SCIENCE CHINA

Information Sciences

• RESEARCH PAPER •



Jun o Lttps o or s

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

Zhengyang SHEN^{1,2}, Yibo YANG³, Qi SHE², Changhu WANG², Jinwen MA^{1*} & Zhouchen LIN^{4,5*}

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

v r vs Auust Apt Jnur u gon, 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	ҮВ	\mathbf{t}	wton s n	s n n C	swy y "n	O,	wton s
n tos	∫ ¶_n In, _	" Au		f ^{ttps} o or	s	.1 .1		

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 [7] 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

🖌 n Z Y, et al. Sci China Inf Sci Jun o

(CS) problems, including signal/image recovery. Mathematically, the purposes of these problems are to infer the original signal x from its randomized measurements y = x, where is a linear projection. Traditional methods for CS solve a well-defined problem, e.g., $\min_x || x - y || + \lambda ||x||$, where $\lambda ||x||$ is the regularization, and employ iterative algorithms to solve it, e.g., the iterative shrinkage-thresholding algorithm (ISTA) [], with iteration $x_k = \mathcal{T}_{\lambda t}(x_k - t \ (x_k - y))$, 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 di erential 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 abovementioned 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 VDC-NNs [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.

🖌 n Z Y, et al. Sci China Inf Sci Jun o

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 di erential 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 [1] 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. [\mathbf{T}] proposed di erentiable architecture search (DARTS) and showed remarkable e ciency improvement. However, it is still unable to o er 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 e cient 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 di erential 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. [\mathbf{T}] 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 di erence 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. [\P] 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.

ot t on	D s r pt on	ot t on	D s r pt on
x_k	$\int output o t k t r$	W_k	w t ^{fl} tr
A	w the tr	•	un t on
\mathbb{S}^n	spos ⁿ (n [°] tr ⁿ trs	\mathbb{R}^{n}	$n \stackrel{\text{int}}{\longrightarrow} ns \text{ on } Eu = n sp$
\mathbb{S}^{n}_{++}	spopostv nt ^m trs	P x ́u	$P' $ π u – \bullet π u
A^{T}	trnspos o "tr v toru	H_k	ppro ^{fi} t nv rs H ss n ^{fi} tr
$\ \cdot\ _2$	nor	$\nabla F \mathcal{I}$ u	r nto $F x$
$ abla^2 F x$ u	H ss n ^{fl} tr o F x u	D ·	nrt on ^M tr
\mathcal{P}	pro ton op r tor	\mathcal{C}	$\mathcal{C} = \{x x \succeq \}$
Ι	nt t ^{III} , tr	U	$U = I - D$ / $Ax_k u$
r	$r = Ax_k u - x_k$	Q	$Q = U^{\mathrm{T}} U$
b	$b - U^{\mathrm{T}} r$	g_t	r nt
d_t	on utr_nt	α_t, β_t	s rs
N	nut roc G trtons o su	L	pt o t wton CG t

 Table 1
 ut fill r o not tons nt sppr

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

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

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 \times 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

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

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 $x_k = (Ax_k)$ solves the optimization problem

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

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) = -(x). *Proof.*

$$\nabla F(x) = A[x - (Ax)],\tag{1}$$

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

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





Figure 1 Cooron n'u 'u n t'u on ott n s t f o why two onvouton oprtons, s so non ott n s u Bo nourppr

$$= x_k - A^- [Ax_k - A (Ax_k)]$$

= (Ax_k). (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

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

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

$$[\nabla F(x)]^{-} = [I - \text{Diag}['(Ax)]A]^{-} A^{-}$$
$$= \int_{n^{-}}^{\infty} (\text{Diag}['(Ax)]A)^{n}A^{-}$$
$$= A^{-} + \int_{n^{-}}^{\infty} (\text{Diag}['(Ax)]A)^{n}A^{-} .$$
(7)

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

Obviously, the remaining term $\sum_{n=0}^{\infty} (\text{Diag}['(Ax)]A)^n A^-$ 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., $H_k = A^- + \sum_{n=0}^{m} (\text{Diag}['(Ax_k)]A)^n A^-$, where m is a given integer). As a result, the iteration scheme becomes

$$x_{k} = \mathcal{P}_{\mathcal{C}}[x_{k} - H_{k}\nabla F(x_{k})]$$

$$= [x_{k} + H_{k}A((Ax_{k}) - x_{k})]$$

$$= [(I - H_{k}A)x_{k} + H_{k}A((Ax_{k})], \qquad (8)$$

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:

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

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 [[]^{'u}.

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 structures⁴. 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 **C**.1. The iteration scheme of the Newton's method is as follows:

$$x_{k} = \mathcal{P}_{\mathcal{C}}\{x_{k} - [\nabla F(x_{k})]^{-} \nabla F(x_{k})\}$$

= $\{x_{k} + [I - \text{Diag}['(Ax_{k})]A]^{-} [(Ax_{k}) - x_{k}]\}.$ (10)

Noting that it is different compute the inverse $[I - \text{Diag}[\ '(Ax_k)]A]^-$ directly, we adopt the conjugate gradient (CG) method to compute it indirectly. We denote $U = I - \text{Diag}[\ '(Ax_k)]A$ 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

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

where

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

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

$$\min_{y} h(y) \equiv \frac{1}{-y} \quad Qy - b \quad y + \frac{1}{-r} \quad r. \tag{\mathbf{F}}$$

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 ovn to opt ton pro tuv to CGI to
Require: pr^{fl} trso t pro $l Q$ n b t null ro tr tons N
Ensure: so ut on o protein $u y$
$g_0 = \nabla h \ y_0 u = Qy_0 - b$
s t $d_0 - g_0$
for $t = 1, \dots, N - do$
$\alpha_t = -\frac{g_t^T d_t}{d_t^T Q d_t}$
$y_{t+1} = y_t - \alpha_t d_t$
$g_{t+1} = \nabla h \ y_{t+1}$ u $Qy_{t+1} - b$
$\beta_t = \frac{g_{t+1}^T Q d_t}{d_t^T Q d_t}$
$d_{t+1} - g_{t+1} = \beta_t d_t$
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 \mathbb{T}).

 $[\]underbrace{\operatorname{nt} t \operatorname{pro}_{\mathcal{A}} t \operatorname{on}_{\mathcal{A}} n^{\mathsf{fl}} \operatorname{ost} \operatorname{stu}_{\mathcal{A}} s t}_{\mathcal{A}} \operatorname{or}_{\mathcal{A}} n \quad \text{wor}$ uAtou Wsstrt s n y ow \mathbf{rn} t trt n t s n r pro, t on so wors n t s $\overset{\text{fit}}{\underset{\text{fit}}}$ or n r of $\overset{\text{fit}}{\underset{\text{fit}}}$ or $\overset{\text{fit}}{\underset{\text{fit}}}$ or $\overset{\text{fit}}{\underset{\text{fit}}}$ or $\overset{\text{fit}}{\underset{\text{fit}}}$ or p^{n n} tt r opt^{ri} propos 7 1 pot sst t t on orr spon to tt r n ur n twor r, t tur

n Z Y, et al. Sci China Inf Sci Jun o

t o	tt n	It r t on
	m	
us wton ⁴⁴ po	m	
	m	
	N	
	N –	
wton CG ^A to o	N	
1.	N	
	N	

Table 2 It r t on nu^{f1} rs o us n qu s wton^{f1} to s n wton CG^{f1} to s to so v to pro ^{f1} u

3.3 Numerical experiments

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



gen n A off w



Algorithm 2 \int , orw r prop ton 0, with CG B 0 , rol x_k to x_{k+1} u

 $[\]begin{array}{c} \textbf{Require:} \quad nput \ x_k \ t nut \ r \circ \ CG \ B \circ \ s \ N \ t \\ s \ rs \ \alpha_0, \alpha_1, \dots, \alpha_{N-1}, \beta_0, \beta_1, \dots, \beta_{N-2} \end{array} \quad \text{onvo ut on} \quad rn \ s \ W_t^{(1)}, W_t^{(2)}, W_t^{(3)}, \ n \ W_t^{(4)}, \ \leqslant t \leqslant N - \frac{t}{N} \\ y_0, -x_k \end{array}$

🖌 n Z Y, et al. Sci China Inf Sci Jun o

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 $_{onv}^{onv} \times$ 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 5×10^{-1} and 10^{-1} , 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 [-u]. We adopt batch normalization (BN) [-6] after each convolution kernel. Following [-1], we perform a linear projection to match the dimensions for addition operation whenever in need, with a 1×1 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-{, }, P, S-{5}, 56-{9,9,9}, and 8 -{1, 1, P}, and get Newton-CGNet-10, 16, R, 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, S, S, R, 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 [T], due to more training epochs.

We plot the results in Figure (a) and observe that when using a few CG Blocks ($N \leq 5$), Newton-CGNets perform worse than its counterpart ResNets. And when we use more CG Blocks ($N \geq 9$), Newton-CGNets perform better. To be specific, with comparable numbers of parameters, Newton-CGNet-56 and 8 surpass ResNet-110 and 10⁻ 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 7, when the learning rate is 0.1, our model performs well when the weight decay is between 10^- and 10^- . When the weight decay is $5 \times 10^-$, 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 $5 \times 10^-$ and 0.1, respectively.

u s^{ri}o, tonsnssr us wtonCGBo, snr nr^{ri}o of rws ^{ri}o, ton s^{ri}nor sposs nor rto^{ri}, nt nt rv s^{ri}



Table 3 st rror r t s Luon C us n s ts n wton CG ts L n o runsu

D tsprof wors n outp r of Figure 3 Co or on n⁴u⁴u wton CG w CG B o s n ts us n ts us n n or CG Bo s ✓u tr_n n urv s on C n Cn s not tr_n n rrors n so not t st rrors n s 7



Figure 4 Cooronn'u 'u st rror r t so, wton CG t why rnn r t o, n r nt w ft w'u 'ut st rror r t so, wton CG t why h w ft o, x 4 n r nt rnn r t s r su

Actually, since that the performance of a CNN model is dependent on multiple factors, such as datasets and training strategies, it is dicult 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 eccently, 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 T.

Newton-CGNets vs. advanced variants of ResNets. On the dataset with data augmentation, Newton-CGNet-8 results in (7.90%) 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-

y o	D pt	r ⁿ s	С	С	С	С	Н
st.							
s two stops p s two ropout s t pr tv tonu			-		-		-
o _ t .							
rt two ropout roppty.				4.60			
HB t .							-
ut.			-		-	-	-
wton CG t oursu			6.80		27.06		1.57
			-		-	23.44	-

Table 4 st rror r t s ~u on CIFA n H t s ts ^{fl} no runsu^a)

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 C.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 $(\mathbf{T} \times \mathbf{T})$, whereas the images in CIFAR and SVHN are only of size \times , directly employing that setting will result in very low resolution (1×1) after the last convolution. So we remove the first three downsampling and preserve the last two, in consistent with the setting of Newton-CGNets for fair comparison. As shown in Table **T**, 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. other optimization-inspired networks. Our method significantly outperforms ReduNet (1.66 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.

C.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

1 *	1 1 1	a	/	- /
t o	D pt	r ^{fM} s	op rror ^ u	op rror ^ u
s t				
s t				
wton CG t				
	L			
	Table 6 ′r s t	t t or t on t	stsus nour pr ^{fil} , nts	

Table 5 / top n top sn rop rror r ts ~uon 1/2 v tonsto ft t t st ^{fit} no runsu

	Table 6 'r s	t t	t or t on	t s ts us	n our	pr ¹ nts
Dtst	r _ n	$^{\mathrm{st}}$	C ss s	Av r	wor s	C t or tonts
AG n ws						En snwstorton
o ou n ws						g nsnws tor ton
DB						nto o ss. t on
Y prv wpo rt						nt ¹¹ , nt n ss
Yprvw, u						nt ¹¹ nt n ss
Y oo nsw rs						op ss. t on
A ^{nt} onry, w.u						nt ¹¹ , nt n ss
A ^{nt} on rywport						nt ¹¹ nt n ss

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

For ResNet- \mathbf{C} , the numbers of the residual blocks with di erent output sizes are \mathcal{K} , 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 $\overset{\times}{}_{\mathbf{X}}$, \times 6 are modified to the CG blocks \times , \mathbf{C} ; \times , \mathbf{C} ; $\overset{\times}{}_{\mathbf{X}}$, \mathbf{C} , composing a Newton-CG block. Correspondingly, the method of computing g_t is modified to

$$g_t = y_t + [g_t \overset{\mathbf{T}}{,} g_t \overset{\mathbf{T}}{,} g_t \overset{\mathbf{T}}{,} g_t \overset{\mathbf{T}}{,}], \tag{18}$$

where $[g_t \, {}^{\mathbf{u}}, g_t \, {}^{\mathbf{u}}, g_t \, {}^{\mathbf{u}}]$ 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 $W_t \, {}^{\mathbf{u}}, W_t \, {}^{\mathbf{u}}$, and $W_t \, {}^{\mathbf{u}}$ 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- \square are 5.98% and 8. %. With comparable numbers of parameters and the same depth, Newton-CGNet- \square surpasses ResNet- \square by 0.75% and $0\square$ % 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 a [8]

Table 7	st	rror r	\mathbf{t}	\mathbf{S}	¶u on	1	\mathbf{t}	\mathbf{s}	ts "		n oʻ		runsu ^{a)}
---------	---------------------	--------	--------------	--------------	-------	---	--------------	--------------	------	--	------	--	---------------------

|--|

a a	DC	₩Ż	out	7	ort	utu	÷	
-----	----	----	-----	---	----------------------	-----	---	--

DC with a ort utu

wton CG t oursu	7.87	3.31	0.98	4.11	35.41	26.28	36.94	3.92
^r u _d strsuts r _d d _t no A	^п о	s us ⁿ (poo, n	, or own	ns n p, n	tw n o	onvo ut on	rs

HZ nX n t D prsu rnn or ^{f14} ronton In ro nso IEEE Con rn on Co^{f4} put r son n ttrn onton -HunG uZ tn t D ns onnt onvouton ntwors In ro nso IEEE Con rn on

Co^{fil} put r son n tt rn ont on

Yn YZ on Zyn t Convouton nur ntworswyttrnt up t jqu In ro jnsoj IEEE Con, rn on Co^{ff} put r son n tt rn on ton

👖 Y Convouton nur ntwors, ors nt n 🛛 ss. ton In ro _n so Con, r n on 🖻 pr 👘 po s, n tur .

tur nu rossn -Z n X Z o J B. Cun Y G r tr v onvouton ntwors, ort t ss. ton In ronso t t Intrn ton Con, rn on ur In off ton rossn st^{at}s -Conn u A w n H B rr ut t r ponvouton ntwors, ort t ss. ton In ronso t t Con rn o t Europ n G ptrot Asso, ton, or Co⁴ put ton in ust s -Jo nson Z n D pp r⁴, onvouton nur ntwors, ort t t or ton In ronso t t Annu tn o t Asso, ton, or Co⁴ put ton in ust s -B r B Gunt t D s n n ur n twor rot t tur susn r nor ⁴ nt rnn In ronso

BrBGupt t Dsnnnur ntwor ret tursusnr, nor ant rnn In ro, nso Intrnton Con, rn on rnn prsnttons 'u C Zop B ann t rorssvnur ret turs re In ro, nso Europ nCon, rn on Co⁶ put r, son -

a HGun YZong Bt Entnur rest turs rev pratragrn In ronsong Intrnton Con, rn on Intrnn

LuH on n Yn YDrts, rnt ryt turs ry In ro, nso Intrnton Con, rn on rnn pr s nt t ons

G n X X X u J t rorssv rnt retturs reer n te pt p twns reen v u ton In ro n so IEEE C F Intrn ton Con rn on Co put r son -Gr or Cun Y rnn, st pprofit tons o sp rs o n In ro n so te Intrn ton Con rn on Intrn ton Con rn on en rnn -

Intrint on Con, in on gint in in -u n J A on r proton nur n twor, or nons⁴ oot opt⁴⁴, ton su, t to n r quits n oun onstrints IEEE r ns ur twit rn st -X n B n Y G o t ⁴⁴, sp rst why p n twors In roins of the Intrinton Con, r n on ur In of ton roiss n st⁴⁴ s -Y n Y un J H B t D p AD tor of prissives ns n I In roins of the Intrinton

Con, r n on ur In, of t on ross n st ${}^{\texttt{IA}}$ s -

Zn J Gn^{at} B I A t_ntrpr t opt^{at} ton_nsp r pn twor_or_^{at} o^t pr ss v s ns n In ro nso IEEE Con, rn on Co⁴ put r son n tt rn ont on

 $G \ r \ s \quad E \quad r \ Y \ C \quad Bronst_n \ A \qquad t \qquad r \quad o \ s \quad tw \ n \ onv \ r \quad n \ sp \qquad n \ r \ onstrut \ on \quad ur \quad n \ nv \ rs$ pro ^{fut}s IEEE r ns n ro ss

BA ou A, st.trtv grn traon ort^a, or nr nv rs pro^{nt}s IA J ft.

v onvouton sprs on J pn oⁿ no YE Convout on n ur n twors n

un X sr , r n D up ry s p sp rs o , n n twor s, or ^{fil} ss ton IEEE rnsf⁴ ross

Gn H YuY YouC t ut wit o prtwor, roll to prn.po, a in rtruton ÄrX v

HYnYGnDt pt^a, ton org^a, nspr pnur ntwor strutur sn In ro nsog gAsnConrn on n. rnn -rJD t DEg n. JC^a, ntonso, nt org^a, sn nur ntwors surv og st tog rt In ro nso, Intrn ton org op on C^a, ntonso, Gn t Aorg^a, sn ur twors nso Intraton or sopon Contintonso Gat Aorta sa ur twors un FHF.^{f.a.}H^{f.}n H t^f unn og strutur n pr^{fa} trso nur ntwor usn n^{fa} prov nt org^{fa} IEEE rns ur tw -

rnss^{''}HrussJGnrtvnurovouton, or prnn ArX v

Down prnnr J HuttrF p nup uto t prpr tropta, tono pnur ntwors trpo tono rnn urvs In ro nso to t Intrn ton Con rn on Art. Int n E A propos on n rn v na ssta s Colar un to t t -'u Y Z on A.' t B on nt rn ur ntwors r n prot turs n nu r rnt qu tons In ro nso to t Intrn ton Con rn on an rnn -H rE utotto. t r,t turs or pn ur ntwors Invrs ro Con u nov Y B tt nout I t ur or nr rnt cu tons In ro noo to n Cor ro

pr_ft turs n nu^fr r nt

Gen u novYBttnourtJt ur or, nr, rnt qutonsIn ro, nso, ty n Con, rn rational control c on ur In of t on rossn st

How r A G Z u G n B t o n ts nt onvout on n ur n twors or n o vs on pp t ons ArX v

r vs A H nton G⁺ rn n ut p⁺ rs o F tur s ro⁴ n f⁴ s n port CYX G r t D p sup rv s n ts In ro n s o y y Int rn t on Con, r n on Art.

Int, n n t t st s -t r Y n Co t s A t , n , ts n n tur , s w trunsup rv s , tur rn n In ro , n s o Con, r n on ur In of t on ross n st s

Hun G un Y u Z t D pn twors wy stop st py In rons o Europ n Con, rn on Cot put r s on -

Zoru o d⁴o s rsuntwors ArXv Dn J Don og r t f⁴ nt r s g r r g ⁴⁴ t s In ro nso IEEE Con r n on Cd⁴ put r son n tt rn onton -H Z n X n t I ntt ⁴⁴ pp n s n p r suntwors In ro nso Europ n Con, r n on

Con put r s on -

Gorot X B n, o Y n rst n, n ty ut o, tr, n n p, orw r nur n twors In ro n so, ty Int rn ton Con, r n on Art. Int n n t t st -Io C B ty nor⁴, ton r t n p n twor tr, n n r u, n, nt rn ov r t sy, t 1

ArX v

rsson G, r nrov GFr t n t utr pn ur n twor swy outrs u s In ro nso. Intrn ton Con, r n on rn pr s nt tons