ConformalLayers: A non-linear sequential neural network with associative layers
– Supplementary Material –

Eduardo Vera Sousa, Leandro A. F. Fernandes, Cristina Nader Vasconcelos
Instituto de Computação, Universidade Federal Fluminense (UFF)
Niterói, Rio de Janeiro, Brazil – ZIP 24210–346
Email: {eduardovera, laffernandes, crisnv}@ic.uff.br

The Supplementary Material is structured as follows. In Section I, we present the algebraic manipulation that takes the ReSPro function from its formulation using the geometric algebra of the conformal model for Euclidean geometry [1] to tensor algebra. Section II shows the matrix representation of typical linear layers and other operations in CNNs. The algebraic manipulation that expands a sequence of calls to $\mathcal{L}^{(l)}$ functions, for $l \in \{1, 2, \cdots, k\}$, to produce the tensor representation of the ConformalLayers is presented in Section III. The hyperparameter values selected for each CNN and dataset used in Experiments I and II is presented in Section IV. Sections V and VI present the hyperparameter sweep. Please refer to the paper for details about the experiments and discussion on the results.

I. ReSPro: From Geometric Algebra to Tensors

We represent the discrete input data as a point $x = (x_1, x_2, \cdots, x_d) \in \mathrm{R}^d$. We include an extra dimension in R^d and embed R^{d+1} in a $(d+3)$-dimensional space with basis vectors $\{e_1, e_2, \cdots, e_{d+1}, e_o, e_\infty\}$ and metric matrix:

$\cdot$	e_1	e_2	$\cdots$	e_{d+1}	e_o	e_∞
e_1	1	0	$\cdots$	0	0	0
e_2	0	1	$\cdots$	0	0	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
e_{d+1}	0	0	$\cdots$	1	0	0
e_o	0	0	$\cdots$	0	0	-1
e_∞	0	0	$\cdots$	0	-1	0

where $\cdot$ denotes the vector inner product. In other words, we assume the conformal model for Euclidean geometry to work with our data. In this model, the finite point x is represented by the vector:

$$V = x'_1 e_1 + x'_2 e_2 + \cdots + x'_d e_d + x'_o e_o - \frac{x'_o}{2} \sum_{i=1}^{d} (x_i)^2 e_\infty, \tag{1}$$

where $x_i = x'_i / x'_o$, for $i \in \{1, 2, \cdots, d+1\}$, and $x'_o \neq 0$. By definition, we set $x_{d+1} = 0$. The extra dimensions e_o and e_∞ are geometrically interpretable as the point at the origin and the point at infinity, respectively.

We construct a geometric algebra over the conformal space. This allows operations on the base space, including reflections, rotations, and translations to be represented using versors of the geometric algebra [1]. Therefore, from now on, the mathematical expressions will be written using geometric algebra. The terms in the expressions that follow represent blades or versors encoded as multivectors in the multivector space $\bigwedge \mathbb{R}^{d+3}$. The half-space denotes the geometric product, while $\wedge$ and $\rfloor$ denote, respectively, the outer (wedge) product and the left contraction.

Let V' be a 2-blade encoding a planar point compute from V and e_∞ using the outer product:

$$\begin{aligned} V' = V \wedge e_\infty &= (x'_1 e_1 + x'_2 e_2 + \cdots + x'_d e_d + x'_o e_o + x'_\infty e_\infty) \wedge e_\infty \\ &= (x'_1 e_1 + x'_2 e_2 + \cdots + x'_d e_d + x'_o e_o) \wedge e_\infty \\ &= v \wedge e_\infty + x'_o \, (e_o \wedge e_\infty) \\ &= (v + x'_o e_o) \wedge e_\infty, \end{aligned} \tag{2}$$

where $v = x'_1 e_1 + x'_2 e_2 + \cdots + x'_d e_d$, and $x'_\infty = -\frac{x'_o}{2} \sum_{i=1}^{d} (x_i)^2$.

The reflection of V' in a hypersphere with center $c = (0, 0, \cdots, \alpha) \in \mathrm{R}^{d+1}$ and radius α, for $\alpha \geq 0$, followed by isotropic scaling by a factor of $2/\alpha$, is done by applying the 3-versor T to V', which leads to the 2-blade V'' encoding a pair of points where the second point of the pair corresponds to the center of the hypersphere properly transformed by scaling. The versor T is given by:

$$T = \left(\cosh\left(\frac{1}{2}\log\frac{2}{\alpha}\right) + \sinh\left(\frac{1}{2}\log\frac{2}{\alpha}\right)(e_o \wedge e_\infty)\right)(\alpha e_{d+1} + e_o), \tag{3}$$

whose inverse is:

$$T^{-1} = \left(\frac{1}{\alpha}e_{d+1} + \frac{1}{\alpha^2}e_o\right)\left(\cosh\left(\frac{1}{2}\log\frac{2}{\alpha}\right) - \sinh\left(\frac{1}{2}\log\frac{2}{\alpha}\right)(e_o \wedge e_\infty)\right). \tag{4}$$

The resulting pair of points V'' is computed using the versor product of V' by T:

$$\begin{aligned} V'' = T\,V'\,T^{-1} &= T\,(V \wedge e_\infty)\,T^{-1} \\ &= \left(-T\,V\,T^{-1}\right) \wedge \left(-T\,e_\infty\,T^{-1}\right) \\ &= \frac{2}{\alpha}(v \wedge e_{d+1}) + \frac{1}{\alpha}(v \wedge e_o) + \frac{2}{\alpha}(v \wedge e_\infty) - x'_o(e_{d+1} \wedge e_o) + x'_o(e_o \wedge e_\infty). \end{aligned} \tag{5}$$

By contracting e_∞ on V'' one gets the dual of the perpendicular bisector of the pair of point V'':

$$\begin{aligned} H = e_\infty \rfloor V'' &= e_\infty \rfloor \left(\frac{2}{\alpha}(v \wedge e_{d+1}) + \frac{1}{\alpha}(v \wedge e_o) + \frac{2}{\alpha}(v \wedge e_\infty) - x'_o(e_{d+1} \wedge e_o) + x'_o(e_o \wedge e_\infty)\right) \\ &= \frac{1}{\alpha}v - x'_o e_{d+1} - x'_o e_\infty. \end{aligned} \tag{6}$$

The bisector can be used as a "mirror" to reflect the known point of the pair to find the unknown point. The computed point is finite and represents the input point transformed by spherical projection followed by scaling. We use the versor product to compute where the point at infinity (*i.e.,* the known point in the original pair V') was mapped by T. Next we use an up-to-scaling versor product to find V''' as the reflection of the transformed point at infinity. The outer product of the resulting point with e_∞ simplifies the expression by computing the flat point at the same location. Finally, the point is projected orthographically to $e_1 \wedge e_2 \wedge \cdots \wedge e_d \wedge e_o$:

$$\begin{aligned} V''' &= \text{PROJECT}\left(\left(-H\left(-T\,e_\infty\,T^{-1}\right)H\right) \wedge e_\infty\right) \\ &= \text{PROJECT}\left(\left(-H\left(\frac{2}{\alpha}e_{d+1} + \frac{1}{\alpha}e_o + \frac{2}{\alpha}e_\infty\right)H\right) \wedge e_\infty\right) \\ &= \text{PROJECT}\left(\left(-\frac{2x'_o}{\alpha^2}v - \frac{2(v \cdot v)}{\alpha^3}e_{d+1} - \frac{1}{\alpha}\left(\frac{(v \cdot v)\,x'_o}{\alpha^2} + x'^2_o\right)e_o - \frac{2(v \cdot v)}{\alpha^3}e_\infty\right) \wedge e_\infty\right) \\ &= \text{PROJECT}\left(-\frac{2x'_o}{\alpha^2}(v \wedge e_\infty) - \frac{2(v \cdot v)}{\alpha^3}(e_{d+1} \wedge e_\infty) - \left(\frac{(v \cdot v)\,x'_o}{\alpha^3} + \frac{x'^2_o}{\alpha}\right)(e_o \wedge e_\infty)\right) \\ &= -\frac{2x'_o}{\alpha^2}(v \wedge e_\infty) - \left(\frac{(v \cdot v)\,x'_o}{\alpha^3} + \frac{x'^2_o}{\alpha}\right)(e_o \wedge e_\infty). \end{aligned} \tag{7}$$

The resulting flat point Y is computed by multiplying V''' by $-\alpha^2/(2x'_o)$ to simplify the expression:

$$\begin{aligned} Y &= -\frac{\alpha^2}{2x'_o}V''' \\ &= -\frac{\alpha^2}{2x'_o}\left(-\frac{2x'_o}{\alpha^2}(v \wedge e_\infty) - \left(\frac{(v \cdot v)\,x'_o}{\alpha^3} + \frac{x'^2_o}{\alpha}\right)(e_o \wedge e_\infty)\right) \\ &= (v \wedge e_\infty) + \left(\frac{(v \cdot v)}{2\alpha} + \frac{\alpha x'_o}{2}\right)(e_o \wedge e_\infty) \\ &= \left(v + \left(\frac{(v \cdot v)}{2\alpha} + \frac{\alpha x'_o}{2}\right)e_o\right) \wedge e_\infty. \end{aligned} \tag{8}$$

Recall that the interpretation of V''' as a flat point at location y is unchanged if its coefficients are multiplied by a common factor different than zero.

Finally, the flat point Y, in (8), can be written using tensors to represent the point y resulting from applying ReSPro to x:

$$Y = \begin{pmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_d \\ y'_o \end{pmatrix} = \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_d \\ \frac{\alpha}{2} x'_o + \frac{1}{2\alpha} \sum_{i=1}^{d} (x'_i)^2 \end{pmatrix} = \left(\begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & \frac{\alpha}{2} \end{pmatrix} + \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ \frac{x'_1}{2\alpha} & \frac{x'_2}{2\alpha} & \cdots & \frac{x'_d}{2\alpha} & 0 \end{pmatrix} \right) \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_d \\ x'_o \end{pmatrix} \tag{9}$$

$$= (F_M + F_T X)\, X. \tag{10}$$

II. Linear CNN Layers and Other Operations as Matrices

Each paragraph that follows models a typical linear operation or configuration in CNNs in matrix form. Without loss of generality, we assume that the discrete signals transformed by these matrices are one-dimensional and have a single channel. The presentation of the multidimensional and multi-channel version of these expressions is out of scope of this Supplementary Material.

a) Data: The data is modeled as vectors in $\mathbb{R}^{d_x+1}$, where d_x is the size of the data and $^\intercal$ denotes matrix transposition:

$$X = \left(x'_1, x'_2, \cdots, x'_{d_x}, x'_o\right)^\intercal. \tag{11}$$

b) Weights: The weights are modeled as vectors in $\mathbb{R}^{d_w+1}$, where d_w is the size of the kernel:

$$W = (w_1, w_2, \cdots, w_{d_w}, 1)^\intercal. \tag{12}$$

c) Padding of Zeros: The matrix $P = (p_{ij})$ is a constant $(d_x + 2\delta_P + 1) \times (d_x + 1)$ matrix that models zero padding of δ_P units, where δ_P is the number of zeros inserted at each end of the signal represented as a vector in $\mathbb{R}^{d_x+1}$:

$$p_{ij} = \begin{cases} 1 & \text{, for } ((i - \delta_P) = j \text{ and } i < P_{\text{rows}} \text{ and } j < P_{\text{cols}}) \text{ or } (i = P_{\text{rows}} \text{ and } j = P_{\text{cols}}), \\ 0 & \text{, otherwise.} \end{cases} \tag{13}$$

d) Dilation: The matrix $D = (d_{ij})$ is a constant $(d_w + 1) \times ((d_w - 1)\delta_D + 2)$ matrix that models dilation with dilation rate δ_D:

$$d_{ij} = \begin{cases} 1 & \text{, for } ((i-1)\delta_D + 1 = j \text{ and } i < D_{\text{rows}} \text{ and } j < D_{\text{cols}}) \text{ or } (i = D_{\text{rows}} \text{ and } j = D_{\text{cols}}), \\ 0 & \text{, otherwise.} \end{cases} \tag{14}$$

e) Stride: The matrix $S = (s_{ij})$ is a constant $\left(\left\lfloor \frac{(d_x + 2\delta_P - \delta_D(d_w-1)-1)}{\delta_S} \right\rfloor + 2\right) \times (d_x - (d_w - 1)\delta_D + 2\delta_P + 1)$ matrix that model stride with displacement δ_S:

$$s_{ij} = \begin{cases} 1 & \text{, for } ((i-1)\delta_S = j - 1 \text{ and } i < S_{\text{rows}} \text{ and } j < S_{\text{cols}}) \text{ or } (i = S_{\text{rows}} \text{ and } j = S_{\text{cols}}), \\ 0 & \text{, otherwise.} \end{cases} \tag{15}$$

f) Convolution: The matrix M that models convolution is computed as a composition of weights, dilation, valid cross-correlation, stride, and padding:

$$M = S\ (W\, D\, C)\ P = W\ (S\ (D\, C)^\intercal\ P)^\intercal, \tag{16}$$

where the constant rank-3 tensor $C = (c_{ijk})$, of size $((d_w - 1)\delta_D + 2) \times (d_x - (d_w - 1)\delta_D + 2\delta_P + 1) \times (d_x + 2\delta_P + 1)$, models the valid cross-correlation:

$$d_{ijk} = \begin{cases} 1 & \text{, for } (i = k - j + 1 \text{ and } i < C_{\text{dim1}} \text{ and } j < C_{\text{dim2}} \text{ and } k < C_{\text{dim3}}) \text{ or } (i = C_{\text{dim1}} \text{ and } j = C_{\text{dim2}} \text{ and } k = C_{\text{dim3}}), \\ 0 & \text{, otherwise.} \end{cases} \tag{17}$$

g) Average Pooling: The constant matrix A modeling average pooling is computed as a composition of constant weights, valid cross-correlation, stride, and padding:

$$A = S\ (W\, C)\ P = W\ (S\, C^\intercal\, P)^\intercal, \tag{18}$$

where $W = \left(\frac{1}{d_w}, \frac{1}{d_w}, \cdots, \frac{1}{d_w}, 1\right)^\intercal \in \mathrm{R}^{d_w+1}$ is a constant vector of weights and d_w is the size of the kernel.

h) Dropout: The $(d_x + 1) \times (d_x + 1)$ diagonal matrix $R = (r_{i,j})$ encodes the dropout operation. Its entries are:

$$r_{ij} = \begin{cases} 1 & \text{, for } i = j \text{ and } \textsc{rand}() > \delta_R, \\ 0 & \text{, otherwise.} \end{cases} \tag{19}$$

Here, $\textsc{rand}()$ is a function that generates random values uniformly distributed on the interval [0, 1], and δ_R is the probability of an element to be zeroed.

III. ConformalLayers: From Successive Evaluation of $\mathcal{L}^{(l)}$ Functions to Tensors

For the sake o clarity, we first show the algebraic manipulation that turns:

$$Y^{(k)} = \mathcal{L}^{(k)}(\mathcal{L}^{(k-1)}(\mathcal{L}^{(k-2)}(\cdots))) \tag{20}$$

into

$$Y^{(k)} = \left(L_M^{(k)} + L_T^{(k)} X\right) X \tag{21}$$

for $k = 3$, and then define the case for any k by induction.

Let $k = 3$. Equation (20) expands to:

$$\begin{aligned}
Y^{(3)} &= \mathcal{L}^{(3)}\left(\mathcal{L}^{(2)}\left(\mathcal{L}^{(1)}(X)\right)\right) \\
&= F_M^{(3)} U^{(3)} \left(F_M^{(2)} U^{(2)} F_M^{(1)} U^{(1)} X + \left(F_M^{(2)} U^{(2)} \left(U^{(1)\intercal} F_T^{(1)\intercal} U^{(1)}\right)^\intercal X\right) X + \left(\left(U^{(1)\intercal} U^{(2)\intercal} F_T^{(2)\intercal} U^{(2)} U^{(1)}\right)^\intercal X\right) X\right) \\
&\qquad + \left(\left(U^{(3)\intercal} F_T^{(3)\intercal} U^{(3)}\right)^\intercal \left(F_M^{(2)} U^{(2)} F_M^{(1)} U^{(1)} X + \left(F_M^{(2)} U^{(2)} \left(U^{(1)\intercal} F_T^{(1)\intercal} U^{(1)}\right)^\intercal X\right) X\right.\right. \\
&\qquad\qquad \left.\left. + \left(\left(U^{(1)\intercal} U^{(2)\intercal} F_T^{(2)\intercal} U^{(2)} U^{(1)}\right)^\intercal X\right) X\right)\right) \\
&\qquad\qquad \left(F_M^{(2)} U^{(2)} F_M^{(1)} U^{(1)} X + \left(F_M^{(2)} U^{(2)} \left(U^{(1)\intercal} F_T^{(1)\intercal} U^{(1)}\right)^\intercal X\right) X\right. \\
&\qquad\qquad\qquad \left. + \left(\left(U^{(1)\intercal} U^{(2)\intercal} F_T^{(2)\intercal} U^{(2)} U^{(1)}\right)^\intercal X\right) X\right) \\
&= F_M^{(3)} U^{(3)} F_M^{(2)} U^{(2)} F_M^{(1)} U^{(1)} X \\
&\qquad + \left(F_M^{(3)} U^{(3)} F_M^{(2)} U^{(2)} \left(U^{(1)\intercal} F_T^{(1)\intercal} U^{(1)}\right)^\intercal X\right) X \\
&\qquad + \left(F_M^{(3)} U^{(3)} \left(U^{(1)\intercal} U^{(2)\intercal} F_T^{(2)\intercal} U^{(2)} U^{(1)}\right)^\intercal X\right) X \\
&\qquad + \left(\left(U^{(3)\intercal} F_T^{(3)\intercal} U^{(3)}\right)^\intercal F_M^{(2)} U^{(2)} F_M^{(1)} U^{(1)} X\right) F_M^{(2)} U^{(2)} F_M^{(1)} U^{(1)} X \\
&= F_M^{(3)} U^{(3)} F_M^{(2)} U^{(2)} F_M^{(1)} U^{(1)} X \\
&\qquad + \left(F_M^{(3)} U^{(3)} F_M^{(2)} U^{(2)} \left(U^{(1)\intercal} F_T^{(1)\intercal} U^{(1)}\right)^\intercal X\right) X \\
&\qquad + \left(F_M^{(3)} U^{(3)} \left(U^{(1)\intercal} U^{(2)\intercal} F_T^{(2)\intercal} U^{(2)} U^{(1)}\right)^\intercal X\right) X \\
&\qquad + \left(\left(U^{(1)\intercal} U^{(2)\intercal} U^{(3)\intercal} F_T^{(3)\intercal} U^{(3)} U^{(2)} U^{(1)}\right)^\intercal X\right) X \\
&= \left(F_M^{(3)} U^{(3)} F_M^{(2)} U^{(2)} F_M^{(1)} U^{(1)} + \right. \\
&\qquad \left.\left(\sum_{l=1}^{3} \left(F_M^{(3)} U^{(3)} F_M^{(2)} U^{(2)} \cdots F_M^{(l+1)} U^{(l+1)}\right) \left(U^{(1)\intercal} U^{(2)\intercal} \cdots U^{(l)\intercal} F_T^{(l)\intercal} U^{(l)} \cdots U^{(2)} U^{(1)}\right)^\intercal\right) X\right) X \\
&= \left(L_M^{(3)} + L_T^{(3)} X\right) X,
\end{aligned} \tag{22}$$

where, by induction:

$$L_M^{(k)} = F_M^{(k)} U^{(k)} F_M^{(k-1)} U^{(k-1)} \cdots F_M^{(1)} U^{(1)} \tag{23}$$

and

$$L_T^{(k)} = \sum_{l=1}^{k} \left(F_M^{(k)} U^{(k)} F_M^{(k-1)} U^{(k-1)} \cdots F_M^{(l+1)} U^{(l+1)}\right) \left(U^{(1)\intercal} U^{(2)\intercal} \cdots U^{(l)\intercal} F_T^{(l)\intercal} U^{(l)} \cdots U^{(2)} U^{(1)}\right)^\intercal. \tag{24}$$

Here, $^\intercal$ denotes the transposition of the first two dimensions of tensors and matrix transposition.

IV. Selected Hyperparameter Values

The hyperparameter values selected for each CNN and dataset are presented in Table I. These values provided higher accuracy during the validation step. The search space is presented in Table II.

TABLE I: The hyperparameter values selected for each CNN and dataset using a Bayesian implemented by the Weights and Biases toolset (`https://www.wandb.com/`). For the `LeNet` and `LeNetCL` networks, the number of epochs was set to 200 in training†, as the range of values shown in Table II was proved insufficient to adjust the models.

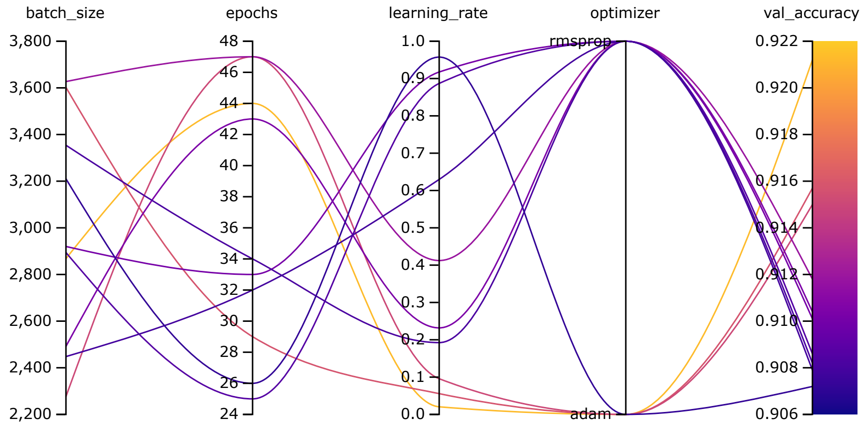
`BaseLinearNet`	MNIST	Fashion-MNIST	CIFAR-10
Batch size	2868	3111	2198
Epochs	44	15	41
Learning rate	0.02039	0.05305	0.09827
Optimizer	Adam	Adam	Adam
`BaseReLUNet`	MNIST	Fashion-MNIST	CIFAR-10
Batch size	2838	3277	3677
Epochs	50	48	21
Learning rate	0.090331	0.08607	0.01
Optimizer	Adam	Adam	RMSprop
`BaseReSProNet`	MNIST	Fashion-MNIST	CIFAR-10
Batch size	2429	2339	3056
Epochs	50	45	19
Learning rate	0.7699	0.5601	0.4542
Optimizer	Adam	Adam	Adam
`LeNet`	MNIST	Fashion-MNIST	CIFAR-10
Batch size	2979	2444	2317
Epochs†	200	200	200
Learning rate	0.004722	0.012	0.00159
Optimizer	RMSprop	Adam	RMSprop
`LeNetCL`	MNIST	Fashion-MNIST	CIFAR-10
Batch size	2088	2763	3595
Epochs†	200	200	200
Learning rate	0.02	0.02302	0.01991
Optimizer	Adam	Adam	Adam

TABLE II: Search space for hyperparameters used in the experiments.

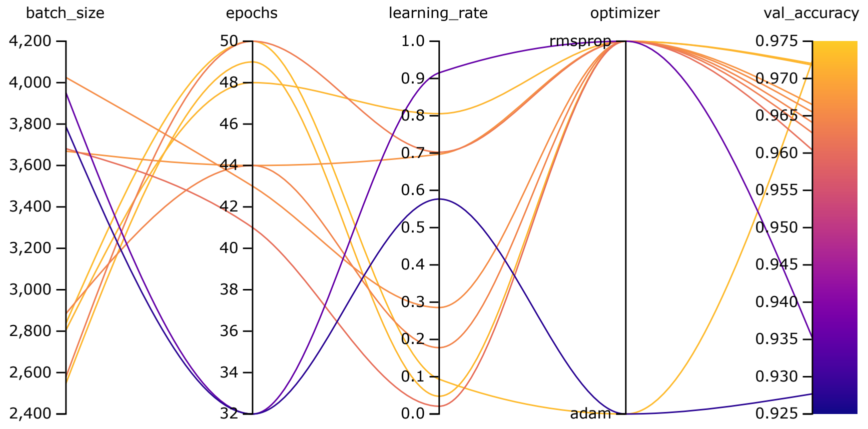
	Lower Bound	Upper Bound	Type
Batch size	2048	4096	Discrete uniform distribution
Epochs	10	50	Discrete uniform distribution
Learning rate	0.001	1.0	Continuous uniform distribution
Optimizer	{Adam, RMSprop}		Categorical distribution

V. Baseline Hyperparameter Sweep

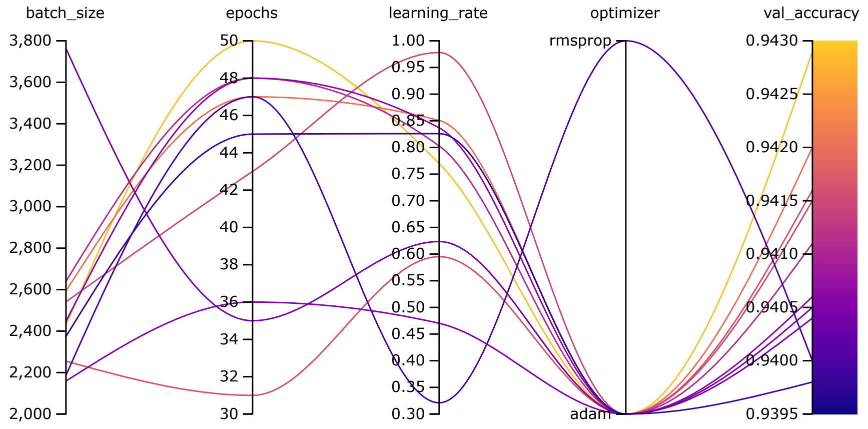
Figs. 1, 2, and 3 present smooth parallel coordinate plots indicating the hyperparameter values selected for the top-10 validation accuracy achieved by `BaseLinearNet`, `BaseReLUNet`, and `BaseReSProNet` during training using, respectively, the MNIST [2], Fashion-MNIST [3], and CIFAR-10 [4] datasets.



(a) `BaseLinearNet`

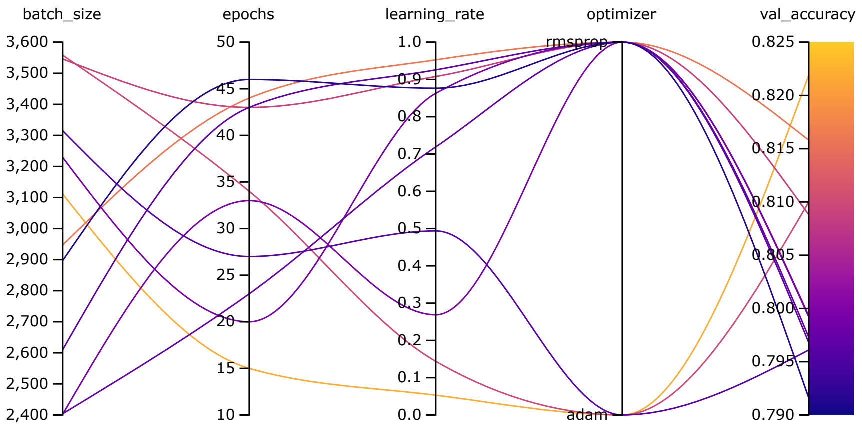


(b) `BaseReLUNet`

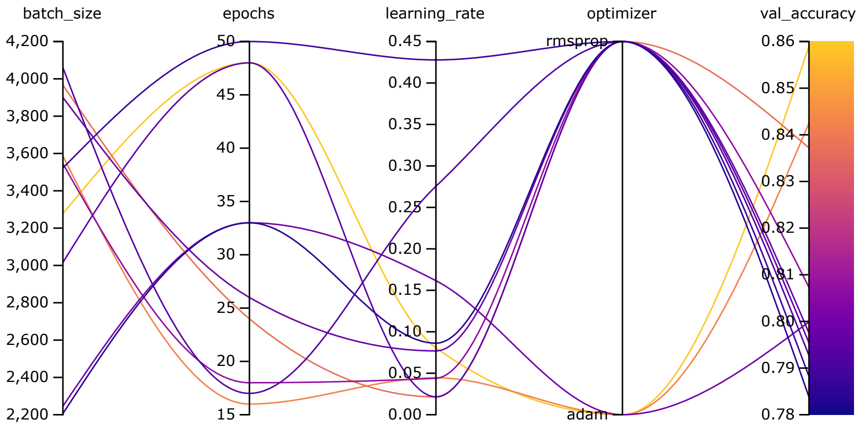


(c) `BaseReSProNet`

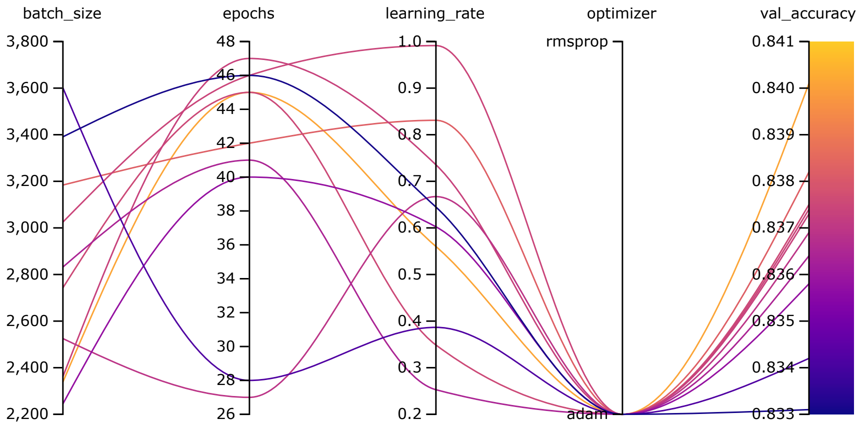
Fig. 1: Selected hyperparameters for the top-10 validation accuracy of CNNs used in Experiment I on the MNIST dataset.



(a) `BaseLinearNet`

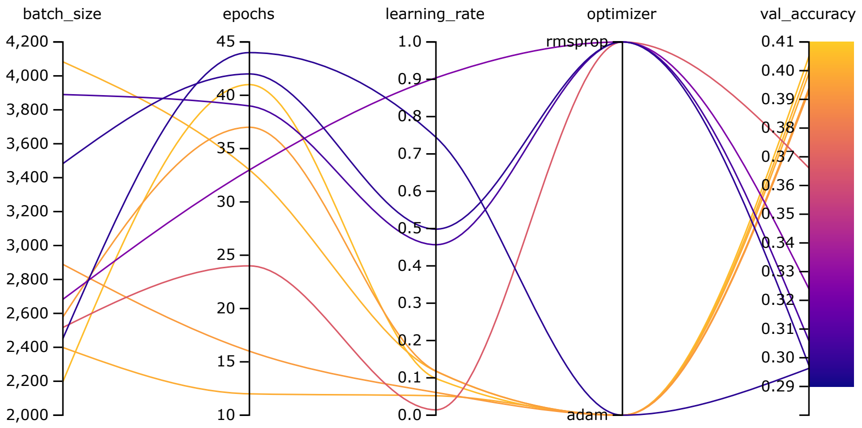


(b) `BaseReLUNet`

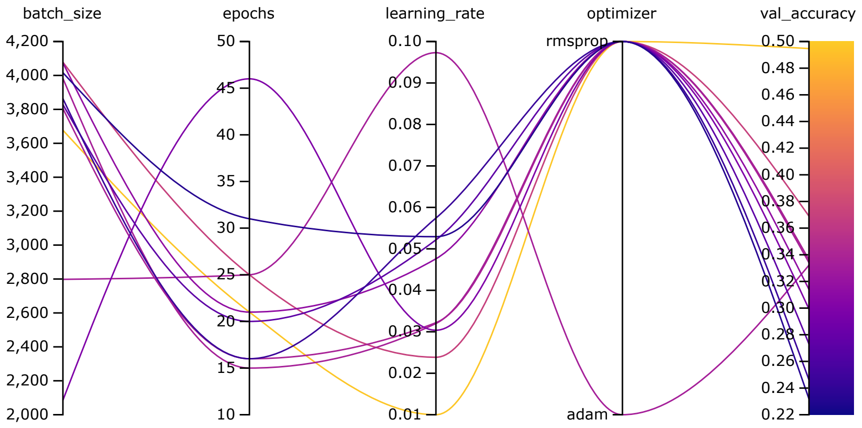


(c) `BaseReSProNet`

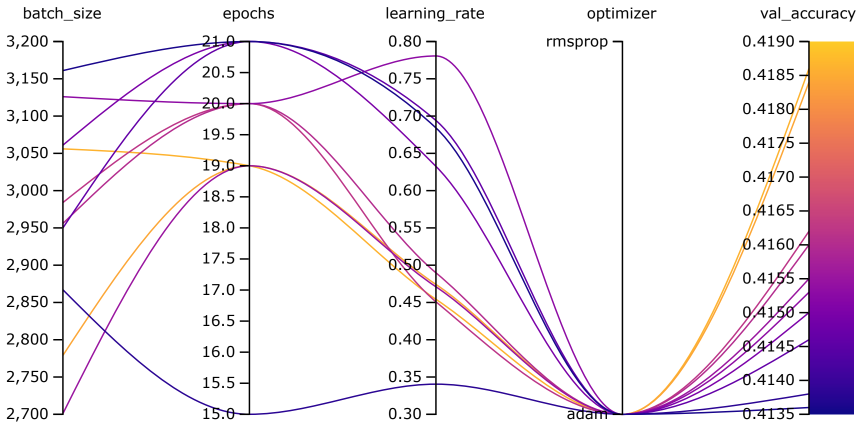
Fig. 2: Selected hyperparameters for the top-10 validation accuracy of CNNs used in Experiment I on the Fashion-MNIST.



(a) `BaseLinearNet`



(b) `BaseReLUNet`

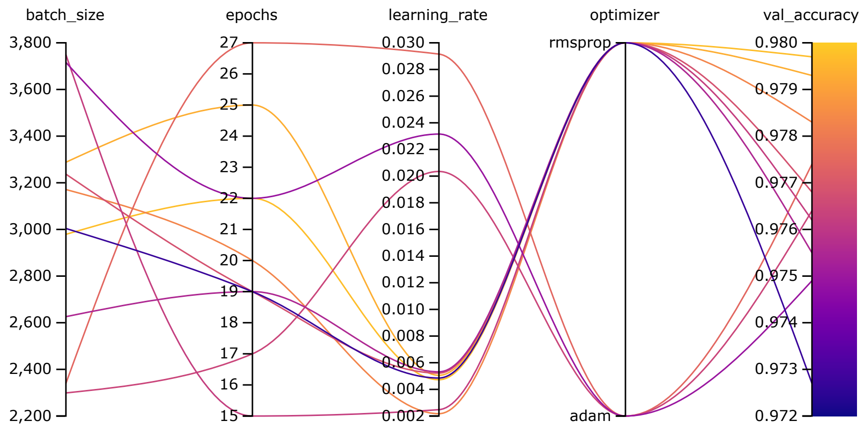


(c) `BaseReSProNet`

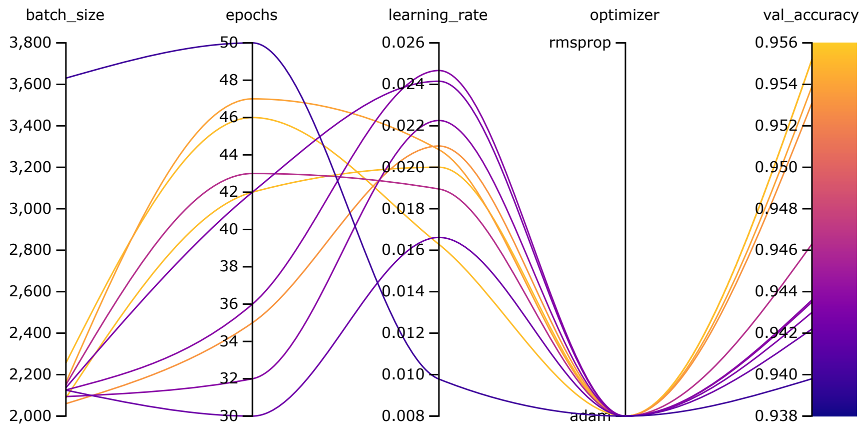
Fig. 3: Selected hyperparameters for the top-10 validation accuracy of CNNs used in Experiment I on the CIFAR-10 dataset.

VI. LeNet Hyperparameter Sweep

Figs. 4, 5, and 6 present smooth parallel coordinate plots indicating the hyperparameter values selected for the top-10 validation accuracy achieved by `LeNet` and `LeNetCL` during training using, respectively, the MNIST [2], Fashion-MNIST [3], and CIFAR-10 [4] datasets. After this initial parameter selection, we have trained each CNN/dataset pair using 200 epochs.

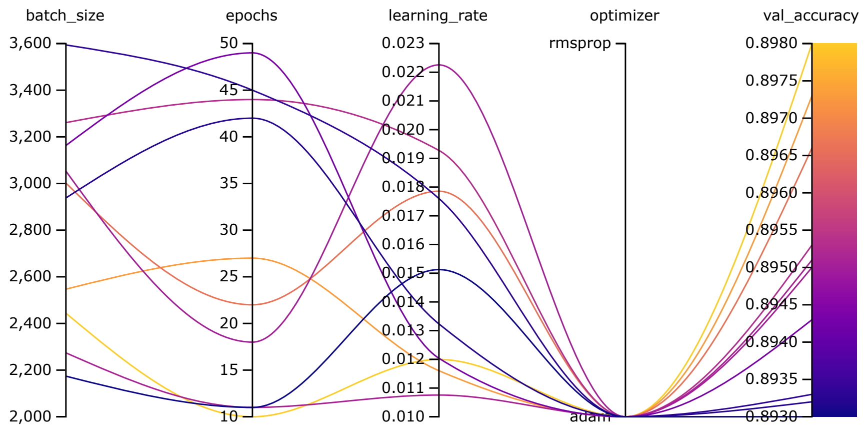


(a) `LeNet`

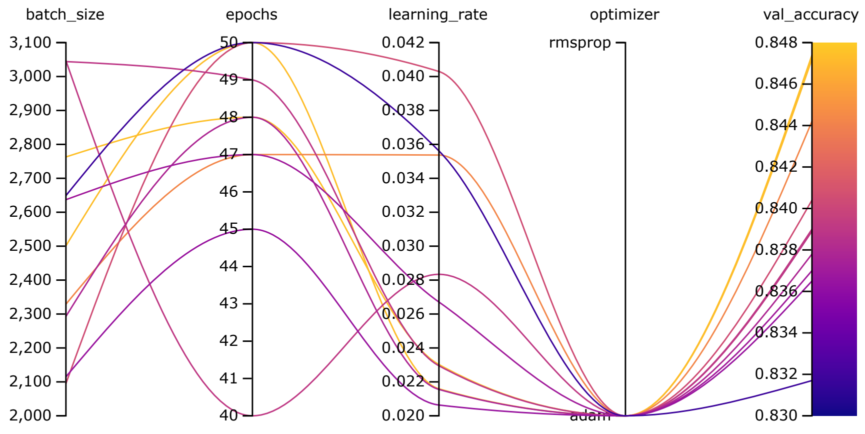


(b) `LeNetCL`

Fig. 4: Selected hyperparameters for the top-10 validation accuracy of CNNs based on the LeNet-5 [5], used in Experiment II on the MNIST dataset.

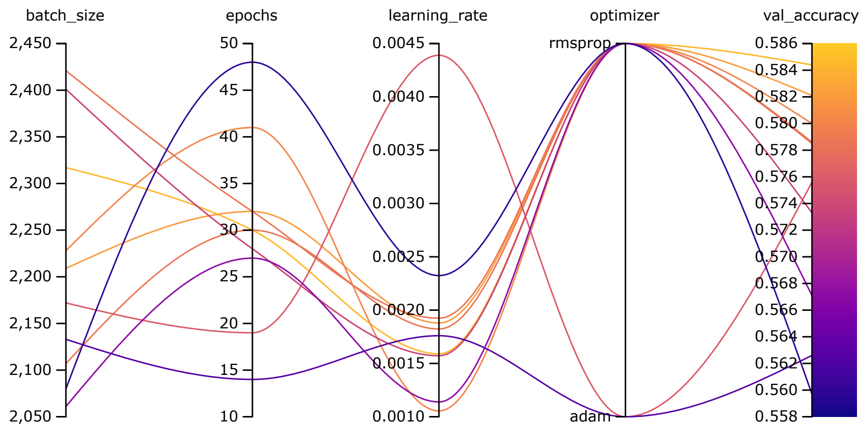


(a) `LeNet`

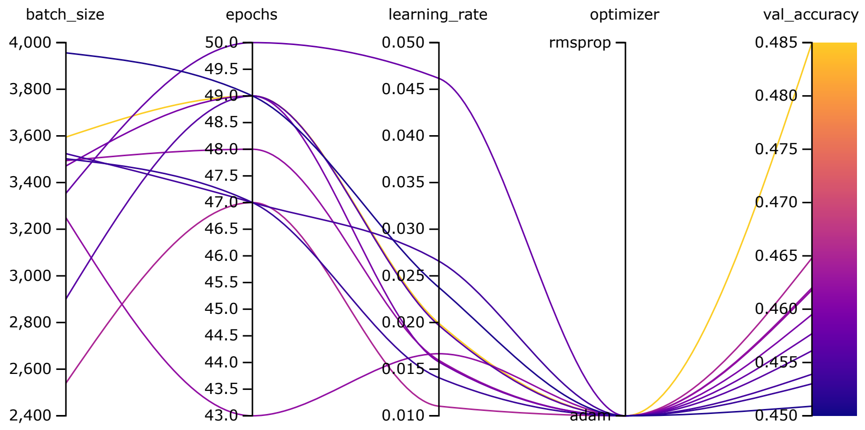


(b) `LeNetCL`

Fig. 5: Selected hyperparameters for the top-10 validation accuracy of CNNs based on the LeNet-5 [5], used in Experiment II on the Fashion-MNIST dataset.



(a) LeNet



(b) LeNetCL

Fig. 6: Selected hyperparameters for the top-10 validation accuracy of CNNs based on the LeNet-5 [5], used in Experiment II on the CIFAR-10 dataset.

References

[1] L. Dorst, D. Fontijne, and S. Mann, *Geometric algebra for computer science: an object-oriented approach to geometry*. Elsevier, 2010.
[2] Y. LeCun, C. Cortes, and C. J. Burges, "MNIST handwritten digit database," ATT Labs, 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist
[3] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms," arXiv: 1708.07747, 2017.
[4] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.
[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.