A Geometric Framework for Convolutional Neural Networks

Anthony L. Caterini and Dong Eui Chang
Department of Applied Mathematics
University of Waterloo
Waterloo, ON, Canada, N2L 3G1
{alcateri, dechang}@uwaterloo.ca

Abstract

In this paper, a geometric framework for neural networks is proposed. This framework uses the inner product space structure underlying the parameter set to perform gradient descent not in a component-based form, but in a coordinate-free manner. Convolutional neural networks are described in this framework in a compact form, with the gradients of standard — and higher-order — loss functions calculated for each layer of the network. This approach can be applied to other network structures and provides a basis on which to create new networks.

1 Introduction

Machine Learning algorithms have long worked with multi-dimensional vector data and parameters, but have not exploited the underlying inner product space structure. A recent paper on deep learning in Nature called for “new paradigms” involving “operations on large vectors” [5] to propel the field forward. This approach is taken to describe the convolutional neural network (CNN) in this paper. In particular, the layers are described as vector-valued maps, and gradients of these maps with respect to the parameters at each layer are taken in a coordinate-free manner. This approach promotes a greater understanding of the network than a coordinate-based approach, and allows for loss function gradients to be calculated compactly using coordinate-free backpropagation of error. This paper also considers a higher-order loss function, as in [7] and [8]. Algorithms to compute one iteration of gradient descent are provided for both types of loss functions to clarify the application of the developed theory. The precise notation developed throughout this paper provides a mathematical standard upon which deep learning can be researched, overcoming the inconsistent notation currently employed across the field. The framework developed in this paper is flexible, and can be extended to cover other types of network structures, and even inspire further developments in deep learning.

2 Mathematical Preliminaries

Some prerequisite notation and concepts are introduced here before CNNs can be fully described.

2.1 Multilinear Algebra and Derivatives

Every individual vector space is assumed to be an inner product space, with the inner product represented by \( \langle , \rangle \). The inner product naturally extends to the direct product \( E_1 \times \cdots \times E_r \) of inner product spaces \( E_1, \ldots, E_r \) and their tensor product \( E_1 \otimes \cdots \otimes E_r \) as follows [2]:

\[
\langle (e_1, \ldots, e_r), (\bar{e}_1, \ldots, \bar{e}_r) \rangle = \sum_{i=1}^{r} \langle e_i, \bar{e}_i \rangle,
\]

\[
\langle e_1 \otimes \cdots \otimes e_r, \bar{e}_1 \otimes \cdots \otimes \bar{e}_r \rangle = \prod_{i=1}^{r} \langle e_i, \bar{e}_i \rangle,
\]
where $e_i, e_j \in E_i, i = 1, \ldots, r$. The symbol $\otimes$ is exclusively used to denote the tensor product operator in this paper. An inner product space $E$ is canonically identified here with its dual space $E^*$ using the inner product on $E$, so dual spaces will rarely be used in this paper. The set of $r$-linear maps from $E_1 \times \cdots \times E_r$ to a vector space $F$ is denoted by $L(E_1, \ldots, E_r; F)$. For a linear map $L \in L(E; F)$, its adjoint map, denoted by $L^*$, is a linear map in $L(F; E)$ defined by the relationship $(L^* f, e) = \langle f, L e \rangle$ for all $e \in E$ and $f \in F$. For each vector $e_i \in E_i$ and any bilinear map $B \in L(E_1, E_2; F)$, define a linear map $(e_1 \cdot B) : E_2 \rightarrow F$ by

\[(e_1 \cdot B)(e) = B(e_1, e)\]

for all $e \in E_2$. Likewise, for each vector $e_2 \in E_2$ and any bilinear map $B \in L(E_1, E_2; F)$, define a linear map $(B \cdot e_2) : E_1 \rightarrow F$ by

\[(B \cdot e_2)(e) = B(e, e_2)\]

for all $e \in E_1$.

Now, notation for derivatives in accordance with [1] is presented. Consider a map $f : E_1 \rightarrow E_2$. The (first) derivative $Df(x)$ of $f$ at a point $x \in E_1$ is a linear map from $E_1$ to $E_2$, i.e. $Df(x) \in L(E_1; E_2)$, and it can be defined as

\[Df(x) \cdot v = \frac{d}{dt} f(x + tv) \bigg|_{t=0}\]

for any $v \in E_1$. The derivative $Df$ can be viewed as a map from $E_1$ to $L(E_1; E_2)$, defined by $x \mapsto Df(x)$. Let $D^*f(x)$ denote the adjoint of $Df(x)$ so that $(w, Df(x) \cdot v) = (D^*f(x) \cdot w, v)$ for all $v \in E_1$ and $w \in E_2$.

Now consider a map $f : E_1 \times F_1 \rightarrow E_2$ written as $f(x; \theta)$ for $x \in E_1$ and $\theta \in F_1$, where the semi-colon is inserted between $x$ and $\theta$ to distinguish the state variable $x$ from the parameters $\theta$. Let $Df(x; \theta)$ denote the derivative of $f$ with respect to $x$ evaluated at $(x; \theta)$, and let $\nabla f(x; \theta)$ denote the derivative of $f$ with respect to $\theta$ evaluated at $(x; \theta)$. It is easy to verify that $Df(x; \theta) \in L(E_1; E_2)$ and $\nabla f(x; \theta) \in L(F_1; E_2)$ and that

\[Df(x; \theta) \cdot e = \frac{d}{dt} f(x + te; \theta) \bigg|_{t=0}, \quad \nabla f(x; \theta) \cdot u = \frac{d}{dt} f(x; \theta + tu) \bigg|_{t=0}\]

for all $e \in E_1$ and $u \in F_1$. The adjoints of $Df(x; \theta)$ and $\nabla f(x; \theta)$ are denoted by $D^*f(x; \theta)$ and $\nabla^* f(x; \theta)$, respectively. Sometimes, $\nabla_\theta f$ is written instead of $\nabla f$, to emphasize differentiation of $f$ with respect to the parameter variable $\theta$.

The second derivative $D^2f(x; \theta)$ of $f$ with respect to $x$ evaluated at $(x; \theta)$ is a bilinear map in $L(E_1, E_1; E_2)$ defined as follows: for any $e, \bar{e} \in E_1$,

\[D^2f(x; \theta) \cdot (e, \bar{e}) = \frac{d}{dt} Df(x + te; \theta) \cdot \bar{e} \bigg|_{t=0} = \frac{\partial^2}{\partial t \partial s} f(x + te + s\bar{e}; \theta) \bigg|_{t=s=0}.\]

It is assumed that every function that appears in this paper is (piecewise) twice continuously differentiable. The second derivative $D^2f(x; \theta)$ is symmetric, i.e. $D^2f(x; \theta) \cdot (e, \bar{e}) = D^2f(x; \theta) \cdot (\bar{e}, e)$ for all $e, \bar{e} \in E_1$. The second derivative $\nabla^2 f(x; \theta)$ of $f$ with respect to $x$ and $\theta$ at the point $(x; \theta)$ is a bilinear map in $L(E_1, F_1; E_2)$ defined as follows: for any $e \in E_1$ and $u \in F_1$,

\[\nabla^2 f(x; \theta) \cdot (u, e) = \frac{d}{dt} Df(x; \theta + tu) \cdot e \bigg|_{t=0}.\]

On the other hand, the second derivative $\nabla Df(x; \theta)$ of $f$ with respect to $\theta$ and $x$ at the point $(x; \theta)$ denotes the bilinear map in $L(F_1, E_1; E_2)$ defined as follows: for any $u \in F_1$ and $e \in E_1$,

\[\nabla Df(x; \theta) \cdot (u, e) = \frac{d}{dt} Df(x; \theta + tu) \cdot e \bigg|_{t=0}.\]

Note that for all $e \in E_1$ and $u \in F_1$, it is easy to verify that

\[\nabla Df(x; \theta) \cdot (e, u) = \nabla Df(x; \theta) \cdot (u, e).\]
2.2 Backpropagation in a Nutshell

Now, backpropagation will be presented in a coordinate-free form. Given two maps \( f_1(x; \theta_1) \in E_2 \) for \( x \in E_1, \theta_1 \in F_1 \) and \( f_2(z; \theta_2) \in E_3 \) for \( z \in E_2, \theta_2 \in F_2 \), the composition \( f := f_2 \circ f_1 \) is the map defined as follows:

\[
f(x; \theta_1, \theta_2) = f_2(f_1(x; \theta_1); \theta_2),
\]
(2)

for \( x \in E_1, \theta_1 \in F_1, \theta_2 \in F_2 \). In this framework, functions are composed with respect to the state variables. By the chain rule,

\[
Df = (Df_2 \circ f_1) \cdot Df_1,
\]

where the dependency on the parameters \( \theta_1 \) and \( \theta_2 \) is suppressed for brevity, which shall be understood throughout the paper. In particular, taking the adjoint of \( \nabla_{\theta_1} f = (Df_2 \circ f_1) \cdot \nabla_{\theta_1} f_1 \) produces

\[
\nabla_{\theta_1}^* f = ((Df_2 \circ f_1) \cdot \nabla_{\theta_1} f_1)^* = \nabla_{\theta_1}^* f_1 \cdot (D^* f_2 \circ f_1)
\]
(4)

which is backpropagation in a nutshell. This can be seen by the following: consider a loss function \( J \) defined by

\[
J(x; \theta_1, \theta_2) = \frac{1}{2} \| f(x; \theta_1, \theta_2) - y \|^2 = \frac{1}{2} \| f(x; \theta_1, \theta_2) - y, f(x; \theta_1, \theta_2) - y \|
\]

for some vector \( y \in E_3 \) that may depend on \( x \), along with \( f \) as in (2). Then, for any \( u \in F_1 \), with \( \theta = \{\theta_1, \theta_2\} \) representing the parameters,

\[
\nabla_{\theta_1} J(x; \theta) \cdot u = \langle f(x; \theta) - y, \nabla_{\theta_1} f(x; \theta) \cdot u \rangle = \langle \nabla_{\theta_1}^* f(x; \theta) \cdot (f(x; \theta) - y), u \rangle.
\]
(5)

Since this holds for any \( u \in F_1 \), the canonical identification of an inner product space with its dual is used to obtain

\[
\nabla_{\theta_1} J(x; \theta) = \nabla_{\theta_1}^* f(x; \theta) \cdot (f(x; \theta) - y)
\]

(6)

\[
= \nabla_{\theta_1}^* f_1(x; \theta_1) \cdot D^* f_2(f_1(x; \theta_1); \theta_2) \cdot (f(x; \theta) - y),
\]

where (4) is used for the second equality. This shows that the error \( (f(x; \theta) - y) \) propagates backward from layer 2 to layer 1 through multiplication by \( D^* f_2 \). The adjoint operator reverses the direction of composition, i.e. \( (L_1 L_2)^* = L_2^* L_1^* \), which is the key to backpropagating the error.

The second derivative \( D^2 f \) of \( f = f_2 \circ f_1 \) is given by

\[
D^2 f(x) \cdot (e, \tilde{e}) = D^2 f_2(f_1(x)) \cdot (Df_1(x), e, Df_1(x), \tilde{e}) + Df_2(f_1(x)) \cdot D^2 f_1(x) \cdot (e, \tilde{e})
\]

for all \( e, \tilde{e} \in E_1 \). The second derivative \( \nabla_{\theta_1} Df \) is given by

\[
\nabla_{\theta_1} Df(x) \cdot (u, e) = \nabla_{\theta_1} Df_2(f_1(x)) \cdot (\nabla_{\theta_1} f_1(x), u, Df_1(x) \cdot e) + Df_2(f_1(x)) \cdot \nabla_{\theta_1} Df_1(x) \cdot (u, e)
\]

for all \( e \in E_1 \) and \( u \in F_1 \), which is equivalent to the following: for any fixed \( e \in E_1 \)

\[
\nabla_{\theta_1} Df(x) \cdot e = (D^2 f_2(f_1(x)) \cdot \nabla_{\theta_1} f_1(x)) \cdot \nabla_{\theta_1} f_1(x) + Df_2(f_1(x)) \cdot \nabla_{\theta_1} Df_1(x) \cdot e.
\]
(7)

which is a linear map from \( F_1 \) to \( E_3 \), or by (1)

\[
e \mapsto D \nabla_{\theta_1} f(x) = ((Df_1(x), e) \mapsto D^2 f_2(f_1(x)) \cdot \nabla_{\theta_1} f_1(x) + Df_2(f_1(x)) \cdot (e \mapsto D \nabla_{\theta_1} f_1(x)).
\]
(8)

The adjoint of (7) or (8) yields higher-order backpropagation of error, say for a loss function \( R = \frac{1}{2} \| Df(x) \cdot e - y \|^2 \) with some \( e \in E_1 \) and \( y \in E_3 \) that may depend on \( x \), but not on the parameters. Higher-order backpropagation will be studied in more detail in the next section.

Backpropagation can be expressed recursively for the composition of more than two functions. Consider \( L \) functions \( f_t(x_t; \theta_t) \in E_t \) for \( x_t \in E_{t-1}, \theta_t \in F_t, t = 1, \ldots, L \). Define the composition \( F = f_L \circ \cdots \circ f_1 \). Let \( \omega_t = f_{L \cdot t} \circ \cdots \circ f_t \) and \( \alpha_t = f_t \circ \cdots \circ f_1 \) for \( t = 1, \ldots, L \) so that

\[
F = \omega_{t+1} \circ \alpha_t, \quad \omega_t = \omega_{t+1} \circ f_1, \quad \alpha_{t+1} = f_{t+1} \circ \alpha_t
\]
(9)

for all \( t = 1, \ldots, L - 1 \). The first and second derivatives of (9) and their adjoints can be easily obtained.
3 Convolutional Neural Networks

This section will describe how the above framework can be applied to convolutional neural networks; refer to [3] or [4], for example, for more on the theory of CNNs. First, the actions of one layer of a generic CNN will be described, and then this will be extended to multiple layers. A coordinate-free gradient descent algorithm will also be described. Note that in this section, all bases of inner product space will be assumed to be orthonormal.

3.1 Single Layer Formulation

The actions of one layer of the network will be denoted \( f(X; W, B) \), where \( X \in \mathbb{R}^{n_1 \times f_1} \otimes \mathbb{R}^{m_1} \) is the state variable, and \( W \in \mathbb{R}^{p \times q} \otimes \mathbb{R}^{m_2} \) and \( B \in \mathbb{R}^{n_1 \times f_1} \otimes \mathbb{R}^{m_2} \) are the parameters. Throughout this section, let \( \{ e_i \} \) (resp. \( \{ e_a \} \) be a basis for \( \mathbb{R}^{m_1} \) (resp. \( \mathbb{R}^{m_2} \)), and let \( \{ E_{jk} \}, \{ E_{jk} \}, \{ E_{jk} \} \) be a basis for \( \mathbb{R}^{1 \times f_1} \) (resp. \( \mathbb{R}^{n_1 \times f_1} \), \( \mathbb{R}^{p \times q} \), \( \mathbb{R}^{n_2 \times f_2} \)). Then, \( X, W \) and \( B \) can be written as follows:

\[
X = \sum_{i=1}^{m_1} X_i \otimes e_i, \quad W = \sum_{a=1}^{m_2} W_a \otimes e_a, \quad B = \sum_{a=1}^{m_2} B_a \otimes e_a.
\]

Each \( X_i \in \mathbb{R}^{n_1 \times f_1} \) is called a feature map, which corresponds to an abstract representation of the input for a generic layer. Each \( W_a \in \mathbb{R}^{p \times q} \) is a filter used in convolution, and each \( B_a \in \mathbb{R}^{1 \times f_1} \) is a bias term. The actions of the layer are then a new set of feature maps, \( f(X; W, B) \in \mathbb{R}^{n_1 \times f_2} \otimes \mathbb{R}^{m_2} \), with explicit form given by:

\[
f(X; W, B) = \Psi (S(C(W, X) + B)),
\]

where \( \Psi \) is a pooling operator, \( S \) is an elementwise nonlinear function, and \( C \) is the convolution operator, all of which will be defined in this section.

3.1.1 Cropping, Embedding and Mixing Operators

The cropping and mixing operators will be used to define the convolution operator \( C \) that appears in (10). The cropping operator, \( \kappa_{jk} \in \mathcal{L}(\mathbb{R}^{n_1 \times f_1} \otimes \mathbb{R}^{m_1}; \mathbb{R}^{p \times q} \otimes \mathbb{R}^{m_1}) \), is defined as:

\[
\kappa_{jk} \left( \sum_{i=1}^{m_1} X_i \otimes e_i \right) = \sum_{i=1}^{m_1} \kappa_{jk}(X_i) \otimes e_i,
\]

where \( \kappa_{jk} \in \mathcal{L}(\mathbb{R}^{n_1 \times f_1}; \mathbb{R}^{p \times q}) \) is defined as:

\[
\kappa_{jk}(X_i) = \sum_{r=1}^{p} \sum_{s=1}^{q} \langle X_i, E_{j+r-1, k+s-1} \rangle \tilde{E}_{rs}.
\]

Define the embedding operator \( \text{Em}_{e,d} \in \mathcal{L}(\mathbb{R}^{p \times q}; \mathbb{R}^{n_1 \times f_1}) \) by

\[
\text{Em}_{e,d}(Y) = \sum_{r=1}^{p} \sum_{s=1}^{q} Y_{rs} E_{c+r-1, d+s-1}
\]

for \( Y = \sum_{r=1}^{p} \sum_{s=1}^{q} Y_{rs} \tilde{E}_{rs} \in \mathbb{R}^{p \times q} \), which corresponds to embedding \( Y \) into the zero matrix when \( \{ E_{jk} \} \) is the standard basis. The adjoints of \( \kappa_{jk} \) and \( \kappa_{jk} \) are calculated as follows:

**Theorem 1.** For any \( Z = \sum_{i=1}^{m_1} Z_i \otimes e_i \in \mathbb{R}^{p \times q} \otimes \mathbb{R}^{m_1} \),

\[
\kappa_{jk}^*(Z_i) = \text{Em}_{j,k}(Z_i).
\]

**Proof.** Let \( Q \in \mathbb{R}^{n_1 \times f_1} \). Then, for any \( i \in \{ 1, \ldots, m_1 \} \),

\[
\langle Z_i, \kappa_{jk}(Q) \rangle = \langle Z_i, \sum_{r=1}^{p} \sum_{s=1}^{q} \langle Q, E_{j+r-1, k+s-1} \rangle \tilde{E}_{rs} \rangle = \sum_{r=1}^{p} \sum_{s=1}^{q} \langle Z_i, \tilde{E}_{rs} \rangle E_{j+r-1, k+s-1}, Q \rangle = \langle \text{Em}_{j,k}(Z_i), Q \rangle,
\]

where, for any \( i \in \{ 1, \ldots, m_1 \} \),

\[
\kappa_{jk}^*(Z_i) = \text{Em}_{j,k}(Z_i).
\]
which proves (14). Furthermore, let \( X = \sum_{i=1}^{m_1} X_i \otimes e_i \in \mathbb{R}^{m_2 \times \ell_1} \otimes \mathbb{R}^{m_1} \). Then,

\[
\langle Z, \kappa_{jk}(X) \rangle = \left\langle \sum_{i=1}^{m_1} Z_i \otimes e_i, \sum_{i=1}^{m_1} \kappa_{jk}(X_i) \otimes e_i \right\rangle
\]

\[
= \sum_{i=1}^{m_1} \langle Z_i, \kappa_{jk}(X_i) \rangle
\]

\[
= \sum_{i=1}^{m_1} (\kappa_{jk}^*(Z_i), X_i)
\]

\[
= \left( \sum_{i=1}^{m_1} \kappa_{jk}^*(Z_i) \otimes e_i, X \right),
\]

which completes the proof.

For \( v \in \mathbb{R}^{m_1} \), the *mixing* operator \( \Phi_v \in \mathcal{L}(\mathbb{R}^{p \times q} \otimes \mathbb{R}^{m_1}; \mathbb{R}^{p \times q}) \) defines how the cropped feature maps are combined into the next layer of feature maps, which is useful in a framework such as [6]. It can be explicitly represented as:

\[
\Phi_v \left( \sum_{i=1}^{m_1} U_i \otimes e_i \right) = \sum_{i=1}^{m_1} v_i U_i,
\]

(15)

where \( v = \sum_{i=1}^{m_1} v_i e_i \). The adjoint operator \( \Phi_v^* \) has a compact form, as the following lemma describes.

**Lemma 2.** For any \( Y \in \mathbb{R}^{p \times q} \) and \( v \in \mathbb{R}^{m_1} \),

\[
\Phi_v^* \cdot Y = Y \otimes v.
\]

**Proof.** Let \( X = \sum_{i=1}^{m_1} X_i \otimes e_i \in \mathbb{R}^{p \times q} \otimes \mathbb{R}^{m_1} \). Then,

\[
\langle Y, \Phi_v(X) \rangle = \left\langle Y, \sum_{i=1}^{m_1} v_i X_i \right\rangle
\]

\[
= \sum_{i=1}^{m_1} \langle Y, v_i X_i \rangle
\]

\[
= \sum_{i=1}^{m_1} (v_i Y) \otimes e_i, X
\]

\[
= \left( Y \otimes \sum_{i=1}^{m_1} v_i e_i, X \right)
\]

\[
= (Y \otimes v, X).
\]

This implies that \( \Phi_v^* \cdot Y = Y \otimes v \) since the above equations are true for any \( X \).

### 3.1.2 Convolution Operator

The \( C \) operator in (10) is known as the *convolution* operator. The convolution \( C \in \mathcal{L}(\mathbb{R}^{p \times q} \otimes \mathbb{R}^{m_2}, \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_1}; \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2}) \) is defined as:

\[
C(W, X) = \sum_{a=1}^{m_2} C_a(W, X) \otimes \tilde{e}_a,
\]

where \( C_a \in \mathcal{L}(\mathbb{R}^{p \times q} \otimes \mathbb{R}^{m_2}, \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_1}; \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2}) \) is a bilinear operator that defines the mechanics of the convolution. The specific form of \( C_a \) is defined using (11) and (15) as follows:

\[
C_a(W, X) = \sum_{j=1}^{n_1} \sum_{k=1}^{n_1} \left( W_a, \Phi_{A_a} \left( \kappa_{1+(j-1)\Delta,1+(k-1)\Delta}(X) \right) \right) \tilde{E}_{jk},
\]

(16)

with \( W = \sum_{a=1}^{m_1} W_a \otimes \tilde{e}_a \). The fixed vectors \( \{ A_a \}_{a=1}^{m_2} \), where \( A_a \in \mathbb{R}^{m_1} \) for each \( a \), define the action of \( \Phi_{A_a} \) and thus the mixing of feature maps. The choice of \( \Delta \) defines the *stride* of the convolution.

The adjoints of the operators \( (C \circ X), (W \circ C) \), and \( (W \circ C_a) \) will be used in gradient calculations. The following theorems describe how to calculate them:
Theorem 3. Let \( Y = \sum_{a=1}^{m_2} Y_a \otimes \tilde{e}_a \in \mathbb{R}^n_1 \times \ell \otimes \mathbb{R}^{m_2} \) and \( X = \sum_{i=1}^{m_1} X_i \otimes e_i \in \mathbb{R}^n_1 \times \ell \otimes \mathbb{R}^{m_1} \). Then,

\[
(C \perp X)^* \cdot Y = \sum_{a=1}^{m_2} \left\{ \sum_{j=1}^{n_1} \sum_{k=1}^{\ell} (Y_a, \tilde{E}_{jk}) \Phi_{A_a} (\mathcal{K}_{1+(j-1)\Delta,1+(k-1)\Delta} (X)) \right\} \otimes \tilde{e}_a.
\]

Proof. Let \( U = \sum_{a=1}^{m_2} U_a \otimes \tilde{e}_a \in \mathbb{R}^{p \times q} \otimes \mathbb{R}^{m_2} \). Then,

\[
\langle Y, (C \perp X) \cdot U \rangle = \langle Y, C(U, X) \rangle
\]

\[
= \sum_{a=1}^{m_2} \left( \sum_{j=1}^{n_1} \sum_{k=1}^{\ell} (Y_a, \tilde{E}_{jk}) \Phi_{A_a} (\mathcal{K}_{1+(j-1)\Delta,1+(k-1)\Delta} (X)) \right) \otimes \tilde{e}_a, U \right).
\]

Since this is true for any \( U \), the proof is complete. \( \square \)

Theorem 4. Let \( W = \sum_{a=1}^{m_2} W_a \otimes \tilde{e}_a \in \mathbb{R}^{p \times q} \otimes \mathbb{R}^{m_2} \) and \( Y \in \mathbb{R}^n_1 \times \ell \). Then,

\[
(W \perp C_a)^* \cdot Y = \sum_{a=1}^{m_2} (W \perp C_a)^* \cdot Y_a.
\]

Furthermore, for any \( Z = \sum_{a=1}^{m_2} Z_a \otimes \tilde{e}_a \in \mathbb{R}^n_1 \times \ell \otimes \mathbb{R}^{m_2} \),

\[
(W \perp C)^* \cdot Z = \sum_{a=1}^{m_2} (W \perp C_a)^* \cdot Z_a.
\]

Proof. Let \( X = \sum_{i=1}^{m_1} X_i \otimes e_i \in \mathbb{R}^n_1 \times \ell \otimes \mathbb{R}^{m_1} \). Then,

\[
\langle Y, (W \perp C_a) \cdot X \rangle = \langle Y, C_a(W, X) \rangle
\]

\[
= \sum_{j=1}^{n_1} \sum_{k=1}^{\ell} \langle W_{a_1}, \Phi_{A_a} \cdot \mathcal{K}_{1+(j-1)\Delta,1+(k-1)\Delta} (X) \rangle \langle Y, \tilde{E}_{jk} \rangle
\]

Also,

\[
\langle Z, (W \perp C) \cdot X \rangle = \langle Z, C(W, X) \rangle
\]

\[
= \sum_{a=1}^{m_2} \left( \sum_{j=1}^{n_1} \sum_{k=1}^{\ell} \langle W_{a_1}, \Phi_{A_a} \cdot \mathcal{K}_{1+(j-1)\Delta,1+(k-1)\Delta} (X) \rangle \right) \langle Z, X \rangle
\]

Both of the above results are true for a generic \( X \in \mathbb{R}^n_1 \times \ell \otimes \mathbb{R}^{m_1} \), which completes the proof. \( \square \)
3.1.3 Elementwise Nonlinearity

The $S$ operator in (10) is an elementwise nonlinear function, $S: \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2} \to \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2}$, that operates as follows:

$$S \left( \sum_{a=1}^{m_2} Y_a \odot \hat{e}_a \right) = \sum_{a=1}^{m_2} \sigma(Y_a) \odot \hat{e}_a,$$

(17)

where $\sigma: \mathbb{R}^{n_1 \times \ell_1} \to \mathbb{R}^{n_1 \times \ell_1}$ is some elementwise nonlinear function, which can be written as $\sigma(Y_a) = \sum_{j=1}^{n_1} \sum_{k=1}^{\ell_1} \tilde{\sigma}((Y_a, \tilde{E}_{jk})) \tilde{E}_{jk}$. The map $\sigma: \mathbb{R} \to \mathbb{R}$ defines the nonlinear action. Common choices for $\tilde{\sigma}$ include the ramp function $\max(0, x)$ (also known as the rectifier), the sigmoidal function, or hyperbolic tangent, for example.

Some more maps are defined to assist in the calculation of the derivative $DS$ of $S$. The elementwise first and second derivatives, $S'$ and $S''$, are maps of the same dimension as $S$, defined with $\tilde{\sigma}$ replaced by $\tilde{\sigma}'$ and $\tilde{\sigma}''$ in the above formulation, respectively. Furthermore, consider a bilinear map $\odot \in \mathcal{L}(\mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2}, \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2}; \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2})$ that operates on $v = \sum_{a=1}^{m_2} v_a \odot \hat{e}_a$ and $w = \sum_{a=1}^{m_2} w_a \odot \hat{e}_a$ — both in $\mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2}$ — according to:

$$v \odot w = \sum_{a=1}^{m_2} \left( \sum_{j=1}^{n_1} \sum_{k=1}^{\ell_1} (v_a, \tilde{E}_{jk})(w_a, \tilde{E}_{jk}) \tilde{E}_{jk} \right) \odot \hat{e}_a.$$

This is an extension of the Hadamard product to the tensor product space. The map $DS$ and its adjoint are now easy to calculate.

**Theorem 5.** For any $v$ and $z \in \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2}$,

$$DS(z) \cdot v = S'(z) \odot v.$$

Furthermore, $DS(z)$ is self-adjoint, i.e. $D^* S(z) = DS(z)$.

**Proof.** Let $z = \sum_{a=1}^{m_2} z_a \odot \hat{e}_a$ and $v = \sum_{a=1}^{m_2} v_a \odot \hat{e}_a$, where $z_a, v_a \in \mathbb{R}^{n_1 \times \ell_1}$ for each $a$. Then,

$$DS(z) \cdot v = \frac{d}{dt} S(z + tv) \bigg|_{t=0}$$

$$= \frac{d}{dt} \sum_{a=1}^{m_2} \left( \sum_{j=1}^{n_1} \sum_{k=1}^{\ell_1} \tilde{\sigma}'((z_a + t v_a, \tilde{E}_{jk}))(\tilde{E}_{jk}) \right) \odot \hat{e}_a \bigg|_{t=0}$$

$$= \sum_{a=1}^{m_2} \left( \sum_{j=1}^{n_1} \sum_{k=1}^{\ell_1} \tilde{\sigma}'((z_a, \tilde{E}_{jk}))(v_a, \tilde{E}_{jk}) \tilde{E}_{jk} \right) \odot \hat{e}_a$$

$$= S'(z) \odot v,$$

where the final line follows from the definition of the Hadamard product and the elementwise first derivative $S'(z)$. To prove that $DS(z)$ is self-adjoint, first note that it is not hard to show that $(y, v \odot w) = (v \odot y, w)$, for any $v, w$, and $y$ in the same space. Thus, for any $y \in \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2}$,

$$(y, DS(z) \cdot v) = (y, S'(z) \odot v)$$

$$= (S'(z) \odot y, v)$$

$$= (DS(z) \cdot y, v).$$

This proves that $D^* S(z) = DS(z)$. \qed

3.1.4 Pooling Operator

The $\Psi$ operator in (10) is known as the pooling operator, and its purpose is to reduce the size of the feature maps at each layer. Only linear pooling is considered in this paper (the framework does extend to the nonlinear case though), so that $\Psi \in \mathcal{L}(\mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_2}, \mathbb{R}^{n_2 \times \ell_2} \otimes \mathbb{R}^{m_2})$ operates as:

$$\Psi \left( \sum_{a=1}^{m_2} Y_a \odot \hat{e}_a \right) = \sum_{a=1}^{m_2} \psi(Y_a) \odot \hat{e}_a$$

(18)
The derivatives of a generic layer

The adjoints of the above operators can be calculated using the reversing property of the adjoint

Theorem 7.

One type of linear pooling is average pooling, which involves taking the average over all elements in the \( r \times r \) neighbourhoods. This can be represented using (12) as:

\[
\psi(Y_a) = \frac{1}{r^2} \sum_{j=1}^{m_2} \sum_{k=1}^{m_3} \left( \sum_{r=1}^{r} (1, \kappa_{1+(j-1)r,1+(k-1)r}(Y_a)) \tilde{E}_{jk} \right),
\]

where the operator \( \kappa_{jk} \in \mathcal{L}(\mathbb{R}^{n_1 \times \ell_1}; \mathbb{R}^{r \times r}) \) is defined in (12) with \( p = q = r \) and

\[
1_r = \sum_{j=1}^{r} \sum_{k=1}^{r} \tilde{E}_{jk} \in \mathbb{R}^{r \times r}.
\]

If \( \{\tilde{E}_{jk}\}_{jk} \) is the standard basis, \( 1_r \) is the all-ones matrix.

The adjoint \( \Psi^* \) of the average pooling operator \( \Psi \) can be computed using the following theorem.

**Theorem 6.** Let \( Z = \sum_{a=1}^{m_2} Z_a \otimes \tilde{e}_a \in \mathbb{R}^{n_2 \times \ell_2} \otimes \mathbb{R}^{m_3} \). Then, using (13) with \( \text{Em}_{c,d} : \mathbb{R}^{r \times r} \rightarrow \mathbb{R}^{n_1 \times \ell_1} \),

\[
\Psi^* \cdot Z = \frac{1}{r^2} \sum_{a=1}^{m_2} \left( \sum_{j=1}^{m_2} \sum_{k=1}^{m_3} \sum_{r=1}^{r} (Z_a, \tilde{E}_{jk}) \text{Em}_{1+(j-1)r,1+(k-1)r}(1_r) \right) \otimes \tilde{e}_a.
\]

**Proof.** First, let \( \gamma_{jk} = (1 + (j-1)r,1 + (k-1)r) \) for notational convenience. Then, for any \( Y = \sum_{a=1}^{m_2} Y_a \otimes \tilde{e}_a \in \mathbb{R}^{n_1 \times \ell_1} \otimes \mathbb{R}^{m_3} \),

\[
(Z, \Psi \cdot Y) = \sum_{a=1}^{m_2} \left( Z_a, \frac{1}{r^2} \sum_{j=1}^{m_2} \sum_{k=1}^{m_3} \left(1_r, \kappa_{\gamma_{jk}}(Y_a) \right) \tilde{E}_{jk} \right)
\]

\[
= \frac{1}{r^2} \sum_{a=1}^{m_2} \sum_{j=1}^{m_2} \sum_{k=1}^{m_3} \left( Z_a, \tilde{E}_{jk} \right) \left(1_r, \kappa_{\gamma_{jk}}(Y_a) \right)
\]

\[
= \frac{1}{r^2} \sum_{a=1}^{m_2} \sum_{j=1}^{m_2} \sum_{k=1}^{m_3} \left( Z_a, \tilde{E}_{jk} \right) \kappa_{\gamma_{jk}}(1_r, Y_a)
\]

\[
= \frac{1}{r^2} \sum_{a=1}^{m_2} \left( \sum_{j=1}^{m_2} \sum_{k=1}^{m_3} \left( Z_a, \tilde{E}_{jk} \right) \text{Em}_{\gamma_{jk}}(1_r) \right) \otimes \tilde{e}_a, Y_a.
\]

Since this is true for any \( Y \), the proof is complete. \( \square \)

### 3.1.5 Single-Layer Derivatives

The derivatives of a generic layer \( f(X; W, B) \), as described in (10), with respect to \( X, W, \) and \( B \) are presented in the following theorem.

**Theorem 7.**

1. \( Df(X; W, B) = \Psi \cdot DS(C(W, X) + B) \cdot (W \cup C) \).
2. \( \nabla_w f(X; W, B) = \Psi \cdot DS(C(W, X) + B) \cdot (C \cup X) \).
3. \( \nabla_B f(X; W, B) = \Psi \cdot DS(C(W, X) + B) \).

**Proof.** These are all direct consequences of the chain rule and linearity of the derivative for the function \( f \) given in (10). \( \square \)

The adjoints of the above operators can be calculated using the reversing property of the adjoint operator \( * \).
Theorem 8.

1. \( D^* f(X;W,B) = (W \cup C)^* \cdot DS(C(W,X) + B) \cdot \Psi^* \).
2. \( \nabla^*_W f(X;W,B) = (C \cup X)^* \cdot DS(C(W,X) + B) \cdot \Psi^* \).
3. \( \nabla^*_B f(X;W,B) = DS(C(W,X) + B) \cdot \Psi^* \).

3.2 Multiple Layers

Suppose now that the network consists of \( L \) layers. Denote the actions of the \( t \)th layer as \( X^{t+1} = f_t(X^t) \), where \( X^t \in \mathbb{R}^{n_t \times \ell_t} \otimes \mathbb{R}^{m_t} \) and \( X^1 \) is one point in the input data. The layer map \( f_t : \mathbb{R}^{n_t \times \ell_t} \otimes \mathbb{R}^{m_t} \to \mathbb{R}^{n_{t+1} \times \ell_{t+1}} \otimes \mathbb{R}^{m_{t+1}} \) can be given explicitly as:

\[
 f_t(X^t) = \Psi_t \left( S_t(C^t(W^t, X^t) + B^t) \right). \tag{20}
\]

Here, \( W^t \in \mathbb{R}^{n_t \times q_t} \otimes \mathbb{R}^{m_{t+1}} \) and \( B^t \in \mathbb{R}^{n_t \times \ell_t} \otimes \mathbb{R}^{m_{t+1}} \). Note that the pooling operator \( \Psi_t \), the nonlinearity \( S_t \), and the convolution operator \( C^t \) are layer-dependent. The entire network’s actions can be denoted as:

\[
 F(X;\theta) := f_L \circ \cdots \circ f_1(X),
\]

where \( \theta := \{W^1, \ldots, W^L, B^1, \ldots, B^L\} \) is the parameter set and \( X \equiv X^1 \) is the input data.

3.2.1 Final Layer

Classification is often the goal of a CNN, thus assume that there are \( N \) classes. This implies the following: \( m_{L+1} = N, n_L = \ell_L = n_{L+1} = \ell_{L+1} = 1, \) and \( F(X;\theta) \in \mathbb{R}^N \). The final layer is assumed to be fully connected, which aligns with the form given in (20) if the cropping operator (11) and pooling operator (18) for the final layer — \( K_{f_L}^f \) and \( \Psi_L \) — are identity maps. Also, \( A^0_a \in \mathbb{R}^{m_a} \) defining the mixing operator \( \Phi_{A^L_a}^f \) in (10) is \( A^{L_a} = \sum^{m_a} \epsilon_a^i \) for each \( a \). Then, the final layer is given as:

\[
 f_L(X^L) = S_L(C^L(W^L, X^L) + B^L) \equiv \sum^{N}_{a=1} \sigma_L \left(C^L_a(W^L, X^L) + B^L_a\right) \epsilon_a^{L+1},
\]

where \( \{\epsilon_a^{L+1}\}_a \) is a basis for \( \mathbb{R}^N \), and \( C^L_a(W^L, X^L) = \sum^{m_a}_{i=1} (W^L_a, X^L) \). Note that \( \sigma_L : \mathbb{R} \to \mathbb{R} \). It is also important to note that this shows that simpler, fully-connected neural networks are just a special case of convolutional neural networks.

3.3 Loss Function & Backpropagation

While training a CNN, the goal is to optimize some loss function \( J \) with respect to the parameters \( \theta \). For example, consider

\[
 J(X;\theta) := \frac{1}{2} \|y - F(X;\theta)\|^2 - \frac{1}{2}(y - F(X;\theta), y - F(X;\theta)),
\]

where \( y \) represents the given data and \( F(X;\theta) \) is the prediction. Gradient descent is used to optimize the loss function, so it is important to calculate the gradient of \( J \) with respect to each of the parameters. For this, define maps \( \omega_t \) and \( \alpha_t \) as:

\[
 \omega_t := f_L \circ \cdots \circ f_t, \quad \alpha_t := f_t \circ \cdots \circ f_1 \tag{21}
\]

for \( t = 1, \ldots, L \), which satisfy (9). Assume \( \omega_{L+1} \) and \( \alpha_0 \) are identity maps for the sake of convenience. Then, for any \( U^t \in \mathbb{R}^{n_t \times \ell_t} \otimes \mathbb{R}^{m_t} \),

\[
 \nabla_{W^t} J(X;\theta) : U^t = \langle F(X;\theta) - y, \nabla_{W^t} F(X;\theta) \cdot U^t \rangle = \langle \nabla_{W^t} F(X;\theta) \cdot (F(X;\theta) - y), U^t \rangle.
\]

Since this holds for any \( U^t \),

\[
 \nabla_{W^t} J(X;\theta) = \nabla_{W^t}^* F(X;\theta) \cdot (F(X;\theta) - y) \tag{22}
\]

by the same logic used to derive (6) from (3). Differentiating \( F(X;\theta) = \omega_{t+1} \circ f_t \circ \alpha_{t-1}(X) \) with respect to \( W^t \) produces

\[
 \nabla_{W^t} F(X;\theta) = D \omega_{t+1}(X^{t+1}) \cdot \nabla_{W^t} f_t(X^t), \tag{23}
\]
where \( X^t = \alpha_{t-1}(X) \) and \( X^{t+1} = f_t(X^t) = \alpha_t(X) \). Taking the adjoint of (23) yields
\[
\nabla_{W_t}^* F(X; \theta) = \nabla_{W_t}^* f_t(X^t) \cdot D^* \omega_{t+1}(X^{t+1}),
\]
which can be substituted into (22). Then, the final step in computing (22) involves computing \( D^* \omega_{t+1} \) in (24), which can be done recursively:
\[
D^* \omega_t(X^t) = D^* f_t(X^t) \cdot D^* \omega_{t+1}(X^{t+1}).
\]

This comes from taking the derivative and then the adjoint of the relationship \( \omega_t = \omega_{t+1} \circ f_t \). Note that \( \nabla_{W_t} f_t(X^t) \) and \( D^* f_t(X^t) \) in (24) and (25) are calculated using Theorem 8. Since \( \nabla_{W_t} J(X; \theta) \) can be calculated, gradient descent can be performed. One iteration of a gradient descent algorithm to update \( B_t \) and \( W_t \) for all \( t \in \{1, \ldots, L\} \) is given in Algorithm 3.1. The method for calculating \( \nabla_{B_t} J(X; \theta) \) is not explicitly shown in the derivation, but is a simpler version of \( \nabla_{W_t} J(X; \theta) \) and is included in the algorithm. The algorithm can be extended to a batch of points by summing the contribution to \( \nabla J \) from each input point \( X \). Note that \( \eta \in \mathbb{R} \) is the learning rate.

**Algorithm 3.1** One iteration of gradient descent for a CNN

```plaintext
function DESCENT ITERATION(X, y, W^1, \ldots, W^L, B^1, \ldots, B^L, \eta)

X^1 ← X

for t ∈ \{1, \ldots, L\} do
  \( Z^t \leftarrow C^t(W^t, X^t) + B^t \)
  \( X^{t+1} ← \Psi_t(S_t(Z^t)) \)
end for

for t ∈ \{L, L-1, \ldots, 1\} do
  W^t ← W^t
  if t = L then
    e ← x^{L+1} - y
  else
    e ← (W^{t+1} \cdot C^{t+1})^* \cdot (S^t_{t+1}(Z^{t+1}) \circ (\Psi_{t+1}^* \cdot e))
  end if
  \( e \leftarrow D^* \omega_{t+1}(X^{t+1}) \cdot (X^{L+1} - y) \)
  end if
  \( \nabla_{B_t} J(X; \theta) \leftarrow (S^t_{t+1}(Z^{t+1}) \circ (\Psi_{t+1}^* \cdot e)) \)
  \( \nabla_{W_t} J(X; \theta) \leftarrow (C^t \cdot X^t)^* \cdot (S^t_{t+1}(Z^{t+1}) \circ (\Psi_{t+1}^* \cdot e)) \)
  \( B^t ← B^t - \eta \nabla_{B_t} J(X; \theta) \)
  \( W^t ← W^t - \eta \nabla_{W_t} J(X; \theta) \)
end for

end function
```

### 3.4 Higher-Order Loss Functions

Suppose that another term is added to the loss function to penalize the first-order derivative of \( F(X; \theta) \), as in [7] or [8] for example. This can be represented using
\[
\begin{align*}
R(X; \theta) := & \frac{1}{2} \|DF(X; \theta) \cdot V_X - \beta_X \|^2,
\end{align*}
\]
for some \( V_X \in \mathbb{R}^{n \times d} \otimes \mathbb{R}^{m_1} \) and \( \beta_X \in \mathbb{R}^N \). When \( \beta_X = 0 \), minimizing \( R(X, \theta) \) promotes invariance of the network in the direction of \( V_X \). This can be useful in image classification, for example, where the class of image is expected to be invariant with respect to rotation. In this case, \( V_X \) would be an infinitesimal generator of rotation. This new term \( R \) can be added to \( J \) to create a new loss function
\[
J := J + \lambda R,
\]
where \( \lambda \in \mathbb{R} \) determines the amount that the higher-order term contributes to the loss function. Note that \( R \) could be extended to contain multiple terms as:
\[
R(X; \theta) = \sum_{(V_X; \beta_X) \in \mathcal{G}_\lambda} \frac{1}{2} \|DF(X; \theta) \cdot V_X - \beta_X \|^2,
\]
where \( B_X \) is a finite set of pairs \((V_X, \beta_X)\) for each \( X \).

The gradient of \( R \) with respect to the parameters must now be taken. This can calculated for a generic parameter \( \theta^t \), which is one of \( W^t \) or \( B^t \):

\[
\nabla_{\theta^t} R(X; \theta) \cdot U^t = (DF(X; \theta) \cdot V_X - \beta_X, (\nabla_{\theta^t} DF(X; \theta) \cdot V_X) \cdot U^t),
\]

for all \( U^t \) in the same space as \( \theta^t \). Again, in the same way that (6) was derived from (5),

\[
\nabla_{\theta^t} R(X; \theta) = (\nabla_{\theta^t} DF(X; \theta) \cdot V_X)^* \cdot (DF(X; \theta) \cdot V_X - \beta_X).
\]

Before (28) can be computed, however, some preliminary results will be given.

**Theorem 9.** Let \( f \) be defined as in (10), and \( V \in \mathbb{R}^{n_l \times \ell_1} \otimes \mathbb{R}^{m_1} \). Let \( Z = C(W, X) + B \). Then,

\[
\begin{align*}
(V \downarrow D\nabla_W f(X; W, B)) &= \Psi \cdot \left( C(W, V) \downarrow D^2 S(Z) \right) \cdot (C \downarrow X) + \Psi \cdot D S(Z) \cdot (C \downarrow V), \\
(V \downarrow D\nabla_B f(X; W, B)) &= \Psi \cdot \left( C(W, V) \downarrow D^2 S(Z) \right), \\
(V \downarrow D^2 f(X; W, B)) &= \Psi \cdot \left( C(W, V) \downarrow D^2 S(Z) \right) \cdot (W \downarrow C).
\end{align*}
\]

**Proof.** Let \( U \in \mathbb{R}^{p \times q} \otimes \mathbb{R}^{m_2} \). Then, prove (29) directly:

\[
\begin{align*}
(V \downarrow D\nabla_W f(X; W, B) \cdot U) \cdot V &= D \left[ (\nabla_W f(X; W, B) \cdot U) \cdot V \right] \\
&= \Psi \cdot D^2 S(Z) \cdot (C(W, V)) + \Psi \cdot D S(Z) \cdot (C \downarrow V) \\
&= \Psi \cdot \left( C(W, V) \downarrow D^2 S(Z) \right) \cdot (W \downarrow C) \cdot V.
\end{align*}
\]

This is true for any \( U \), so equation (29) is proven. Equation (30) can be proven similarly, so its proof is omitted. Also, let \( \tilde{V} \in \mathbb{R}^{n_l \times \ell_1} \otimes \mathbb{R}^{m_1} \). Then, equation (31) can also be proven directly:

\[
\begin{align*}
(V \downarrow D^2 f(X; W, B)) &= \Psi \cdot \left( C(W, V) \downarrow D^2 S(Z) \right) \cdot (W \downarrow C) \cdot \tilde{V}.
\end{align*}
\]

This is true for any \( \tilde{V} \), so the proof is completed. \( \square \)

The next lemma shows how to actually calculate \( D^2 S(Z) \) so that the above equations can be computed.

**Lemma 10.** For any \( X, V \) and \( \tilde{V} \in \mathbb{R}^{n_l \times \ell_1} \otimes \mathbb{R}^{m_2} \) with \( S \) defined in (17),

\[
D^2 S(X) \cdot (V \downarrow \tilde{V}) = S''(X) \circ V \circ \tilde{V},
\]

where \( S'' \) is defined similarly to \( S \), but with \( \sigma'' \) replacing \( \sigma \). Furthermore, \( (V \downarrow D^2 S(X)) \) is self-adjoint, i.e. \( (V \downarrow D^2 S(X))^* = (V \downarrow D^2 S(X)) \).

**Proof.** From the definition of the second derivative,

\[
\begin{align*}
D^2 S(X) \cdot (V \downarrow \tilde{V}) &= D \left( D S(X) \cdot \tilde{V} \right) \cdot V \\
&= D \left( S'(X) \circ \tilde{V} \right) \cdot V \\
&= (S''(X) \circ \tilde{V}) \circ V,
\end{align*}
\]

where the last equality follows from viewing \( S'(X) \circ \tilde{V} \) as an elementwise function in \( X \). As for the adjoint, let \( Y \in \mathbb{R}^{n_l \times \ell_1} \otimes \mathbb{R}^{m_2} \). Then,

\[
\langle Y, (V \downarrow D^2 S(X)) \cdot \tilde{V} \rangle = \langle Y, S''(X) \circ V \circ \tilde{V} \rangle = \langle (S''(X) \circ V \circ \tilde{V}) \cdot Y, \tilde{V} \rangle.
\]

This proves that \((V \downarrow D^2 S(X))\) is self-adjoint. \( \square \)
The adjoints of the equations in Theorem 9 can now easily be calculated using the above lemma and the reversing property of the adjoint operator.

**Theorem 11.** Let \( f \) be defined as in \([10]\), and \( V \in \mathbb{R}^{n_1 \times f_1} \otimes \mathbb{R}^{m_1} \). Let \( Z = C(W, X) + B \). Then,

\[
(V \cdot D \nabla W f(X; W, B))^* = (C \cdot X)^* \cdot (C(W, V) \cdot D^2 S(Z)) \cdot \Psi^* + (C \cdot V)^* \cdot DS(Z) \cdot \Psi^*,
\]

\[
(V \cdot D \nabla B f(X; W, B))^* = (C(W, V) \cdot D^2 S(Z)) \cdot \Psi^*,
\]

\[
(V \cdot D^2 f(X; W, B))^* = (W \cdot C)^* \cdot (C(W, V) \cdot D^2 S(Z)) \cdot \Psi^*.
\]

Now, propagation through the network can be described in the spirit of \([8]\). *Forward* propagation through the network can be computed recursively, using \( \alpha_t = f_t \circ \alpha_{t-1} \):

\[
D\alpha_t(X) = Df_t(X^t) \cdot D\alpha_{t-1}(X),
\]

for any \( t \in \{1, \ldots, L\} \) and \( X \in \mathbb{R}^{n_1 \times f_1} \otimes \mathbb{R}^{m_1} \). *Backward* propagation through the network is described in the next theorem.

**Theorem 12.** Let \( f_t \) be defined as in \([20]\) and \( \omega_t \) and \( \alpha_t \) be defined as in \([21]\). Then, for any \( X, V \in \mathbb{R}^{n_1 \times f_1} \otimes \mathbb{R}^{m_1} \), and \( t \in \{1, \ldots, L\} \),

\[
\left( (D\alpha_{t-1}(X) \cdot V) \cdot D^2 \omega_t(X^t) \right)^* = D^* f_t(X^t) \cdot \left( (D\alpha_t(X) \cdot V) \cdot D^2 \omega_{t+1}(X^{t+1}) \right)^* + \left( (D\alpha_{t-1}(X) \cdot V) \cdot D^2 f_t(X^t) \right)^* \cdot D^* \omega_{t+1}(X^{t+1}),
\]

where \( X^t = \alpha_{t-1}(X) \). Also, \( (D\alpha_L(X) \cdot V) \cdot D^2 \omega_{L+1}(X^{L+1}) \) is the zero operator.

**Proof.** Since \( \omega_{L+1} \) is the identity, its second derivative is the zero operator. Now consider the case when \( t \in \{1, \ldots, L\} \). Take any \( \tilde{X}, \tilde{Y} \), and \( Y \in \mathbb{R}^{n_1 \times f_1} \otimes \mathbb{R}^{m_1} \). Then,

\[
\left( \tilde{Y} \cdot D^2 \omega_t(\tilde{X}) \right) \cdot Y = D \left( D(\omega_{t+1} \circ f_t)(\tilde{X}) \cdot Y \cdot \tilde{Y} \right)
\]

\[
= D(\omega_{t+1}(f_t(\tilde{X}))) \cdot Df_t(\tilde{X}) \cdot \tilde{Y} \cdot Y + D\omega_{t+1}(f_t(\tilde{X})) \cdot D^2 f_t(\tilde{X}) \cdot \tilde{Y} \cdot Y + D\omega_{t+1}(f_t(\tilde{X})) \cdot (\tilde{Y} \cdot D^2 f_t(\tilde{X})) \cdot Y,
\]

where the third equality follows from the product rule. Removing the trailing \( Y \) from both sides, and setting \( \tilde{Y} = D\alpha_{t-1}(X) \cdot V \) and \( \tilde{X} = \alpha_{t-1}(X) = X^t \),

\[
\left( (D\alpha_{t-1}(X) \cdot V) \cdot D^2 \omega_t(X^t) \right) = \left( (D\alpha_t(X) \cdot V) \cdot D^2 \omega_{t+1}(X^{t+1}) \right) \cdot Df_t(X^t) + D\omega_{t+1}(X^{t+1}) \cdot \left( (D\alpha_{t-1}(X) \cdot V) \cdot D^2 f_t(X^t) \right),
\]

since \( D\alpha_t(X) = Df_t(X^t) \cdot D\alpha_{t-1}(X) \) and \( X^{t+1} = f_t(X^t) = \alpha_t(X) \). Taking the adjoint of this result completes the proof. \( \square \)

Note that calculating (33) involves taking the adjoint of (31), which can be done using Theorem 11 along with Theorems 4 and 6 and Lemma 10. The above results are crucial for the next theorem, which is the main result.

**Theorem 13.** Suppose \( V \) and \( X \in \mathbb{R}^{n_1 \times f_1} \otimes \mathbb{R}^{m_1} \), \( t \in \{1, \ldots, L\} \), and \( F, \alpha_t, \) and \( \omega_t \) are defined as in \([9]\). Then, for a generic parameter \( \theta \in \{W^t, B^t\} \),

\[
(\nabla_\theta DF(X; \theta) \cdot V)^* = \nabla_{\theta}^* f_t(X^t) \cdot \left( (D\alpha_t(X) \cdot V) \cdot D^2 \omega_{t+1}(X^{t+1}) \right)^* + \left( (D\alpha_{t-1}(X) \cdot V) \cdot D\nabla_{\theta} f_t(X^t) \right)^* \cdot D^* \omega_{t+1}(X^{t+1}),
\]

where \( X^t = \alpha_{t-1}(X) \).
Proof. For any $U$ in the same space as $\theta^t$,
\[
(\nabla_{\theta^t} DF(X;\theta) \cdot V) \cdot U = D(\nabla_{\theta^t} F(X;\theta) \cdot U) \cdot V \\
= D\left( D(\omega_{t+1}(\alpha_t(X)) \cdot V \cdot f_t(\alpha_{t-1}(X))) \cdot U \right) \\
= D^2(\omega_{t+1}(\alpha_t(X)) \cdot (D\alpha_t(X) \cdot V \cdot f_t(\alpha_{t-1}(X))) \cdot U) \\
+ D\omega_{t+1}(\alpha_t(X)) \cdot (D\nabla_{\theta^t} f_t(\alpha_{t-1}(X)) \cdot (D\alpha_{t-1}(X) \cdot V, U) \\
= ((D\alpha_t(X) \cdot V) \cdot D^2(\omega_{t+1}(X^{t+1})) \cdot \nabla_{\theta^t} f_t(X^t) \cdot U) \\
+ D\omega_{t+1}(X^{t+1}) \cdot ((D\alpha_{t-1}(X) \cdot V) \cdot D\nabla_{\theta^t} f_t(X^t) \cdot U),
\]
where the final equality follows since $X^t = \alpha_{t-1}(X)$ for all $t \in \{1, \ldots, L+1\}$. Removing the trailing $U$ from both sides and taking the adjoint produces equation (34). \qed

Note that in Equation (34), $\nabla_{\theta^t} f_t$ and $D\nabla_{\theta^t} f_t$ can be replaced by their corresponding expressions in Theorem 7 and 9 respectively, once $\theta_t$ is replaced by one of $W_t$ or $B_t$. Then, (28) can be computed with Theorem 13, where $DF(X;\theta) = D\alpha_L(X)$ is computed recursively by (32). Algorithm 3.2 shows one iteration of a gradient descent algorithm to optimize $J$ defined in (26) for one point $X$. This algorithm extends to a batch of updates, and for $R$ defined with multiple $(V_X, \beta_X)$ pairs as in (27).

4 Conclusion and Future Work

This work has developed a geometric framework for convolutional neural networks. The input data and parameters are defined over a vector space equipped with an inner product. The parameters are learned using a gradient descent algorithm that acts directly over the inner product space, avoiding the use of individual coordinates. Derivatives for higher-order loss functions are also explicitly calculated in a coordinate-free manner, providing the basis for a gradient descent algorithm. This mathematical framework can be extended to other types of deep networks, including recurrent neural networks, autoencoders and deep Boltzmann machines. Another interesting future direction is to expand the capabilities of automatic differentiation (AD) into this coordinate-free realm, strengthening the hierarchical approach to AD [9].

This paper has shown how to express a particular deep neural network, end-to-end, in a precise format. However, this framework should not be limited to only expressing previous results, and it should not be written off as simply a derivative calculation method. The stronger mathematical understanding of neural networks provided by this work should promote expansion into new types of networks.

References

[1] R. Abraham, J. Marsden, and T. Ratiu. Manifolds, Tensor Analysis, and Applications (2nd edition). Springer, 1988.
[2] W. Greub. Multilinear Algebra. Springer Verlag, 1978.
[3] F. Huang and Y. LeCun. Large-scale learning with SVM and convolutional networks for generic object recognition. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006.
[4] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.
[5] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. Nature, 521(7553):436–444, 2015.
[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.
[7] S. Rifai, Y. Dauphin, P. Vincent, Y. Bengio, and X. Muller. The manifold tangent classifier. In Advances in Neural Information Processing Systems, pages 2294–2302, 2011.
[8] P. Simard, B. Victorri, Y. LeCun, and J. Denker. Tangent Prop — A formalism for specifying selected invariances in an adaptive network. In Advances in neural information processing systems, pages 895–903, 1992.
Algorithm 3.2 One iteration of gradient descent for a higher-order CNN

function DESCENT ITERATION($X, y, V_X, \beta_X, W^1, \ldots, W^L, B^1, \ldots, B^L, \eta, \lambda$) {
  
  $X^1 \leftarrow X$
  $V^1 \leftarrow V_X$
  
  for $t \in \{1, \ldots, L\}$ do 
    $Z^t \leftarrow C^t(W^t, X^t) + B^t$
    $X^{t+1} \leftarrow \Psi_t(S_t(Z^t))$
    $V^{t+1} \leftarrow \Psi_t(S_t(Z^t) \odot C^t(W^t, V^t))$
  end for 

  for $t \in \{L, \ldots, 1\}$ do
    $\tilde{W}^t \leftarrow W^t$
    if $t = L$ then
      $e_y \leftarrow \alpha^{L+1} - y$
      $e_w \leftarrow 0$
      $e_v \leftarrow \alpha^{L+1} - \beta_X$
    else
      $e_y \leftarrow (\tilde{W}^{t+1} \odot C^{t+1})^* \cdot (S_{t+1}(Z^{t+1}) \odot (\Psi_{t+1} \cdot e_y))$
      $e_w \leftarrow (\tilde{W}^{t+1} \odot C^{t+1})^* \cdot (S_{t+1}(Z^{t+1}) \odot (\Psi_{t+1} \cdot e_w))$
      $e_v \leftarrow (\tilde{W}^{t+1} \odot C^{t+1})^* \cdot (S_{t+1}(Z^{t+1}) \odot (\Psi_{t+1} \cdot e_v))$
    end if

    $\nabla_B J(X; \theta) \leftarrow (S_t(Z^t) \odot (\Psi_t \cdot e_y))$
    $\nabla_W J(X; \theta) \leftarrow (C^t \odot X^t)^* \cdot (S_t(Z^t) \odot (\Psi_t \cdot e_y))$
    $\nabla_B R(X; \theta) \leftarrow (S_t(Z^t) \odot (\Psi_t \cdot e_w)) + S''_t(Z^t) \odot C^t(W^t, V^t) \odot (\Psi_t \cdot e_v)$
    $\nabla_W R(X; \theta) \leftarrow (C^t \odot X^t)^* \cdot (S''_t(Z^t) \odot C^t(W^t, V^t) \odot (\Psi_t \cdot e_v))$
      $+ (C^t \odot V^t)^* \cdot (S_t(Z^t) \odot (\Psi_t \cdot e_v)) + (W^t \odot C^t)^* \cdot (S_t(Z^t) \odot (\Psi_t \cdot e_v))$
    ▷ Both $\nabla_B R$ and $\nabla_W R$ can be computed via Thm. 11 along with Thms. 8 and 11

    $B^t \leftarrow B^t - \eta (\nabla_B J(X; \theta) + \lambda \nabla_B R(X; \theta))$
    $W^t \leftarrow W^t - \eta (\nabla_W J(X; \theta) + \lambda \nabla_W R(X; \theta))$
  end for

end function

[9] S. Walter and L. Lehmann. Algorithmic differentiation in Python with AlgoPy. Journal of Computational Science, 4(5):334–344, 2013.