% on C10+ and . % vs. .7 % on C100+), and better without data augmentation (6.80% vs. 7. % on C10, 7.06% vs. 8. 0% on C100, and 1.57% vs. 1.87% on SVHN), using less than 10% parameters, which indicates great parameter e ciency of our method.

**Newton-CGNets vs. other optimization-inspired networks.** Our method significantly outperforms ReduNet ( .66 vs. 7.00 on C10+), which is designed by solving a rate reduction objective. Also, ReduNet needs to use a large batch size (about 1000), so it has a much higher computational cost. Compared with HB-Nets, our models achieve comparable results on C10+ and C100+, and perform much better on C10 and C100. Noting that HB-Nets are essentially inspired by a first-order optimization method, while ours are by a second-order method, the better performance also indicates that a faster optimization method would help design a better network architecture.

### .1.5 *Training details and results for ImageNet*

Also, we get the Newton-CGNet architecture for ImageNet by modifying ResNet. Particularly, the ResNets with 50, 101, and 15 layers are stacked by multiple "bottleneck" building blocks, di erent from the non-bottleneck residual blocks derived in our paper (see Figure 1(b)). As for ResNet-18, each group of convolution kernels only contains residual blocks. As discussed in Subsection .1. , Newton-CGNets with a few CG Blocks do not perform well. Thus it dose not make sense to modify ResNet-18 to the

**Table 5** top n top p n rop rror r t s on t v t on s t o I t t s t n o p runs

| t o | D pt | r s | op rror | op p rror |
|---|---|---|---|---|
| s t | | p | | |
| s t | | | | p |
| ton CG t | | | p | |

**Table 6** r s t t t or t on t s ts us n our p r nts

| D t s t | r n | st | C ss s | Av r or s | C t or t on t s |
|---|---|---|---|---|---|
| AG n s | | | p | p | En s n s t or t on |
| o ou n s | p | | p | p | C n s n s t or t on |
| DB | p | | p | pp | nto o ss t on |
| Y p r v po r t | p | | | p | nt nt n s s |
| Y p r v u | p | p | p | pp | nt nt n s s |
| Y oo ns rs | p | | | | op ss t on |
| A on r v u | | p | p | | nt nt n s s |
| A on r v po r t | | p | | | nt nt n s s |

Newton-CGNet. Consequently, we choose ResNet- as our basic model.

For ResNet- , the numbers of the residual blocks with dif erent output sizes are , ,6, and , respectively. In order not to get too shallow Newton-CG blocks, we only modify the third group of convolution kernels to a Newton-CG block, with the other parts unchanged. In addition, in order to utilize enough CG blocks without introducing more parameters, we specifically reduce the parameters in each CG block. Concretely, the residual blocks $\begin{bmatrix} \times & ; \\ \times & ; \end{bmatrix}$ p $\times 6$ are modified to the CG blocks $\begin{bmatrix} \times & ,6 ; \\ \times & ,6 ; \\ \times & ; \end{bmatrix}$ p $\times 6$, composing a Newton-CG block. Correspondingly, the method of computing $g_t$ is modified to

$$g_t = y_t + [g_t \quad , g_t \quad , g_t \quad ], \tag{18}$$

where $[g_t \quad , g_t \quad , g_t \quad ]$ refers to the concatenation of the feature-maps produced in three branches. It is not contradictory to the derivation in lines –6 of Algorithm , because this equals to the case that some channels of $W_t \quad , W_t \quad$, and $W_t^{\prime}$ are fixed to be zeros.

We initialize the learning rate as 0.1, with the batch size of 56. We set the dropout rate as 0. . For using dropout, we train our model for 100 epochs and drop the learning rate by 0.1 at epoch 0, 60, and 90. The other training details are the same as that for CIFAR and SVHN. We report the median of 5 runs, and the results are shown in Table 5. The top-1 and top-5 single-crop error rates resulted from Newton-CGNet- are 5.98% and 8. %. With comparable numbers of parameters and the same depth, Newton-CGNet- surpasses ResNet- by 0.75% and 0. % for top-1 and top-5 single-crop error rates, respectively. This indicates that our proposed models can also be applied on large datasets. In future work, we will study how to modify the bottleneck structure to our Newton-CG block.

### 4.2 Text categorization

As for text processing, we evaluate our Newton-CGNet on 8 freely available large-scale datasets introduced by [8] which cover several text categorization tasks, including English/Chinese news categorizatiot ccil[8

**Table 7** st rror r t s on t t s ts n o p uns [a)]

| t o | D pt | r s | AG | o ou | DB | Y p | Y p F | Y A | A | F | A |
|---|---|---|---|---|---|---|---|---|---|---|---|
| DC t out s ort ut | | p | | | | p | p | p | | | |
| DC t s ort ut | | p | | | | p | p | p | | | |
| ton CG t ours | | | **7.87** | **3.31** | **0.98** | **4.11** | **35.41** | **26.28** | **36.94** | **3.92** | |

st r su ts r t n o A t o s us poo n or o ns p n t n onvo ut on rs

H Z n X n t D p r s u rn n or r o n t on In ro n s o IEEE Con r n on Co put r s on n tt rn o n t on -
Hu n G u Z t n t D ns onn t onvo ut on n t or s In ro n s o IEEE Con r n on Co put r s on n tt rn o n t on
Y n Y Z on Z n t Convo ut on n ur n t or s t trn t up t qu In ro n s o IEEE Con r n on Co put r s on n tt rn o n t on -
Y Convo ut on n ur n t or s or s nt n ss t on In ro n s o Con r n on E p r t o s n tur n u ro ss n - p
Z n X Z o J B Cun Y C r t r v onvo ut on n t or s or t t ss t on In ro n s o t t Int rn t on Con r n on ur In or t on ro ss n st s p - p
Conn u A n H B rr ut t r p onvo ut on n t or s or t t ss t on In ro n s o t pt Con r n o t Europ n C pt r o t Asso t on or Co put t on n u st s -
Jo nson Z n D p p r onvo ut on n ur n t or s or t t or t on In ro n s o t pp Annu t n o t Asso t on or Co put t on n u st s p - p
B r B Gupt t D s n n ur n t or r t tur s us n r n or nt rn n In ro n s o Int rn t on Con r n on rn n pr s nt t ons
u C Zop B u nn t ro r ss v n ur r t tur s r In ro n s o Europ n Con r n on Co put r s on -
H Gu n Y Zop B t E nt n ur r t tur s r v p r t r s r n In ro n s o t pt Int rn t on Con r n on n rn n p
uH on n Y n Y D rts r nt r t tur s r In ro n s o Int rn t on Con r n on rn n pr s nt t ons
C n X X uJ t ro r ss v r nt r t tur s r r n t pt p t n s r n v u t on In ro n s o IEEE C F Int rn t on Con r n on Co put r s on -
Gr or Cun Y rn n st ppro t ons o sp rs o n In ro n s o t t Int rn t on Con r n on Int rn t on Con r n on n rn n
u n J A on r pro t on n ur n t or or nons oot opt t on su t to n r qu t s n oun onstr nts IEEE r ns ur t rn st -
X n B n Y G o t sp rs t t pnt or s In ro n s o t t Int rn t on Con r n on ur In or t on ro ss n st s
Y n Y un J H B t D p AD t or o pr ss v s ns n I In ro n s o t t Int rn t on Con r n on ur In or t on ro ss n st s -
Z n J G n B I A t nt rpr t opt t on nsp r pnt or or o pr ss v s ns n In ro n s o IEEE Con r n on Co put r s on n tt rn o n t on -
G r s E r Y C Bronst n A t r os t n onv r n sp n r onstru t on ur n nv rs pro s IEEE r ns n ro ss -
B A ou A st t r t v s r n t r s o n or t or n r nv rs pro s IA J I -
p n o no Y E Convo ut on n ur n t or s n onvo ut on sp rs o n J rn s -
un X sr r n D up rv s psp rs o n n t or s or ss t on IEEE r ns I ro ss p
C n H Yu Y You C t u t t o pnt or ro t pr n p o n r t r u t on ArX v p
H Y n Y C n D t pt t on or t nsp r pnur n t or stru tur s n In ro n s o t t As n Con r n on n rn n -
r J D t D Es n J Co nt t ons o n t or t s n nur n t or s surv o t st t o t rt In ro n s o Int rn t on or s op on Co nt t ons o G n t A or t s n ur t or s -
un F H F H n H t un n o t stru tur n p r t rs o n ur n t or us n n prov nt or t IEEE r ns ur t -
r n s s H r uss J G n r t v n uro vo ut on or p rn n ArX v p
Do n pr n n r J Hutt r F p n up uto t p rp r t r opt t on o pnur n t or s tr po t on o rn n urv s In ro n s o t t Int rn t on Con r n on Art Int n p
E A propos on n rn n v n s st s Co un t t t p -
u Y Z on A t B on n t r n ur n t or s r n p r t tur s n nu r r nt qu t ons In ro n s o t pt Int rn t on Con r n on n rn n - p
H r E ut otto t r t tur s or pn ur n t or s Inv rs ro
C n u nov Y B tt n ourt J t ur or n r r nt qu t ons In ro n s o t n Con r n on ur In or t on ro ss n st p - p
Ho r A G Z u C n B t o nts nt onvo ut on n ur n t or s or o v s on pp t ons ArX v
r vs A H nton G rn n ut p rs o F tur s ro n I s n port
C Y X G r t D p sup rv s nts In ro n s o t t Int rn t on Con r n on Art

Int n nt t t st s p p -p
t r Y n Co t s A t n ts n n tur s t unsup rv s tur rn n In ro n s o Con r n on ur In or t on ro ss n st s
Hu n G un Y u Z t D p n t or s t sto st pt In ro n s o Europ n Con r n on Co put r s on -
Z oru o o o s r s u n t or s ArX v p
D n J Don o r t I n t r s r r t s In ro n s o IEEE Con r n on Co put r s on n tt rn o n t on - pp
H Z n X n t I nt t pp n s n p r s u n t or s In ro n s o Europ n Con r n on Co put r s on -p
uts v r I rt ns J D G t n t port n o n t t on n o ntu n p rn n In ro n s o t t Int rn t on Con r n on Int rn t on Con r n on n rn n -
H Z n X n t D v n p nto r t rs surp ss n u n v p r or n on n t ss t on In ro n s o IEEE C F Int rn t on Con r n on Co put r s on p -
p G orot X B n o Y n rst n n t u t o tr n n p or r n ur n t or s In ro n s o t t Int rn t on Con r n on Art Int n n t t st -p
Io C B t nor t on r t n p n t or tr n n r u n nt rn ov r t s t p ArX v p
rsson G r n rov G Fr t n t u tr p n ur n t or s t out r s u s In ro n s o Int rn t on Con r n on rn n pr s nt t ons