Journal of Computational Mathematics
Vol.39, No.1, 2021, 147–158.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1912-m2018-0279

# CONVERGENCE OF BACKPROPAGATION WITH MOMENTUM FOR NETWORK ARCHITECTURES WITH SKIP CONNECTIONS*

Chirag Agarwal
*Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607, USA*
*Email: chiragagarwall12@gmail.com*
Joe Klobusicky
*Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180, USA*
*Email: klobuj@rpi.edu*
Dan Schonfeld
*Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607, USA*
*Email: dans@uic.edu*

**Abstract**

We study a class of deep neural networks with architectures that form a directed acyclic graph (DAG). For backpropagation defined by gradient descent with adaptive momentum, we show weights converge for a large class of nonlinear activation functions. The proof generalizes the results of Wu et al. (2008) who showed convergence for a feed-forward network with one hidden layer. For an example of the effectiveness of DAG architectures, we describe an example of compression through an AutoEncoder, and compare against sequential feed-forward networks under several metrics.

*Mathematics subject classification:* 68M07, 68T01.
*Key words:* Backpropagation with momentum, Autoencoders, Directed acyclic graphs.

## 1. Introduction

Neural networks have recently enjoyed an acceleration in popularity, with new research adding to several decades of foundational work. From multilayer perceptron (MLP) networks to the more prominent recurrent neural networks (RNNs) and convolutional neural networks (CNNs), neural networks have become a dominant force in the fields of computer vision, speech recognition, and machine translation [11]. Increase in computational speed and data collection have legitimized the training of increasingly complex deep networks. The flow of information from input to output is typically performed in a strictly sequential feed-forward fashion, in which for a network consisting of $L$ layers, nodes in the $i$th layer receive input from the $(i-1)$st layer, compute an output for each neuron through an activation function, and in turn use this output as an input for the $(i+1)$st layer. A natural extension to this network structure is the addition of "skip connections" between layers. Specifically, we are interested in the class of architectures in which the network of connections form a directed acyclic graph (DAG). The defining property of a DAG is that it can always be decomposed into a *topological ordering* of $L$ layers, in which nodes in layer $i$ may be connected to layer $j$, where $j > i$. A skip connections is a connection

* Received December 18, 2018 / Revised version received August 4, 2019 / Accepted December 9, 2019 / Published online August 30, 2020 /

between nodes in layers $i$ and $j$, with $j > i+1$. There has been an increasing interest in studying networks with skip connections which skip a small number of layers, with examples including Deep Residual Networks (ResNet) [5], Highway Networks [13], and FractalNets [8]. ResNets, for instance, use "shortcut connections" in which a copy of previous layers is mapped through an identity mapping to future layers. Kothari and Agyepong [7] introduced "lateral connections" in the form of a chain, with each unit in a hidden layer connected to the next. The full generality of neural networks for DAG architectures was considered in [6], which demonstrated superior performance of neural networks, entitled DenseNets, under a wide variety of skip-connections.

As an example of the efficacy of DAG architectures considered in [6], we consider AutoEncoders, a class of neural networks which provide a means of data compression. For an AutoEncoder, input data, such as a pixelated image, is also the desired output for a neural network. During an encoding phase, input is compressed through several hidden layers before arriving at a middle hidden layer, called the code, having dimension smaller than the input. The next phase is decoding, in which input from the code is fed through several more hidden layers until arriving at the output, which is of the same dimension as the input. The goal of compression is to minimize the difference between input data and output. In [1], Agarwal et al. introduced CrossEncoders and demonstrated its superior performance against AutoEncoders with no skip-connections. In Section 3, we extend the previous results to include the MNIST and Olivetti faces public datasets. We validate our results against several commonly used compression based performance metrics.

Our main theoretical result is the convergence of backpropagation with DAG architectures using gradient descent with momentum. It is well known that feed-forward architectures converge under backpropagation, which is essentially gradient descent applied to an error function (see [3], for instance). Updates for weights in backpropagation may be generalized to include a momentum term, which can help with increasing the convergence rate [12]. Momentum can help with escaping local minima, but concerns of overshooting require careful arguments for establishing convergence. Formal arguments for convergence have so far been restricted to simple classes of neural networks. Bhaya [2] and Torii [15] studied the convergence with backpropagation using momentum under a linear activation function. Zhang et al. [17] generalized convergence for a class of common nonlinear activation functions, including sigmoids, for the case of a zero hidden layer networks. Wu et al. [16] further generalized to one layer by demonstrating that error is monotonically decreasing under backpropagation iterations for sufficiently small momentum terms. The addition of a hidden layer required [16] to make the additional assumption of bounded weights during the iteration procedure.

It is not evident whether applying the methods of [16] would generalize to networks with several hidden layers and skip connections, or if they would require stronger assumptions on boundedness of weights or the class of activation functions. We show in Section 4 that convergence indeed does hold, with similar assumptions to the proof of convergence of one hidden layer. In Theorem 4.1, we give the key inequality for proving Theorem 2.1, a recursive form for increments of error and output values of hidden layers after each iteration. This estimate allows us to show that for sufficiently small momentum parameters (including the case of zero momentum), error decreases with each iteration. Our approach to convergence is somewhat more explicit than the traditional proof of gradient descent, which minimizes a loss function without considering network architecture.

## 2. Architecture for a Feed-Forward Network with Cross-Layer Connectivity

In this section, we formally explain DAG architectures, and the associated backpropagation algorithm with momentum. We then state a theorem for the convergence of error through backpropagation, whose proof is presented in Section 4.

### 2.1. DAG architecture and backpropagation

We now present the architecture for neural networks on DAGs. Nodes of a DAG can always be ordered into layers $0, \ldots, L$, in which connections (or directed edges) point to layers labeled with higher indices. We will consider $J$ input values $x^p \in \mathbb{R}^{l_0}$, $p = 1, \ldots, J$. For each layer $i = 0, \ldots, L$, there are $l_i$ nodes, with layer 0 denoting the input. Under this ordering, define $v_{(i,j)}^{l,m}$ as the weight between node $l$ in layer $i$ and node $m$ in layer $j$, where $i < j$. Let $v_{(i,j)}$ denote the matrix of weights from layer $i$ to $j$. Over all nodes, we use a single (possibly nonlinear) activation function $g : \mathbb{R} \to \mathbb{R}$ for the determination of output values.

The explicit output values of the $l_j$ nodes in layer $j$ are denoted as

$$H_j = (H_j^1, \ldots, H_j^{l_j}), \quad 0 \le j \le L. \tag{2.1}$$

These are defined recursively from forward propagation, where the $j$th layer receives input from all layers $H_i$ with $i < j$. Explicitly,

$$H_0 = x, \qquad H_1 = g\left(H_0 v_{(0,1)}\right), \tag{2.2}$$

$$H_j = g\left(\sum_{i<j} H_i v_{(i,j)}\right), \qquad H_L = y = g\left(\sum_{i<L} H_i v_{(i,L)}\right). \tag{2.3}$$

Note that here and in the future, for a real valued function $f$, and a vector $v = (v_1, \ldots, v_n)$, we will use the notation $f(v) = (f(v_1), \ldots, f(v_n))$. Node inputs are defined as

$$S_j = \sum_{i<j} H_i v_{(i,j)}. \tag{2.4}$$

We seek to minimize the difference between a set of $J$ desired outputs $d^1, \ldots, d^J \in \mathbb{R}^{l_L}$, and the corresponding network outputs $y^1, \ldots, y^J \in \mathbb{R}^{l_L}$. We measure the distances between desired and network outputs with the total quadratic error

$$E = \frac{1}{2} \sum_{p=1}^{J} \|d^p - y^p\|^2. \tag{2.5}$$

The norm $\|b\|^2 = \sum b_i^2$ denotes the usual Euclidean norm for a vector $b = (b_1, \ldots, b_n)$. Gradients of the error with respect to weights are then defined as

$$\frac{\partial E}{\partial v_{(i,j)}^{l,m}} = q_{(i,j)}^{l,m}. \tag{2.6}$$

The iteration of weights by backpropagation is done through gradient descent with momentum. Here and in the future, a superscript $k$ is used as an iteration variable, and $\Delta X^k = X^k - X^{k-1}$ for any quantity $X$. Weights are updated as

$$\Delta v_{(i,j)}^{m,l;k+1} = \tau_{(i,j)}^{m,l;k} \Delta v_{(i,j)}^{m,l;k} - \eta q_{(i,j)}^{m,l;k}. \tag{2.7}$$

The second term in (2.7) corresponds to traditional backpropagation through gradient descent, while the first term, for a predetermined $\tau \in (0,1)$, is the contribution from adaptive momentum, with

$$\tau_{(i,j)}^{m,l;k} = \begin{cases} \frac{\tau \|q_{(i,j)}^{m,l;k}\|}{\|\Delta v_{(i,j)}^{m,l;k}\|} & \|\Delta v_{(i,j)}^{m;k}\| \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

When the norm acts on a matrix $A = (a_{i,j})_{n\times m}$, it is treated as the Frobenius norm, with $\|A\|^2 = \sum_{i,j} a_{i,j}^2$. For clarity, we sometimes place a variable denoting iteration after a semicolon to distinguish it from node indices. We note that a similar choice of momentum was also used in [17].

### 2.2. Convergence of backpropagation

Our major theorem is a statement of convergence under backpropagation with momentum. Specifically, for some input $x^j \in l_0$, we will use a generic desired output of $d^j \in \mathbb{R}$. We use a 1-D output for clarity in exposition. The proof of convergence for output in multiple dimensions is essentially the same as the one presented here. The error in this case is then

$$E = \sum_{p=1}^{J} \frac{(d^p - y^p)^2}{2} := \sum_{p=1}^{J} \phi_p(S_L^p). \tag{2.9}$$

We will need some regularity and boundedness assumptions. These assumptions are similar to those used in [16], and may also be found in other nonlinear optimization problems such as [4].

**Assumption 2.1.**

1. *The function g, and its first two derivatives g' and g", are bounded in* $\mathbb{R}$.
2. *The weights* $v_{(i,j)}^k$ *are uniformly bounded over layers* $0 \le i < j \le L$ *and iterations* $k = 1, 2, \ldots$.
3. *The gradient* $\nabla E$ *vanishes only at a finite set of points.*

It readily follows from these assumptions that we may also uniformly bound $q_{(i,j)}^k, H_i^k$, $\phi_p$, $\phi_p'$, and $\phi_p''$.

The purpose of Assumption 3 is to establish convergence with the following Lemma (see [14]):

**Lemma 2.1.** *Let* $f \in C^1(\mathbb{R}^n, \mathbb{R})$*, and suppose that* $\nabla f$ *vanishes at a finite set of points. Then, for a sequence* $\{x_k\}$*, if* $\|\Delta x^k\| \to 0$ *and* $\|\nabla f(x^k)\| \to 0$*, then for some* $x^* \in \mathbb{R}^n$*,* $x^k \to x^*$ *and* $\nabla f(x^*) = 0$.

**Theorem 2.1.** *Under assumptions* 1 *and* 2 *in Assumption* 2.1*, for any* $s \in [0,1)$ *and* $\tau = s\eta$*, there exists* $C > 0$ *such that if*

$$\eta < \frac{1-s}{C(s^2+1)}, \tag{2.10}$$

*then for* $k = 1, 2, \ldots,$

$$E^k = E(v_{(i,j)}^k) \to E^*, \qquad 1 \le i < j \le L, \tag{2.11}$$

$$q_{(i,j)}^k \to 0. \tag{2.12}$$

*If part* (3) *of the Assumptions is satisfied, weights* $v^k_{(i,j)} \to v^*_{(i,j)}$, *and* $E^* = E(v^*_{(i,j)})$ *is a stationary point* ($\nabla E = 0$).

**Remark 2.1.** In the case of $s = 0$, Theorem 2.1 is a statement convergence for backpropagation without momentum. This can be quickly demonstrated through gradient descent on the error function $E$. The proof for 2.1 differs from traditional gradient descent by introducing a recursive formula for $\Delta E^k$ and $\Delta H^k_n$ given in Theorem 4.1 which uses the intrinsic structure of the network.

The constant $C$ used is solely dependent on fixed parameters form the network, and the uniform bounds from the assumptions. A complete proof for Theorem 2.1 is provided in Section 4.

## 3. Experiments

In this section, we give examples of the efficacy for both DAG architectures and the addition of momentum to backpropagation with the example of AutoEncoders, a framework of data compression. The addition of skip connections in AutoEncoders, entitled CrossEncoders, was studied by Agarwal et al. [1]. We will apply CrossEncoders to the MNIST and Olivetti face dataset[1)] .

For the problem of compression, we require a code layer with index $0 < \mathbf{c} < L$ and dimension $l_{\mathbf{c}} < l_0$. Since we are now comparing input and output, layer $L$ also contains $l_0$ nodes. For the error defined in (2.5), we set $d^p = x^p$, and thus

$$E = \frac{1}{2}\sum_{p=1}^{J} \|x^p - y^p\|^2. \tag{3.1}$$

Since decoding should be solely dependent from the code layer, we also require that skip connections cannot occur between encoding layers and decoding layers. Thus

$$v^{l,m}_{(i,j)} = 0 \quad \text{if} \quad i < \mathbf{c} < j. \tag{3.2}$$

See Fig. 3.1 for a visual representation of the CrossEncoder architecture.

### 3.1. Momentum analysis and performance

To study the effect of backpropogation with momentum, we use the standard MNIST [9] dataset of handwritten digits. Each sample is a $28 \times 28$ binary pixel image, transformed to a $1\times784$ vector. The complete dataset consists of $70,000$ images, divided into $60,000$ training and $10,000$ testing images. We train a $784(L_0)-64(L_1)-64(L_2)-code(L_3)-64(L_4)-64(L_5)-784(L_6)$ network using four different cases by considering architectures with and without skip connections, and also backpropagation with zero and positive momentum (set to 0.95). For the CrossEncoder, each node in layer $L_1$ is connected to all nodes in $L_3$ and each node in layer $L_4$ is connected to all nodes in $L_6$. In Fig. 3.2, we show the training loss curve for all the four cases by plotting epochs against mean squared error. We find that for momentum term improves the speed of convergence for architectures with and without skip connections. Furthermore, the addition of skip connections leads to faster convergence for both zero and positive momentum backpropagation.

[1)] Both of these datasets are public, and may may be obtained from https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html (Olivetti) and http://yann.lecun.com/exdb/mnist (MNIST).

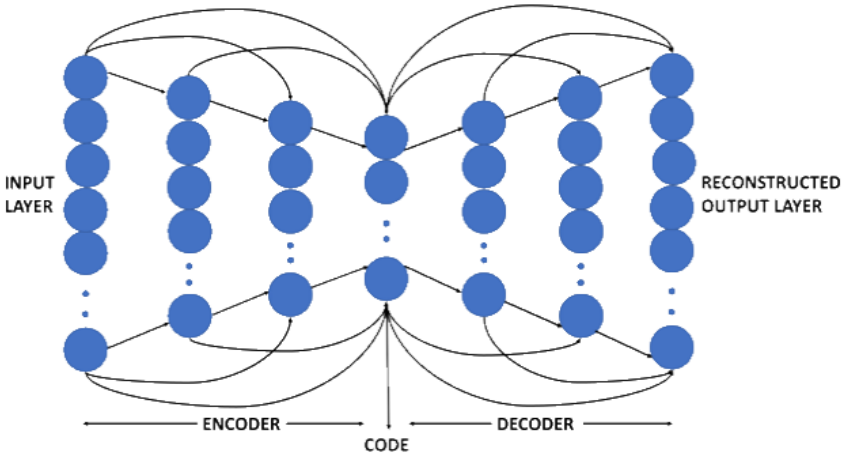


Fig. 3.1. **Architecture for CrossEncoders.** A directed edge from a node (circle) in one layer (column of circles) to another node in a different layer represents a connection. Additional edges between nodes, suppressed for presentation, may also exist. Note, however, that edges may not connect encoding and decoding layers.

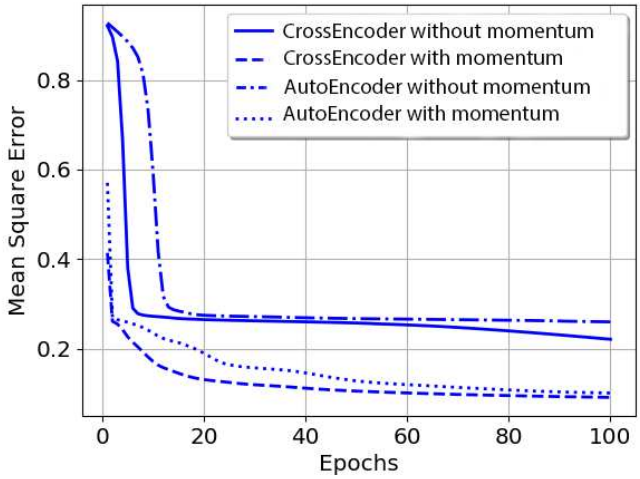


Fig. 3.2. Effect of adding momentum to an optimizer.

### 3.2. DAG architecture and performance

The Olivetti faces dataset [10] is comprised of a set of 400 gray-scale face images consisting of ten different images of 40 distinct subjects. Images for some subjects were taken with varying lighting, facial expressions (e.g. open / closed eyes, smiling / not smiling), and facial details (e.g. glasses / no glasses). The images are $64 \times 64$ in size and are quantized to 8-bit [0-255] scale. A $4096(L_0) - 500(L_1) - 500(L_2) - code(L_3) - 500(L_4) - 500(L_5) - 4096(L_6)$ MLP network was used for training the face dataset. Like the previous example, each node in layer $L_1$ is connected to all nodes in $L_3$ and each node in layer $L_4$ is connected to all nodes in $L_6$.

The original $64 \times 64$ images were transformed to a $1 \times 4096$ vector. For training, 350

images were used, and 50 images were used for the testing dataset. Both AutoEncoders and CrossEncoders were trained for 300 epochs using SGD optimizer with a learning rate set to 0.001 and momentum of 0.95. For the given task, we used several lower dimension representations, such as $1 \times 600$, $1 \times 300$, and $1 \times 30$, respectively. Table 3.1 illustrates the performance of the respective networks for different code size using peak signal to noise ratio (PSNR), structural similarity index (SSIM), and normalized root mean squared error (NRMSE) metrics. In Table 3.1, we observe improved performance for CrossEncoders across all performance metrics.

Table 3.1: PSNR, SSIM, and NRMSE values of CrossEncoder and Autoencoder between reconstructed and the original images for Olivetti face dataset. Higher PSNR and SSIM values, and lower NRSME values, imply more accurate results.

| | **CrossEncoder** | | | **Autoencoder** | | |
|---|---|---|---|---|---|---|
| **Code** | **PSNR** | **SSIM** | **NRMSE** | **PSNR** | **SSIM** | **NRMSE** |
| $1 \times 600$ | **79.4679** | **0.9040** | **0.1464** | 75.8858 | 0.8554 | 0.2217 |
| $1 \times 300$ | **79.4551** | **0.9046** | **0.1467** | 75.8919 | 0.8555 | 0.2215 |
| $1 \times 30$ | **77.8398** | **0.8791** | **0.1764** | 75.9066 | 0.8560 | 0.2211 |

## 4. Proof of Convergence

### 4.1. Notation and conventions

In what follows, we will need some notation for matrix and tensor manipulation. First, we recall the entrywise, or Hadamard, product, which for two matrices $A = (a_{i,j})_{n\times m}, B = (b_{i,j})_{n\times m}$, is defined as $(A \circ B)_{i,j} = a_{i,j}b_{i,j}$. By taking the sum of all entries of a Hadamard product, we obtain the Frobenius inner product $A : B = \sum_{i,j}(A\circ B)_{i,j}$. Also, a matrix gradient of a real (vector) valued function is matrix (tensor) valued, with an element-wise representation as

$$\frac{\partial f}{\partial v^k_{(i,j)}} = \left(\frac{\partial f}{\partial v^{a,b;k}_{(i,j)}}\right)_{l_i\times l_j}, \quad \frac{\partial H^k_m}{\partial v^k_{(i,j)}} = \left(\frac{\partial H^{c;k}_m}{\partial v^{a,b;k}_{(i,j)}}\right)_{l_i\times l_j\times l_m}.$$

In all future estimates, we look at backpropagation over a single input, meaning $J = 1$. This allows us to suppress the variable $p$, which is essentially done for the sake of presentation. The proofs of Theorem 4.1 and Lemma 4.1 generalize immediately to the case of multiple inputs by taking sums over all inputs. Finally, in our estimates, we will use the constant $C > 0$ which depends solely on fixed parameters in the network, such as the input value $x$, uniform bounds of node outputs $H_j$ and inputs $S_j$, and for generalizing to multiple inputs, the size of the dataset $J$. The constant $C$ is used in multiple estimates, and may increase each time it appears.

### 4.2. Estimates on node output increments

Our major technical theorem shows that the increments of outputs $\Delta H^{k+1}_n$ are similar, up to first order, to $Q^k(H^k)$, where $Q^k$ denotes the differential operator

$$Q^k = \sum_{i<j\le L} \Delta v^{k+1}_{(i,j)} : \frac{\partial}{\partial v^k_{(i,j)}}. \tag{4.1}$$

Note that when $Q^k$ acts on a length $l_m$ vector, the matrix inner product in (4.1) is between a $l_i \times l_j$-sized matrix and a $l_i \times l_j \times l_m$-sized tensor, and is a vector of size $l_m$.

The major utility of introducing $Q^k$ is that it provides a simple bound when acting on $E^k$. Specifically, using (2.6)–(2.8), it is straightforward to show that

$$Q^k(E^k) \leq (-\eta + \tau) \sum_{i<j\leq L} \|q^k_{(i,j)}\|^2. \tag{4.2}$$

**Theorem 4.1.** *There exists a universal constant $C > 0$ such that*

$$|Q^k(E^k) - \Delta E^{k+1}| \leq C \left( \sum_{n\leq L} \|\Delta H^{k+1}_n\|^2 + \sum_{m<n\leq L} \|\Delta v^{k+1}_{(m,n)}\|^2 \right). \tag{4.3}$$

*Proof.* We show (4.3) follows through three steps: (1) finding a recurrence relation, with respect to the ordering of hidden layers, for $Q^k(H^k_n)$ and $Q^k(E^k)$; (2) finding a similar relation for $\Delta H^{k+1}_n$ and $\Delta E^{k+1}$; and (3) comparing the two relations.

(1) (A recurrence for $Q^k(H^k_n)$ and $Q^k(E^k)$). Applying the chain rule to the total error (2.9), using (2.3), and rearranging sums,

$$\begin{aligned} Q^k(E^k) &= \phi'(S^k_L) \sum_{i<j\leq L} \Delta v^{k+1}_{(i,j)} : \frac{\partial}{\partial v^k_{(i,j)}} \left( \sum_{m<L} H^k_m v^k_{(m,L)} \right) \\ &= \phi'(S^k_L) \sum_{m<L} \sum_{i<j\leq L} \Delta v^{k+1}_{(i,j)} : \frac{\partial}{\partial v^k_{(i,j)}} \left( H^k_m v^k_{(m,L)} \right). \end{aligned} \tag{4.4}$$

We now focus on expressing (4.4) in a recursive form. We begin with considering the terms in (4.4) with $j = L$. We first work elementwise by differentiating with respect to the $(a, b)$ entry of the matrix derivative for $\frac{\partial}{\partial v^k_{(i,j)}} \left( H^k_m v^k_{(m,L)} \right)$. From the product rule, this can be written as a sum of vectors, with

$$\begin{aligned} &\frac{\partial}{\partial v^{a,b;k}_{(i,L)}} \left( H^k_m v^k_{(m,L)} \right) = \frac{\partial H^k_m}{\partial v^{a,b;k}_{(i,L)}} v^k_{(m,L)} + H^k_m \frac{\partial v^k_{(m,L)}}{\partial v^{a,b;k}_{(i,L)}} \\ =&\frac{\partial H^k_m}{\partial v^{a,b;k}_{(i,L)}} v^k_{(m,L)} + (0, \ldots, \underbrace{\delta_{i,m} H^{a;k}_m}_{b^{th} \text{ entry}}, \ldots, 0) \\ =:& A^{a,b;k}_{i,m} + B^{a,b;k}_{i,m}. \end{aligned} \tag{4.5}$$

Each of these terms is handled in turn. First, summing the Frobenius inner product of the matrix $\Delta v^{k+1}_{(i,L)}$ and the tensor $A^k_{i,m}$, we may write

$$\begin{aligned} \sum_{m<L} \sum_{i<L} \Delta v^{k+1}_{(i,L)} : A^k_{i,m} &= \sum_{m<L} \sum_{i<L} \sum_{\substack{a<l_i \\ b<l_L}} \Delta v^{a,b;k+1}_{(i,L)} \frac{\partial H^k_m}{\partial v^{a,b;k}_{(i,L)}} v^k_{(m,L)} \\ &= \sum_{m<L} \sum_{i<L} \left( \Delta v^{k+1}_{(i,L)} : \frac{\partial H^k_m}{\partial v^k_{(i,L)}} \right) v^k_{(m,L)}. \end{aligned} \tag{4.6}$$

For $B^k_{i,m}$, we also work elementwise, and write the Frobenius inner product as

$$\sum_{m<L}\sum_{i<L}\Delta v^{k+1}_{(i,L)} : B^k_{i,m} = \sum_{m<L}\sum_{\substack{a\le l_m\\ b\le l_L}} \Delta v^{a,b;k+1}_{(m,L)} B^{a,b;k}_{m,m}$$

$$= \sum_{m<L}\left(\sum_{a\le l_m} H^{a;k}_m \Delta v^{a,1;k+1}_{(m,L)}, \ldots, \sum_{a\le l_m} H^{a;k}_m \Delta v^{a,l_L;k+1}_{(m,L)}\right)$$

$$= \sum_{m<L} H^k_m \Delta v^{k+1}_{(m,L)}. \tag{4.7}$$

Calculations for double sum in (4.4) for the remaining terms with $j < L$ are similar to the case $j = L$, except that there is no corresponding $B^k_{i,m}$ term. Indeed, we can show

$$\sum_{m<L}\sum_{i<j<L}\Delta v^{k+1}_{(i,j)} : \frac{\partial}{\partial v^k_{(i,j)}}\left(H^k_m v^k_{(m,L)}\right) = \sum_{m<L}\sum_{i<j<L}\left(\Delta v^{k+1}_{(i,j)} : \frac{\partial H^k_m}{\partial v^k_{(i,j)}}\right) v^k_{(m,L)}. \tag{4.8}$$

Putting together (4.4)-(4.8), we arrive at

$$\sum_{m<L}\sum_{i<j\le L}\Delta v^{k+1}_{(i,j)} : \frac{\partial}{\partial v^k_{(i,j)}}\left(H^k_m v^k_{(m,L)}\right) \tag{4.9}$$

$$= \sum_{m<L}\left(H^k_m \Delta v^{k+1}_{(m,L)} + \sum_{i<j\le m}\left(\Delta v^{k+1}_{(i,j)} : \frac{\partial H^k_m}{\partial v^k_{(i,j)}}\right) v^k_{(m,L)}\right) \tag{4.10}$$

$$= \sum_{m<L}\left(H^k_m \Delta v^{k+1}_{(m,L)} + Q^k(H^k_m) v^k_{(m,L)}\right). \tag{4.11}$$

Note that (4.10) uses the fact that since $H^k_m$ only depends on layers 1 through $m-1$, we may truncate the sum of $Q^k$ and write

$$Q^k(H^k_m) = \sum_{i<j\le m}\Delta v^{k+1}_{(i,j)} : \frac{\partial H^k_m}{\partial v^k_{(i,j)}}. \tag{4.12}$$

We may now substitute (4.9) into (4.4) to yield the recursive formula

$$Q^k(E^k) = \phi'\left(S^k_L\right)\sum_{m<L}\left(H^k_m \Delta v^{k+1}_{(m,L)} + Q^k(H^k_m) v^k_{(m,L)}\right). \tag{4.13}$$

From similar calculations, the formula over a node $H^k_n$, with $n < L$, is

$$Q^k(H^k_n) = g'\left(S^k_n\right) \circ \sum_{m<n}\left(H^k_m \Delta v^{k+1}_{(m,n)} + Q^k(H^k_m) v^k_{(m,n)}\right). \tag{4.14}$$

(2) (A recurrence for $\Delta H^{k+1}_n$ and $\Delta E^{k+1}$). A recursive formula for $\Delta H^{k+1}_n$ is found through a Taylor expansion of $E(S^{k+1}_L)$ centered at $S^k_L$. Specifically, there exists $t_k$ between $S^k_L$ and

$S_L^{k+1}$ with

$$\begin{aligned}\Delta E^{k+1} &= \phi'\left(S_L^k\right)\Big(\sum_{m<L}\Delta(H_m^{k+1}v_{(m,L)}^{k+1})\Big) + \frac{1}{2}\phi''(S_L^k)\left(\sum_{m<L}\Delta(H_m^{k+1}v_{(m,L)}^{k+1})\right)^2\\ &= \phi'\left(S_L^k\right)\sum_{m<L}\Big(\Delta H_m^{k+1}v_{(m,L)}^k + H_m^k\Delta v_{(m,L)}^{k+1} + \Delta H_m^{k+1}\Delta v_{(m,L)}^{k+1}\Big)\\ &\quad + \frac{1}{2}\phi''(t_k)\left(\sum_{m<L}\Delta(H_m^{k+1}v_{(m,L)}^{k+1})\right)^2. \end{aligned} \tag{4.15}$$

Similarly, there exist $t_{n,k} = (t_{n,k}^1, \ldots, t_{n,k}^{l_l})$ where each $t_{n,k}^r$ lies between $S_n^{r;k}$ and $S_n^{r;k+1}$ for $r = 1, \ldots l_n$ and

$$\begin{aligned}\Delta H_n^{k+1} =& g'\left(S_n^k\right)\circ\sum_{m<n}\Big(\Delta H_m^{k+1}v_{(m,n)}^k + H_m^k\Delta v_{(m,n)}^{k+1} + \Delta H_m^{k+1}\Delta v_{(m,n)}^{k+1}\Big)\\ &+ \frac{1}{2}g''(t_{n,k})\circ\left(\sum_{m<n}\Delta(H_m^{k+1}v_{(m,n)}^{k+1})\right)^2. \end{aligned} \tag{4.16}$$

(3) (Comparing recurrences). From (1) and (2) of Assumption 2.1, we may derive the simple bound

$$\|\Delta H_m^{k+1}\Delta v_{(m,n)}^{k+1}\| \le C\Big(\|\Delta v_{(m,n)}^{k+1}\|^2 + \|\Delta H_m^{k+1}\|^2\Big) \tag{4.17}$$

for some constant $C > 0$. Taking differences of (4.16) and (4.14), for any $n < L$, we then obtain the recurrence inequality

$$\begin{aligned}&\|Q^k(H_n^k) - \Delta H_n^{k+1}\|\\ \le& C\left(\sum_{m<n}\|Q^k(H_m^k) - \Delta H_m^{k+1}\|\right) + C\sum_{m<n}(\|\Delta v_{(m,n)}^{k+1}\|^2 + \|\Delta H_m^{k+1}\|^2). \end{aligned} \tag{4.18}$$

Replacing $H_n^k$ with $E^k$ in (4.18) produces the same type of inequality, with the sum in (4.18) now ranging from $m = 1, \ldots, L-1$. Repeated applications of (4.18) to $E^k$ and subsequently to $H_n^k$, for $n = 1, \ldots, L-1$, result in

$$\begin{aligned}&|Q^k(E^k) - \Delta E^{k+1}|\\ \le& C\left(\|Q^k(H_0^k) - \Delta H_0^{k+1}\|\right) + C\left(\sum_{n<L}\|\Delta H_n^{k+1}\|^2 + \sum_{m<n<L}\|\Delta v_{(m,n)}^{k+1}\|^2\right). \end{aligned} \tag{4.19}$$

To complete the proof, we note that the input data $x$ does not change under iterations, so

$$Q^k(H_0^k) - \Delta H_0^{k+1} \equiv 0. \tag{4.20}$$

This completes the proof of the theorem. □

We now bound the quadratic terms in (4.3).

**Lemma 4.1.** *For some constant $C > 0$, we have*

$$\|\Delta v_{(m,n)}^{k+1}\| \le (\eta + \tau)\|q_{(m,n)}^k\|, \tag{4.21}$$

$$\|\Delta H_n^{k+1}\| \le C(\eta + \tau)\sum_{i<j<n}\|q_{(i,j)}^k\|. \tag{4.22}$$

*Proof.* We may show (4.21) immediately from (2.7) and (2.8). For (4.22), we use strong induction, assuming the inequality holds for layers $m < n$ (note that the base case holds trivially for $n = 0$). From the Taylor expansion used in (4.16), and the boundedness of $g'$ and $g''$:

$$\begin{aligned}\|\Delta H_n^{k+1}\| \leq & C \sum_{m<n} \left\| \Delta H_m^{k+1} v_{(m,n)}^k + H_m^k \Delta v_{(m,n)}^{k+1} + \Delta H_m^{k+1} \Delta v_{(m,n)}^{k+1} \right\| \\ & + C \left\| \left( \sum_{m<n} \Delta (H_m^{k+1} v_{(m,n)}^{k+1}) \right)^2 \right\|. \end{aligned} \tag{4.23}$$

From the induction hypothesis, the boundedness of weights and node outputs, and (4.21), the right hand side of (4.23) is bounded by the right hand side of (4.22) for some $C > 0$. From the boundedness assumptions,

$$\left\| \left( \sum_{m<n} \Delta (H_m^{k+1} v_{(m,n)}^{k+1}) \right)^2 \right\| \leq C \left\| \sum_{m<n} \Delta (H_m^{k+1} v_{(m,n)}^{k+1}) \right\|, \tag{4.24}$$

which implies that we may use similar estimates for (4.24) which we used in (4.23) to arrive at (4.22). □

From Theorem 4.1, (4.2), and Lemma 4.1, the iteration of error may now be estimated as

$$\begin{aligned}\Delta E^{k+1} &\leq C \left( \sum_{n \leq L} \|\Delta H_n^{k+1}\|^2 + \sum_{m<n \leq L} \|\Delta v_{(m,n)}^{k+1}\|^2 \right) + Q^k(E^k) \\ &\leq \left( -\eta + \tau + C(\tau^2 + \eta^2) \right) \sum_{m<n \leq L} \|q_{(m,n)}^k\|^2. \end{aligned} \tag{4.25}$$

### 4.3. Proof of convergence

For some $s \in [0, 1)$, assume $\tau = s\eta$. It is straightforward to show that the term in front of the norms in (4.25) is negative when

$$\eta < \frac{1 - s}{C(s^2 + 1)}. \tag{4.26}$$

Under this constraint, $E^k$ is decreasing under each iteration. The summability for $\|q_{(i,j)}^k\|^2$ also follows, since

$$\sum_{k=1}^{\infty} \|q_{(i,j)}^k\|^2 \leq \frac{1}{(\eta - \tau - C(\tau^2 + \eta^2))} \sum_{k=1}^{\infty} \Delta E^k < \infty. \tag{4.27}$$

Thus $\|q_{(i,j)}^k\| \to 0$ and, from (4.21), $\|\Delta v_{(i,j)}^k\| \to 0$. Lemma 2.1 and part (3) of Assumption 2 imply a set of minimum weights $v_1^*, v_2^*, z^*, w^*$, which determine a stationary point of $E$. This shows Theorem 2.1. □

## 5. Conclusion

We have studied a feed-forward network with skip-layer connections. The possible directed graph architectures are the class of directed acyclic graphs. As shown in [6], introducing skip

connections often increases the performance of a deep neural network. In [1] and in Section 3, we have demonstrated increased performance in the setting of AutoEncoders. For our main result, we have established the convergence of backpropagation with adaptive momentum of networks with skip connections. This generalizes the result of Wu et al. [16] who established convergence for a feed forward network with one hidden layer. While we have considered general DAG architectures, it remains to investigate, both through theory and experiment, the optimality properties with regards to the number of layers and skip-connections. We hope to address these properties in future works.

## References

[1] C. Agarwal, M. Sharifzadeh and D. Schonfeld, Crossencoders: An image compression framework. *Accepted in Electronic Imaging 2018* (2018).

[2] A. Bhaya and E. Kaszkurewicz, Steepest descent with momentum for quadratic functions is a version of the conjugate gradient method. *Neural Networks 17*, **1** (2004), 65-71.

[3] C.M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995.

[4] M. Gori and M. Maggini, Optimal convergence of on-line backpropagation. *IEEE transactions on neural networks 7*, **1** (1996), 251-254.

[5] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), 770-778.

[6] G. Huang, Z. Liu, L. Van Der Maaten and K.Q. Weinberger, Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 4700-4708.

[7] R. Kothari and K. Agyepong, On lateral connections in feed-forward neural networks. In *Neural Networks, 1996., IEEE International Conference on* **1** (1996), IEEE, 13-18.

[8] G. Larsson, M. Maire and G. Shakhnarovich, Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648* (2016).

[9] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, **11** (1998), 2278-2324.

[10] M. Minear and D.C. Park, A lifespan database of adult facial stimuli. *Behavior Research Methods, Instruments, & Computers 36*, **4** (2004), 630-633.

[11] F. Rosenblatt, Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Tech. rep., DTIC Document, 1961.

[12] D.F. Rumelhart, J.L. McClelland, P.R. Group, et al. *Parallel Distributed Processing*, vol. 1. MIT Press Cambridge, MA, 1987.

[13] R.K. Srivastava, K. Greff and J. Schmidhuber Training very deep networks. In *Advances in neural information processing systems* (2015), 2377-2385.

[14] W. Sun and Y.X. Yuan, *Optimization Theory and Methods: Nonlinear Programming*, vol. 1. Springer Science & Business Media, 2006.

[15] M. Torii and M.T. Hagan, Stability of steepest descent with momentum for quadratic functions. *IEEE Transactions on Neural Networks 13*, **3** (2002), 752-756.

[16] W. Wu, N. Zhang, Z. Li, L. Li and Y. Liu, Convergence of gradient method with momentum for back-propagation neural networks. *Journal of Computational Mathematics* (2008), 613-623.

[17] N. Zhang, W. Wu and G. Zheng, Convergence of gradient method with momentum for two-layer feedforward neural networks. *IEEE Transactions on Neural Networks 17*, **2** (2006), 522-525.