# NEWTON RESIDUAL LEARNING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

A plethora of computer vision tasks, such as optical flow and image alignment, can be formulated as non-linear optimization problems. Before the resurgence of deep learning, the dominant family for solving such optimization problems was numerical optimization, e.g, Gauss-Newton (GN). More recently, several attempts were made to formulate learnable GN steps as cascade regression architectures. In this paper, we investigate recent machine learning architectures, such as deep neural networks with residual connections, under the above perspective. To this end, we first demonstrate how residual blocks (when considered as discretization of ODEs) can be viewed as GN steps. Then, we go a step further and propose a new residual block, that is reminiscent of Newton's method in numerical optimization and exhibits faster convergence. We thoroughly evaluate the proposed Newton-ResNet by conducting experiments on image/speech classification and image generation. All the experiments demonstrate that Newton-ResNet requires less parameters to achieve the same performance with the original ResNet.

## 1 INTRODUCTION

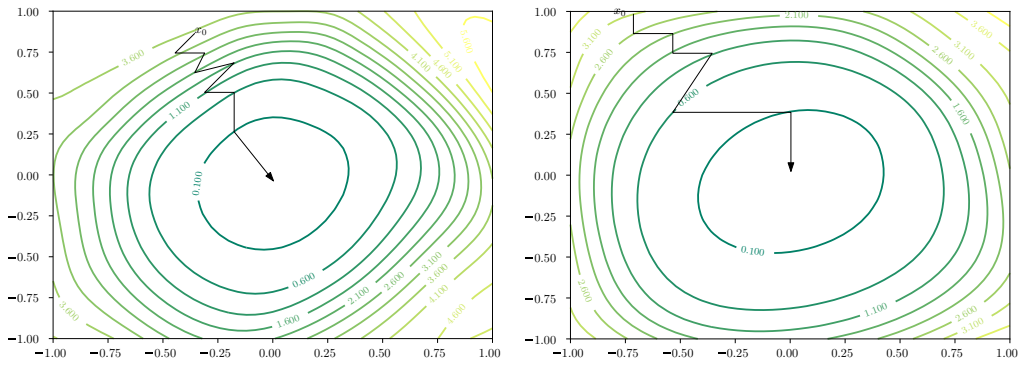
A wealth of computer vision problems (e.g., structure from motion (Buchanan & Fitzgibbon, 2005), stereo (Lucas et al., 1981; Clark et al., 2018), image alignment (Antonakos et al., 2015), optical flow (Zikic et al., 2010; Baker & Matthews, 2004; Rosman et al., 2011)) are posed as nonlinear optimization problems. Before the resurgence of the machine learning era, the dominant family for solving such optimization problems[1] was numerical optimization, e.g, Gauss-Newton (GN). Recently, it was proposed that the GN steps, called descent directions, can be learned and represented as a cascade regression to solve non-linear least square problems (Xiong & De la Torre, 2013). With the advent of deep learning, the aforementioned ideas were combined with learnable feature representations using deep convolutional neural networks for solving problems such as alignment and stereo (Trigeorgis et al., 2016; Clark et al., 2018). In this paper, we first try to draw similarities between learning descent directions and the structure of the popular residual networks. Motivated by that, we further extend residual learning by adopting ideas from Newton's numerical optimization method, which exhibits faster convergence rate than Gauss-Newton based methods (both theoretically and empirically as we show in our experiments).

ResNet (He et al., 2016) is among the most popular architectures for approximating non-linear functions through learning. The core component of ResNet is the residual block which can be seen as a linear difference equation. That is, the $t^{th}$ residual block is expressed as $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{C}\boldsymbol{x}_t$ for input $\boldsymbol{x}_t$. By considering the residual block as a discretization of Euler ODEs (Haber et al., 2018; Chen et al., 2018), each residual block expresses a learnable, first order descent direction.

We propose to accelerate the convergence (i.e., employ fewer residual blocks) in approximation of non-linear functions by introducing a novel residual block that exploits second-order information in analogy to Newton's method in non-linear optimization. Since the (second order) derivative is not analytically accessible, we rely on the idea of Xiong & De la Torre (2014) to learn the descent directions by exploiting second order information of the input. We build a deep model, called Newton-ResNet, that involves the proposed residual block. Newton-ResNet requires less residual blocks to achieve the same accuracy compared to original ResNet. This is depicted in Fig. 1; the

[1] We assume that the function we want to approximate is Lipschitz continuous and differentiable.

contour[2] shows the loss landscape near the minimum of each method and indeed the proposed method requires fewer steps.



(a) ResNet18 (b) Newton-ResNet

Figure 1: Contours around the minima of (a) ResNet18, (b) Newton-ResNet. Starting from a random point $\boldsymbol{x}_0$, ResNet needs more steps, i.e. more residual blocks, to reach the minimum.

Our contributions are as follows:

- We first establish a conceptual link between residual blocks in deep networks and standard optimization techniques, such as Gauss-Newton. This motivates us to design a novel residual block that learns the descent directions with second order information (akin to Newton steps in nonlinear optimization). A deep network composed of the proposed residual blocks is coined as Newton-ResNet.
- We show that Newton-ResNet can effectively approximate non-linear functions, by demonstrating that it requires less residual blocks and hence significantly less parameters to achieve the performance of the original ResNet. We experimentally verify our claim on four different datasets of images and speech in classification tasks. Additionally, we conduct experiments on image generation with Newton-ResNet-based GAN (Goodfellow et al., 2014).
- We empirically demonstrate that Newton-ResNet is a good function approximator even in the absence of activation functions, where the corresponding ResNet performs poorly.

## 2 Related work

The literature on resnets is vast; we focus below on the perspectives of a) theoretical understanding, b) alternative architectures and c) modifications of the transformation path.

A significant line of research is the theoretical understanding behind the performance of residual connections. The work of Hardt & Ma (2017) proves that arbitrarily deep linear residual networks have no spurious local optima; all critical points correspond to a global minimum. Zaeemzadeh et al. (2018) attribute the success of resnet to the norm presentation. More recently, Shamir (2018) proves that a network with residual connections is provably better than the corresponding network without the residuals. Balduzzi et al. (2017) focus on the gradients in residual connections; they study the correlations during initialization and introduce an appropriate initialization. These works are orthogonal to ours; they methodologically study the theoretical properties of deep learning, while we focus on reducing the number of residual blocks required.

The perspective of topological extension of the residual block is closer to our work. The popular densenet (Huang et al., 2017) uses dense connections which use concatenation instead of addition of the representation. Dual path networks of Chen et al. (2017) use a new block structure that includes both dense and residual connections. Wang et al. (2018) prove that resnet and densenet share the

[2]The method of Li et al. (2018); Im et al. (2016) is used to compute the contours.

same topology with different feature fusion method (addition and concatenation respectively). They propose a framework that generalizes both residual and dense connections.

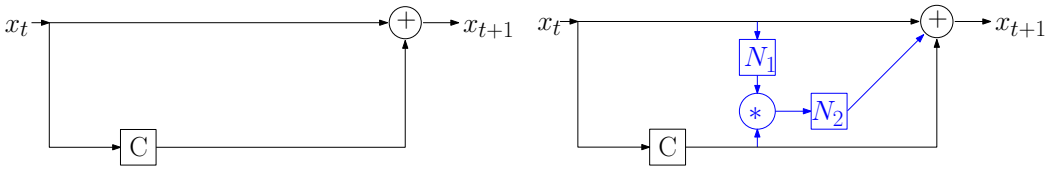
The work that is most closely related to ours is that of Srivastava et al. (2015); they define a topology that includes residual connections and higher order correlations. However, we offer a new perspective on the higher order correlations. In addition, we experiment with a) large scale problems, b) with linear blocks that highway networks have not used.

A popular line of research modifies the transformation path of each residual block. In ResNeXt (Xie et al., 2017) and Inception (Szegedy et al., 2017) the authors add group convolutions in the transformation path. In Zhang et al. (2017) the transformation path is a (set of) residual blocks itself, i.e. they obfuscate one residual block inside another. In wide residual networks (Zagoruyko & Komodakis, 2016) they advocate for increased width of each block. All related works are complementary to ours, since we do not modify the transformation path modules.

The applications of residual networks are diverse and often with impressive results. Such applications include object detection/recognition (Szegedy et al., 2017), face recognition (Deng et al., 2019), generative models (Miyato et al., 2018). However, these networks have tens or hundreds of residual blocks. A line of research that reduces the number of parameters is that of pruning the network (Han et al., 2015; Li et al., 2017; Chin et al., 2018). Han et al. (2015) propose to prune the weights with small magnitude, while Chin et al. (2018) propose a meta-learning technique to improve heuristic pruning techniques. However, pruning does not reduce the training resources (it even increases the time because for a single model, we train the network at least twice), and it is largely based on hand-engineered heuristics. That is, there is no solid understanding of the theoretical properties of pruning methods.

## 3 METHOD

To develop our intuition, we explore the linear residual block in sec. 3.1, i.e. the residual block without any activation functions. In sec. 3.2, we extend the proposed formulation in the presence of activation functions as typically used in ResNet.




(a) Original residual block (b) Our residual block

Figure 2: Schematic of the (a) original, (b) our residual block for the $t^{th}$ layer. The path that includes $\boldsymbol{C}$ is referred to as the transformation path; while the other (with the identity transformation) is referred to as the shortcut path. The symbol $\boldsymbol{C}$ denotes the operations in the transformation path, e.g. convolutions in He et al. (2016). The symbols $\boldsymbol{N}_1, \boldsymbol{N}_2$ are normalization layers, e.g. batch normalization or $1 \times 1$ convolutions.

### 3.1 LINEAR RESIDUAL BLOCK

Before the introduction of the residual block, all the neural networks were a composition of linear layers, e.g. fully-connected or convolutional layers, and activation functions. The (input) representation was transformed in each layer through a linear operation if we ignore the activation functions. The residual block of He et al. (2016) enables the input representation to pass through unchanged by introducing a two-pathway block consisting of a shortcut path and a transformation path. The $t^{th}$ residual block (in a network) is $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{C}\boldsymbol{x}_t$ for input $\boldsymbol{x}_t$[3]. That is, the residual block expresses a linear difference equation.

[3]Each residual block has different parameters $\boldsymbol{C}^{(t)}$, however we drop the dependence on $t$ for simplification.

We propose instead a new residual block that captures second order information. The new residual block is:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{C}\boldsymbol{x}_t + \alpha \boldsymbol{x}_t^T \boldsymbol{S}\boldsymbol{x}_t \tag{1}$$

for input $\boldsymbol{x}_t$ with $\boldsymbol{S}$ a square matrix operator. The scalar parameter $\alpha$ is learnable and plays the role of scaling the significance of the second order interactions. To reduce the number of parameters, we can share the parameters of $\boldsymbol{S}$ and $\boldsymbol{C}$; we introduce some normalization in the quadratic term. The proposed residual block then is expressed as:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \boldsymbol{C}\boldsymbol{x}_t + \alpha N_2(N_1(\boldsymbol{x}_t^T)\boldsymbol{C}\boldsymbol{x}_t) \tag{2}$$

with $N_1, N_2$ two normalization operators. The proposed residual block is depicted in Fig. 2.

### 3.2 Residual block with activation functions

Frequently activation functions are used in conjunction with the residual block. We consider a residual block with activation functions and two convolutions in the transformation path. If we define the function $\boldsymbol{f}_t(\boldsymbol{x}) = \boldsymbol{C}_2(\phi(\boldsymbol{C}_1\boldsymbol{x}))$, the residual block is $\boldsymbol{x}_{t+1} = \phi(\boldsymbol{x}_t + \boldsymbol{f}_t(\boldsymbol{x}_t))$ with $\phi$ denoting an activation function, such as RELU. To avoid cluttering the notation, batch normalization is ignored in the last equation.

The proposed residual block in the presence of activation functions becomes:

$$\boldsymbol{x}_{t+1} = \phi(\boldsymbol{x}_t + \boldsymbol{f}_t(\boldsymbol{x}_t) + \alpha N_2(N_1(\boldsymbol{x}_t^T)\boldsymbol{f}_t(\boldsymbol{x}_t))) \tag{3}$$

The proposed residual block can be used with different architectures, e.g. three convolutions or with group convolutions.

## 4 Experiments

**Implementation details**: All the optimization-related hyper-parameters, e.g. the optimizer, the learning rate, the initializations, the number of epochs etc., remain the same as in the original ResNet. Further improvement can be obtained by tuning those values for our residual block, but this is out of our scope. Unless mentioned otherwise, each experiment is conducted **5 times** and the average and the standard deviation are reported.

**Datasets**: The following four datasets are used in this work:

1. *CIFAR10* (Krizhevsky et al., 2014): This is a widely used dataset that contains $60,000$ images of natural scenes. Each image is of resolution $32 \times 32 \times 3$ and is classified in one of the $10$ classes.
2. *CIFAR100* (Krizhevsky et al.): This is an extension over CIFAR10; it includes the same amount of images but there are $100$ classes.
3. *ImageNet* (Russakovsky et al., 2015): The ImageNet 2012 dataset (Russakovsky et al., 2015) comprises $1.28$ million training images and $50$K validation images from $1000$ different classes. We train networks on the training set and report the top-1 and top-5 error on the validation set.
4. *Speech Commands* (Warden, 2018): This newly released dataset contains $60,000$ audio files; each audio contains a single word of a duration of one second. There are $35$ different words (classes) with each word having $1,500 - 4,100$ recordings. Every audio file is converted into a mel-spectrogram of resolution $32 \times 32$.

Below, we conduct an experiment on image classification on CIFAR10 in sec. 4.1; we modify the experiment by removing the activation functions, i.e. have networks linear with respect to the weights, in sec. 4.2. Sequentially, image classification experiments on CIFAR100 and ImageNet are conducted in sec. 4.3 and sec. 4.4 respectively. In addition to the image classification experiments, we exhibit how the proposed residual block can be used on image generation in sec. 4.5. Furthermore, an experiment on audio classification is conducted in sec. 4.6.

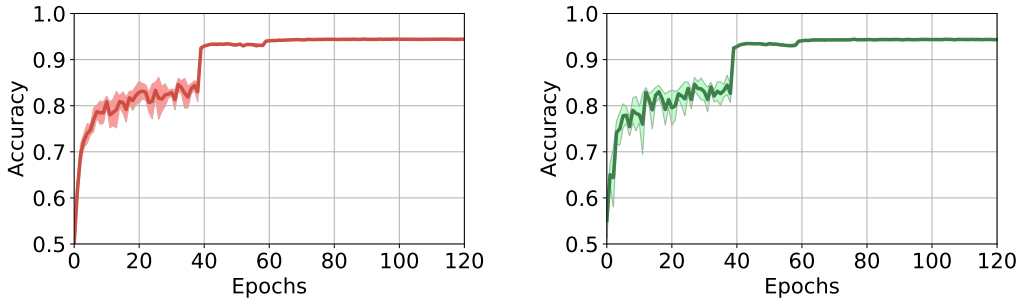
### 4.1 RESNET CLASSIFICATION ON CIFAR10

We utilize CIFAR10 as a popular dataset for classification. We train each method for $120$ epochs with batch size $128$. The SGD optimizer is used with initial learning rate of $0.1$. The learning rate is multiplied with a factor of $0.1$ in epochs $40, 60, 80, 100$. We use two ResNet architectures, i.e. ResNet18 and ResNet34, as baselines. Our model, called Newton-ResNet, is built with the proposed residual blocks; we add enough blocks to match the performance of the respective baseline.

In Table 1 the two different ResNet baselines are compared against Newton-ResNet; the respective Newton-ResNet models have the same accuracy. However, each Newton-ResNet has $\sim 40\%$ less parameters than the respective baseline. In addition, we visualize the test accuracy for ResNet18 and the respective Newton-ResNet in Fig. 3. The test error of the two models is similar throughout the training; a similar phenomenon is observed for ResNet34 in Fig. 6.

Table 1: Image classification on CIFAR10 with ResNet. The # abbreviates 'number of', while the parameters are measured in millions. The term 'block' abbreviates a 'residual block'. Note that each baseline, e.g. ResNet18, has the same performance with the respective Newton-ResNet, but significantly more parameters.

| CIFAR10 classification with ResNet | | | |
|---|---|---|---|
| Model | # blocks | # params (M) | Accuracy |
| ResNet18 | $[2, 2, 2, 2]$ | 11.2 | $0.945 \pm 0.000$ |
| Newton-ResNet | $[2, 2, 1, 1]$ | **6.0** | $0.945 \pm 0.001$ |
| ResNet34 | $[3, 4, 6, 3]$ | 21.3 | $0.948 \pm 0.001$ |
| Newton-ResNet | $[3, 3, 2, 2]$ | **13.0** | $0.949 \pm 0.002$ |




(a) ResNet18

(b) Newton-ResNet

Figure 3: The test accuracy of (a) ResNet18 and (b) the respective Newton-ResNet are plotted (CIFAR10 training). The two models perform similarly throughout the training, while ours has $46\%$ less parameters. The width of the highlighted region denotes the standard deviation of each model over $5$ runs.

### 4.2 RESNET WITH LINEAR BLOCKS ON CIFAR10

We remove all the activation functions, both from the transformation path and the output activation functions. The rest of the settings remain the same as in sec. 4.1.

As can be noticed in Table 2, Newton-ResNet outperforms ResNet18 by a significant margin when removing the activation functions. It is worth noting that the performance of Newton-ResNet with/without activation functions differs by $7\%$, i.e. decent performance can be obtained without any activation functions.

Table 2: In this experiment all activation functions are removed. The performance of ResNet18 drops dramatically by $58\%$ when we remove the activation functions. On the contrary, the performance of Newton-ResNet is decreased by less than $7\%$ (in comparison to Table 3).

| CIFAR10 classification with ResNet | | | |
|---|---|---|---|
| Model | # blocks | # params (M) | Accuracy |
| ResNet18 | $[2, 2, 2, 2]$ | 11.2 | $0.391 \pm 0.001$ |
| Newton-ResNet | $[1, 1, 1, 1]$ | **5.3** | $0.876 \pm 0.001$ |
| Newton-ResNet | $[2, 2, 2, 2]$ | 11.9 | $0.908 \pm 0.002$ |
| Newton-ResNet | $[3, 3, 3, 3]$ | 18.5 | $0.916 \pm 0.002$ |

**Algorithm 1:** Our residual block

**function** $res\_block(\boldsymbol{x}_t, n\_convs, x\_proj, lin\_proj, \phi, \phi_2)$
  $\boldsymbol{o} \leftarrow \boldsymbol{x}_t$
  **for** *i=1:n_convs* **do**
    $\boldsymbol{o} \leftarrow \phi(\text{BN}(\text{conv}(\boldsymbol{o})))$;
  **end**
  **for** *i=1:x_proj* **do**
    $\boldsymbol{x}_t \leftarrow \phi_2(\text{conv}_{1\times1}(\boldsymbol{x}_t))$;
  **end**
  $\boldsymbol{s} \leftarrow \boldsymbol{x}_t * \boldsymbol{o}$
  **for** *i=1:lin_proj* **do**
    $\boldsymbol{s} \leftarrow \phi_2(\text{norm}(\text{conv}_{1\times1}(\boldsymbol{s})))$;
  **end**
  $\boldsymbol{x}_{t+1} \leftarrow \boldsymbol{x}_t + \boldsymbol{o} + \boldsymbol{s}$
  return $\boldsymbol{x}_{t+1}$
**end function**

Table 3: The differences of the proposed method with the original residual block are highlighted in blue. The $x\_proj, lin\_proj$ are $1 \times 1$ convolutions added for normalization purposes in the proposed residual block.

### 4.3 Classification on CIFAR100

We verify the aforementioned classification results on CIFAR100. The settings remain the same as in sec. 4.1. As can be noticed in Table 4 the test accuracy of ResNet34 and Newton-ResNet is similar, however Newton-ResNet has $\sim 44\%$ less parameters.

The experiment of sec. 4.2 with the linear blocks is repeated on CIFAR100. That is, we remove all the activation functions and train the networks. The accuracy of each method is reported on Table 5. The difference observed in sec. 4.1 becomes even more pronounced. That is, ResNet performs poorly in this case and is substantially outperformed by Newton-ResNet.

Table 4: CIFAR100 classification with ResNet. The accuracy of the compared methods is similar, but Newton-ResNet has $30\%$ less parameters.

| CIFAR100 classification with ResNet | | | |
|---|---|---|---|
| Model | # blocks | # params (M) | Accuracy |
| ResNet34 | $[3, 4, 6, 3]$ | 21.3 | $0.769 \pm 0.003$ |
| Newton-ResNet | $[3, 4, 3, 2]$ | **14.7** | $0.769 \pm 0.001$ |

### 4.4 Classification on ImageNet

We perform a large-scale classification experiment on ImageNet; due to the computational resources required, this experiment is conducted only once. Following standard practices, we utilize the following data augmentation techniques: (1) normalization through mean RGB-channel subtraction, (2) random resized crop to $224 \times 224$, (3) scale from $5\%$ to $100\%$, (4) aspect ratio from $\frac{3}{4}$ to $\frac{4}{3}$, and

Table 5: Experiment on CIFAR100 by removing all activation functions.

| CIFAR100 classification with linear ResNet | | | |
|---|---|---|---|
| Model | # blocks | # params (M) | Accuracy |
| ResNet18 | $[2, 2, 2, 2]$ | 11.2 | $0.168 \pm 0.001$ |
| Newton-ResNet | $[1, 1, 1, 1]$ | **5.3** | $0.620 \pm 0.001$ |
| Newton-ResNet | $[2, 2, 2, 2]$ | 11.9 | $0.670 \pm 0.002$ |
| Newton-ResNet | $[3, 3, 3, 3]$ | 18.5 | $0.689 \pm 0.002$ |

(5) random horizontal flip. During inference, we perform the following augmentation techniques: (1) normalization through mean RGB-channel subtraction, (2) scale to $256 \times 256$, and (3) single center crop to $224 \times 224$.

All models are trained on a DGX station with four Tesla V100 (32GB) GPUs. We use Mxnet [4] and choose float16 instead of float32 to achieve $3.5\times$ acceleration and half the GPU memory consumption. In our preliminary experiments, we noticed that the second order might cause numeric overflow in float16; this was not observed in the rest of the experiments that use float32. Hence, we use a $\tanh$ as a normalization for the second order term, i.e. the last term of (3). Optimization is performed using SGD with momentum $0.9$, weight decay $1e-4$ and a mini-batch size of $1024$. The initial learning rate is set to $0.4$ and decreased by a factor of $10$ at $30, 60$, and $80$ epochs. Models are trained for $90$ epochs from scratch, using linear warm-up of the learning rate during first five epochs according to Goyal et al. (2017). For other batch sizes due to the limitation of GPU memory, we linearly scale the learning rate (e.g. learning rate 0.1 for batch size 256).

We report both the Top-1 and Top-5 single-crop validation error in Table 6. For a fair comparison, we report the results from our training in both the original ResNet and Newton-ResNet[5]. Newton-ResNet consistently improves the performance with an extremely small increase in computational complexity and model size. Remarkably, Newton-ResNet50 achieves a single-crop Top-5 validation error of $6.358\%$, exceeding ResNet50 (6.838%) by 0.48% and approaching the performance achieved by the much deeper ResNet101 network (6.068% Top-5 error). The loss and Top-1 error throughout the training are visualized in Fig. 4, which demonstrates that the proposed method performs favourably to the baseline ResNetwhen the same amount of residual blocks are used.

Table 6: Image classification (ImageNet) with ResNet. The column of "Speed" refers to the inference speed (images/s) of each method.

| ImageNet classification with ResNet | | | | | |
|---|---|---|---|---|---|
| Model | # Blocks | Top-1 error (%) | Top-5 error (%) | Speed | Model Size |
| ResNet50 | $[3, 4, 6, 3]$ | 23.570 | 6.838 | 8.5K | 50.26 MB |
| Newton-ResNet50 | $[3, 4, 6, 3]$ | 22.875 | 6.358 | 7.5K | 68.81 MB |
| ResNet101 | $[3, 4, 23, 3]$ | 22.208 | 6.068 | 6.2K | 87.67 MB |

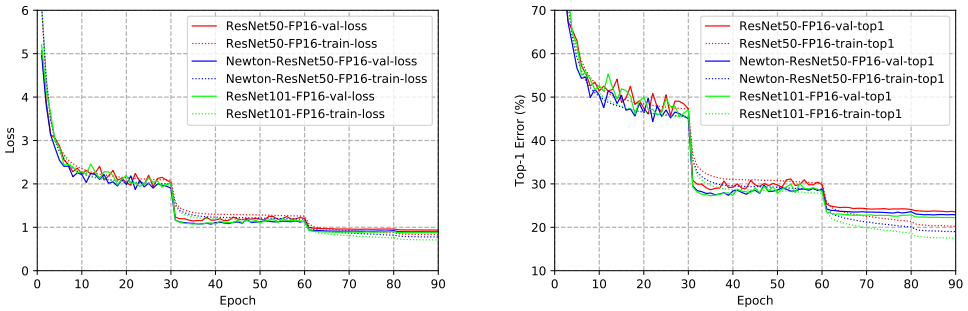
In Table 7, we remove all the "relu" activation functions for both baseline and the proposed method. Without "relu", Newton-ResNet50 achieves a single-crop Top-5 validation error of $9.114\%$, significantly exceeding ResNet50 (71.562%) and approaching the performance achieved by the "relu" version (6.068% Top-5 error).

Table 7: Image classification (ImageNet) without "relu". Newton-ResNet outperforms the corresponding ResNet by a substantial margin.

| ImageNet classification without "relu" | | | |
|---|---|---|---|
| Model | # Blocks | Top-1 error (%) | Top-5 error (%) |
| ResNet50 | $[3, 4, 6, 3]$ | 85.124 | 71.562 |
| Newton-ResNet50 | $[3, 4, 6, 3]$ | 27.292 | 9.114 |

[4] https://github.com/NVIDIA/DeepLearningExamples/tree/master/MxNet/Classification/RN50v1.5

[5] The reported performance for the original ResNet is superior to the one reported in He et al. (2016); the re-implementation of Hu et al. (2018) reports similar results to ours.




(a) Loss (b) Top 1 Error

Figure 4: Baseline ResNetarchitectures (ResNet50 and ResNet101) and the Newton counterpart (Newton-ResNet50) on the ImageNet validation dataset. The proposed method produces consistent gains in the performance which are sustained throughout the training process.

### 4.5 RESNET IN GENERATIVE MODELS

Deep discriminative networks include hundreds of layers, while their generative counterparts' depth is critical and restricted (mainly because they are hard to train and fit in existing hardware). We explore whether we can reduce the number of residual blocks in generative models. Generative Adversarial Networks (GAN) of Goodfellow et al. (2014) have dominated the related literature (Miyato et al., 2018) due to their impressive visual results.

GANs include two modules, a generator and a discriminator, which are both implemented with resnet-based neural networks. The generator samples $\boldsymbol{z}$ from a prior distribution, e.g. uniform, and tries to model the target distribution; the discriminator tries to distinguish between the samples synthesized from the generator and the target distribution. GAN is typically optimized with an alternating gradient descent method.

We select the architecture of Miyato et al. (2018) (SNGAN) as a strong baseline on CIFAR10. The baseline includes 3 resnet blocks in the generator and 3 in the discriminator. We replace the original residual blocks with the proposed residual blocks; two such blocks in each module suffice to achieve the same performance. That is, we reduce by 1 the blocks in both the generator and the discriminator. In Table 8 the experimental result is summarized. Note that the experiment is conducted 10 times and the mean and variance are reported. In Fig. 5 some random samples synthesized by the two methods are depicted; visually the generated samples are similar.

Table 8: Quantitative results on image generation with resnet-based generator and discriminator. 'Ours' denotes a modified SNGAN model; specifically we replace the residual blocks with the proposed ones. The Inception Score (IS) and the Frechet Inception Distance (FID) are standard metrics in the GAN literature; higher IS/lower FID imply a superior performance. '# G params' abbreviates the number of generator parameters, while '# D params' the number of discriminator parameters. The parameters are measured in millions.

| Model | # G params | # D params | IS | FID |
|---|---|---|---|---|
| SNGAN | 4.28 | 1.05 | $8.06 \pm 0.10$ | $19.06 \pm 0.50$ |
| Ours | **3.56** | **0.76** | $8.03 \pm 0.11$ | $19.03 \pm 0.39$ |

### 4.6 CLASSIFICATION ON SPEECH COMMANDS

We evaluate the performance of ResNet on the Speech Commands dataset. This dataset has a different distribution from the natural images that ResNet is typically applied to. We train each method for 70 epochs with SGD and initial learning rate of 0.01. The learning rate is reduced if the validation accuracy does not improve for two epochs. We use ResNet34 as a baseline architecture; we build respectively a network that matches the performance of the baseline.

(a) GT (b) SNGAN (c) Ours

Figure 5: Synthesized samples from GAN (sec. 4.5). It can be visually verified that both the SNGAN and 'Ours' (with the proposed residual blocks) result in similar visual samples.

The quantitative results are added in Table 9. The two models share the same accuracy, however Newton-ResNet includes $38\%$ less parameters. This is consistent with the experiments on classical image datasets, i.e. sec. 4.1.

Table 9: Speech classification with ResNet. The accuracy of the compared methods is similar, but Newton-ResNet has $38\%$ less parameters.

| Speech Commands classification with ResNet | | | |
|---|---|---|---|
| Model | # blocks | # params (M) | Accuracy |
| ResNet34 | $[3, 4, 6, 3]$ | 21.3 | $0.951 \pm 0.002$ |
| Newton-ResNet | $[3, 3, 3, 2]$ | **13.2** | $0.951 \pm 0.002$ |

## 5 Conclusion

In this work, we establish a link between the residual blocks of ResNet architectures and learning decent directions in solving non-linear least squares (e.g., each block can be considered as a decent direction). We exploit this link and we propose a novel residual block that uses second order interactions as reminiscent of Newton's numerical optimization method (i.e., learning Newton-like descent directions). Newton-type methods are likely to converge faster than first order methods (e.g., Gauss-Newton). We demonstrate that in the proposed architecture this translates to less residual blocks (i.e., less decent directions) in the network for achieving the same performance. We conduct validation experiments on image and audio classification with residual networks and verify our intuition. Furthermore, we illustrate that with our block it is possible to remove the non-linear activation functions and still achieve competitive performance.

## References

Epameinondas Antonakos, Joan Alabort-i Medina, and Stefanos Zafeiriou. Active pictorial structures. In *IEEE Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5435–5444, 2015.

Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV)*, 56(3):221–255, 2004.

David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *International Conference on Machine Learning (ICML)*, pp. 342–350, 2017.

Aeron M Buchanan and Andrew W Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *IEEE Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pp. 316–322, 2005.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems (NeurIPS)*, pp. 6571–6583, 2018.

Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. In *Advances in neural information processing systems (NeurIPS)*, pp. 4467–4475, 2017.

Ting-Wu Chin, Cha Zhang, and Diana Marculescu. Layer-compensated pruning for resource-constrained convolutional neural networks. *arXiv preprint arXiv:1810.00518*, 2018.

Ronald Clark, Michael Bloesch, Jan Czarnowski, Stefan Leutenegger, and Andrew J Davison. Ls-net: Learning to solve nonlinear least squares for monocular stereo. *arXiv preprint arXiv:1809.02966*, 2018.

Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *IEEE Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4690–4699, 2019.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems (NeurIPS)*, 2014.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv:1706.02677*, 2017.

Eldad Haber, Lars Ruthotto, Elliot Holtham, and Seong-Hwan Jun. Learning across scales-multiscale methods for convolution neural networks. In *AAAI Conference on Artificial Intelligence*, 2018.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems (NeurIPS)*, pp. 1135–1143, 2015.

Moritz Hardt and Tengyu Ma. Identity matters in deep learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pp. 7132–7141, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4700–4708, 2017.

Daniel Jiwoong Im, Michael Tao, and Kristin Branson. An empirical analysis of deep network loss surfaces. *arXiv:1612.04010*, 2016.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research). URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55, 2014.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations (ICLR)*, 2017.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in neural information processing systems (NeurIPS)*, pp. 6389–6399, 2018.

Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Guy Rosman, Shachar Shem-Tov, David Bitton, Tal Nir, Gilad Adiv, Ron Kimmel, Arie Feuer, and Alfred M Bruckstein. Over-parameterized optical flow using a stereoscopic constraint. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pp. 761–772. Springer, 2011.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

Ohad Shamir. Are resnets provably better than linear predictors? In *Advances in neural information processing systems (NeurIPS)*, pp. 507–516, 2018.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI Conference on Artificial Intelligence*, 2017.

George Trigeorgis, Patrick Snape, Mihalis A Nicolaou, Epameinondas Antonakos, and Stefanos Zafeiriou. Mnemonic descent method: A recurrent process applied for end-to-end face alignment. In *IEEE Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4177–4187, 2016.

Wenhai Wang, Xiang Li, Jian Yang, and Tong Lu. Mixed link networks. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *IEEE Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 532–539, 2013.

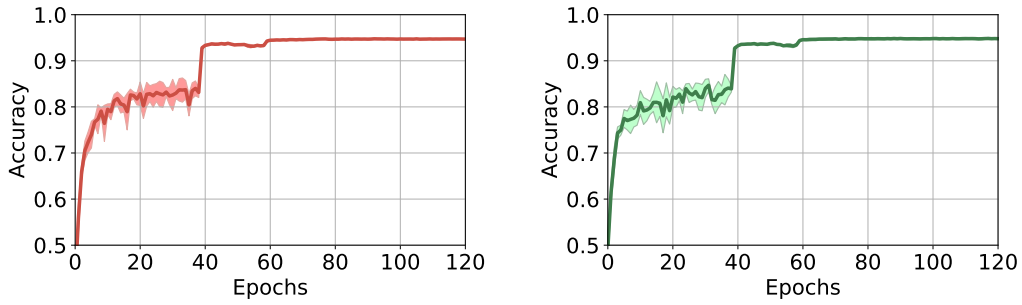
Xuehan Xiong and Fernando De la Torre. Supervised descent method for solving nonlinear least squares problems in computer vision. *arXiv preprint arXiv:1405.0601*, 2014.

Alireza Zaeemzadeh, Nazanin Rahnavard, and Mubarak Shah. Norm-preservation: Why residual networks can become extremely deep? *arXiv preprint arXiv:1805.07477*, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

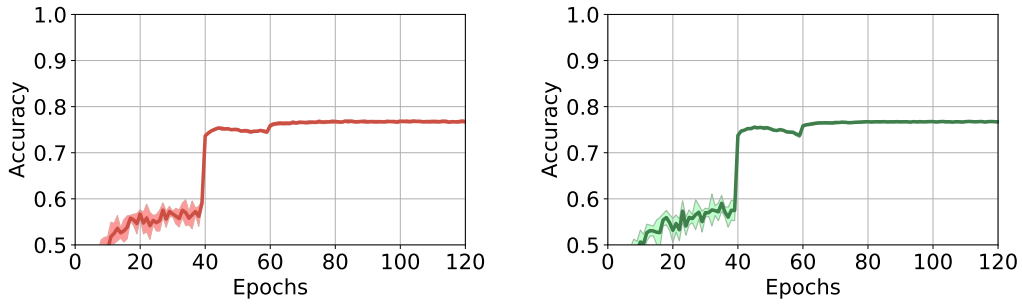
Ke Zhang, Miao Sun, Tony X Han, Xingfang Yuan, Liru Guo, and Tao Liu. Residual networks of residual networks: Multilevel residual networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(6):1303–1314, 2017.

Darko Zikic, Ali Kamen, and Nassir Navab. Revisiting horn and schunck: Interpretation as gauss-newton optimisation. In *BMVC*, pp. 1–12, 2010.



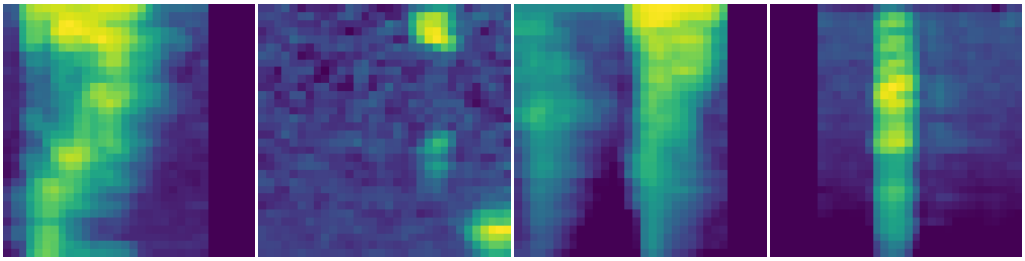

(a) ResNet34 (b) Newton-ResNet

Figure 6: The test accuracy of (a) ResNet34 and (b) the respective Newton-ResNet are plotted (CIFAR10 training; sec. 4.3). The two models perform similarly throughout the training. The width of the highlighted region denotes the standard deviation of each model over 5 runs.




(a) ResNet34 (b) Newton-ResNet

Figure 7: The test accuracy of (a) ResNet34 and (b) the respective Newton-ResNet are plotted (CIFAR100 training; sec. 4.3). The two models perform similarly throughout the training. The width of the highlighted region denotes the standard deviation of each model over 5 runs.



(a) 'Zero' (b) 'Yes' (c) 'No' (d) 'Up'

Figure 8: Four input mel-spectrograms; each image is used as input in the speech classification experiment in sec. 4.6. The label is the corresponding word for each image.

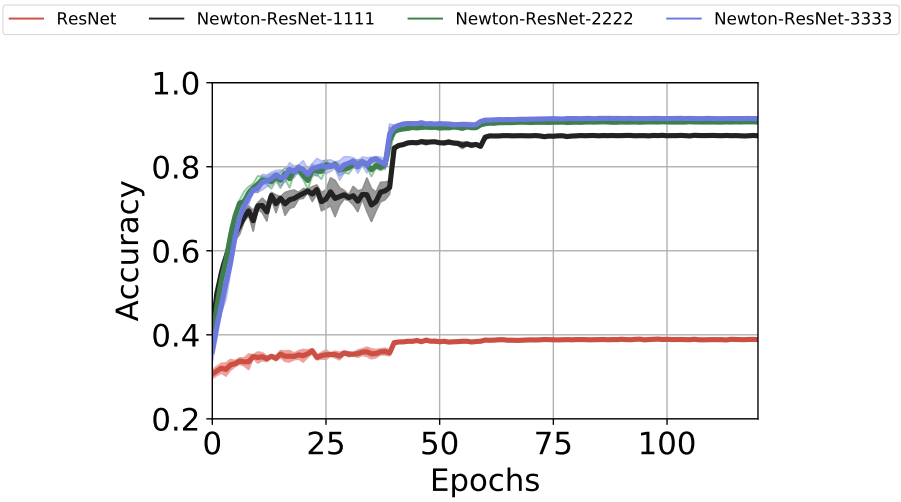



Figure 9: Training accuracy in the models without activation functions (further details in sec. 4.2).