Appendix

A. *Tensors to Matrices*

Order	# full tensor terms	# unique terms	# reductions in state	Reduction ratio	Maximum reduction ratio
1	m	m	0	0	0
2	m^2	$\frac{m(m+1)}{2}$	$\frac{m^2-m}{2}$	$\frac{1}{2}-\frac{1}{2m}$	$\frac{1}{2}$
3	m^3	$\frac{m(m+1)(m+2)}{3!}$	$\frac{5m^3-3m^2-2m}{6}$	$\frac{5}{6}-\frac{1}{2m}-\frac{1}{3m^2}$	$\frac{5}{6}$
4	m^4	$\frac{m(m+1)(m+2)(m+3)}{4!}$	$\frac{23m^4-6m^3-11m^2-6m}{24}$	$\frac{23}{24}-\frac{1}{4m}-\frac{11}{24m^2}-\frac{1}{4m^3}$	$\frac{23}{24}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
d	m^d	$\prod_{i=1}^{d} \frac{m+i-1}{i}$			$\frac{d!-1}{d!}$

TABLE VII
PROGRESSION OF DERIVATIVE ORDER, NUMBER OF TENSOR TERMS, UNIQUE COMBINATIONS, AND REDUCTION RATIO

Order	Derivative tensor	Dim. of tensor	State	Dim. of state	Derivative matrix	Dim. of matrix	Multinomial state and coefficient vector	Dim. of multinomial state and coefficient vector
0	$\boldsymbol{f}(\boldsymbol{x_0})$	$\mathbb{R}^{n\times 1}$						
1	$J_f^1(\boldsymbol{x_0})$	$\mathbb{R}^{n\times m}$	$\boldsymbol{x}$	$\mathbb{R}^{m\times 1}$				
2	$J_f^2(\boldsymbol{x_0})$	$\mathbb{R}^{n\times m\times m}$	$\boldsymbol{x}^{\otimes 2}$	$\mathbb{R}^{m\times m\times 1}$	$\tilde{J}_f^2(\boldsymbol{x_0})$	$\mathbb{R}^{n\times\frac{m(m+1)}{2}}$	$\tilde{\boldsymbol{x}}^{\otimes 2}, \boldsymbol{a}^2$	$\mathbb{R}^{\frac{m(m+1)}{2}\times 1}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
d	$J_f^d(\boldsymbol{x_0})$	$\mathbb{R}^{n\times m\times\ldots\times m}$	$\boldsymbol{x}^{\otimes d}$	$\mathbb{R}^{m\times m\times 1}$	$\tilde{J}_f^2(\boldsymbol{x_0})$	$\mathbb{R}^{n\times n_d}$	$\tilde{\boldsymbol{x}}^{\otimes d}, \boldsymbol{a}^d$	$\mathbb{R}^{n_d\times 1}$

TABLE VIII
ASCENDING ORDER OF DERIVATIVE WITH ASSOCIATED TENSOR REPRESENTATION, DIMENSION OF TENSOR, STATE REPRESENTATION, AND DIMENSION OF STATE

Multinomial state vector term	Equivalent state tensor term	State composition	Index mapping
$\tilde{\boldsymbol{x}}^{\otimes 2}(j)$	$\boldsymbol{x}^{\otimes 2}(i_1, i_2)$	$x_{i_1}x_{i_2}$	$j = i_1 + \frac{i_2(i_2-1)}{2}$ for $i_2 = 1:m$, for $i_1 = 1:i_2$
$\tilde{\boldsymbol{x}}^{\otimes 3}(j)$	$\boldsymbol{x}^{\otimes 3}(i_1, i_2, i_3)$	$x_{i_1}x_{i_2}x_{i_3}$	$j = i_1 + \frac{i_2(i_2-1)}{2} + \frac{(i_3+1)i_3(i_3-1)}{3!}$ for $i_3 = 1:m$, for $i_2 = 1:i_3$, for $i_1 = 1:i_2$
$\tilde{\boldsymbol{x}}^{\otimes 4}(j)$	$\boldsymbol{x}^{\otimes 4}(i_1, i_2, i_3, i_4)$	$x_{i_1}x_{i_2}x_{i_3}x_{i_4}$	$j = i_1 + \frac{i_2(i_2-1)}{2} + \frac{(i_3+1)i_3(i_3-1)}{3!} + \frac{(i_4+2)(i_4+1)i_4(i_4-1)}{4!}$ for $i_4 = 1:m$, for $i_3 = 1:i_4$,for $i_2 = 1:i_3$, for $i_1 = 1:i_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\tilde{\boldsymbol{x}}^{\otimes d}(j)$	$\boldsymbol{x}^{\otimes d}(i_1, i_2, \cdots, i_d)$	$x_{i_1}x_{i_2}\cdots x_{i_d}$	$j = i_1 + \frac{i_2(i_2-1)}{2} + \cdots + \prod_{k=1}^{d} \frac{i_d+k-2}{k}$ for $i_d = 1:m$, for $i_{d-1} = 1:i_d$, $\cdots$, for $i_1 = 1:i_2$

TABLE IX
TENSOR TO VECTOR UNFOLDING OF MULTINOMIAL STATE FOR ASCENDING ORDER TERMS

B. *Coefficient Tensor Derivations*

1) Perceptron Layer with Binary Activation: The simplest activation function is binary, defined by a piece-wise constant function, given in Table I. The binary activation function is typically the last layer in a network, answering questions like yes/no in a classification application. For a fully-connected perceptron layer with binary activation function, the derivatives of this function truncate immediately. The only term in the Taylor approximation is the 0^{th} term, a_i^0 in (78). Note that the Taylor series is equivalent to the original function expressions, which shall be the case for any function that is originally in polynomial form. This form is the most general binary expression, although a common simpler expression relates the only the i^{th} input to the i^{th} output, given in (79). In this special case, the weight matrix is a square identity matrix and the biases are all zeros.

Multinomial coefficient vector term	Corresponding state composition	Multinomial coefficient value
$\boldsymbol{a}^2(j)$	$x_{i_1}x_{i_2}$	$= \frac{1}{2!}\binom{2}{1,1} = 2$ if $i_1 \neq i_2$ $= \frac{1}{2!}\binom{2}{0,2} = 2$ if $i_1 = i_2$
$\vdots$	$\vdots$	$\vdots$
$\boldsymbol{a}^d(j)$	$x_{i_1}x_{i_2}\cdots x_{i_d}$	$= \frac{1}{d!}\binom{d}{n_1, n_2,, \cdots, n_m}$ for $n_1 = \mathcal{O}(x_1) \in x_{i_1}x_{i_2}\cdots x_{i_d}, \cdots, n_m = \mathcal{O}(x_m) \in x_{i_1}x_{i_2}\cdots x_{i_d}$

TABLE X
MULTINOMIAL COEFFICIENT VECTOR VALUES AND INDICES FOR ASCENDING ORDER TERMS

Derivative matrix term	Equivalent derivative tensor term	Corresponding state composition	Index mapping
$\tilde{J}^2_f(:, j)$	$J^2_f(:, i_1, i_2)$	$x_{i_1}x_{i_2}$	$j = i_1 + \frac{i_2(i_2-1)}{2}$ for $i_2 = 1 : m$, for $i_1 = 1 : i_2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\tilde{J}^d_f(:, j)$	$J^2_f(:, i_1, i_2, \cdots, i_d)$	$x_{i_1}x_{i_2}\cdots x_{i_d}$	$j = i_1 + \frac{i_2(i_2-1)}{2} + \cdots + \prod_{k=1}^{d} \frac{i_d+k-2}{k}$ for $i_d = 1 : m$, for $i_{d-1} = 1 : i_d$, $\cdots$, for $i_1 = 1 : i_2$

TABLE XI
TENSOR TO MATRIX UNFOLDING OF DERIVATIVE TERM FOR ASCENDING ORDER TERMS

$$\begin{aligned} y_j &= f_j(x_i = 0) \\ &= a^0_j = \begin{cases} 1, & \text{if } \sum_{i=1}^{m} w_{ji}x_i + b_j \geq 0 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{78}$$

$$\begin{aligned} y_i &= f(x_i = 0) \\ &= a^0_i = \begin{cases} 1, & \text{if } x_i \geq 0 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{79}$$

2) Perceptron Layer with Linear Activation: The linear activation function is a continuously differentiable function, typically used in regression applications, given in Table I. The Taylor approximation series truncates after the 1^{st} order term, given in (80). The 0^{th} and 1^{st} order terms are given in (81) and (82). Much like the binary expression, the Taylor approximation yields an exact representation to the original function expression.

$$y_j = a^0_j + \sum_{i=1}^{m} A^1_{ji}x_i \tag{80}$$

$$a^0_j = f_j(x_i = 0) = b_j \tag{81}$$

$$A^1_{ji} = \frac{\partial f_j}{\partial x_i}\bigg|_{x_i=0} = w_{ji} \tag{82}$$

3) Perceptron Layer with ReLU Activation: The rectified linear unit activation function is a piece-wise linear function combining the binary and linear activation functions, given in Table I. The ReLU activation function is one of the most popular activation functions, used in a variety of different network architectures. The Taylor approximation series truncates after the 1^{st} order term, identical to the linear case, given in (80). The 0^{th} and 1^{st} order terms are given in (83) and (84). Much like the binary and linear expressions, the Taylor approximation yields an exact representation to the original function expression.

$$a^0_j = \begin{cases} b_j, & \text{if } \sum_{i=1}^{m} w_{ji}x_i + b_j \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{83}$$

$$A^1_{ji} = \begin{cases} w_{ji}, & \text{if } \sum_{i=1}^{m} w_{ji}x_i + b_j \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{84}$$

4) Perceptron Layer with Sigmoid Activation: The sigmoid activation function, a special case of a logistic function, is a continuously differentiable nonlinear activation function, given in Table I. The sigmoid activation function is one of the most popular activation functions, used in a variety of different network architectures. An arbitrarily exact Taylor series expansion never truncates as the derivatives are infinitely differentiable. Realistically, a Taylor approximation series is artificially truncated at a user-defined order d. For ease of defining the polynomial coefficient terms, the sigmoid function is not explicitly expressed in length, but compactly represented by the symbol $\sigma(\cdot)$. The 0^{th} to 2^{nd} order terms are given in (85) to (87) to give a sense of the iterative pattern. The higher order polynomial coefficient tensors are iteratively derived with immediately previous polynomial coefficient tensor, given in (23). These equations fully define the polynomial coefficient tensors for any arbitrary order term to populate the entire Taylor approximation series.

$$a_j^0 = f_j(x_i = 0) = \sigma(b_j) \tag{85}$$

$$\begin{aligned} A_{ji}^1 &= \frac{\partial f_j}{\partial x_i}\Big|_{x_i=0} \\ &= w_{ji}\sigma(b_j)(1-\sigma(b_j)) \end{aligned} \tag{86}$$

$$\begin{aligned} A_{ji_1i_2}^2 &= \frac{\partial}{\partial x_{i_2}}[\frac{\partial f_j}{\partial x_{i_1}}]\Big|_{x_{i_1},x_{i_2}=0} \\ &= w_{ji_1}w_{ji_2}(1-2\sigma(b_j))\sigma(b_j)(1-\sigma(b_j)) \end{aligned} \tag{87}$$

5) Perceptron Layer with Tanh Activation: The hyberbolic tangent function is a scaled and shifted sigmoid function, such that the output ranges from $[-1, 1]$. Like the sigmoid function, the tanh function is nonlinear, continuously differentiable, and infinitely differentiable. The exact and approximate Taylor series are equivalently represented by the sigmoid defined Taylor series in (22). For this subsection, the tanh function is compactly represented by the symbol $\sigma(\cdot)$. The 0^{th} to 2^{nd} and d^{th} order terms are given in (88) to (24) to give a sense of the iterative pattern. These equations fully define the polynomial coefficient tensors for any arbitrary order term to populate the entire Taylor approximation series.

$$a_j^0 = f_j(x_i = 0) = \sigma(b_j) \tag{88}$$

$$\begin{aligned} A_{ji}^1 &= \frac{\partial f_j}{\partial x_i}\Big|_{x_i=0} \\ &= w_{ji}(1-\sigma^2(b_j)) \end{aligned} \tag{89}$$

$$\begin{aligned} A_{ji_1i_2}^2 &= \frac{\partial}{\partial x_{i_2}}[\frac{\partial f_j}{\partial x_{i_1}}]\Big|_{x_{i_1},x_{i_2}=0} \\ &= w_{ji_1}w_{ji_2}(-2\sigma(b_j))(1-\sigma^2(b_j)) \end{aligned} \tag{90}$$

6) Perceptron Layer with Softmax Activation: The softmax activation function is an expression in the exponential family, given in Table I. Like the binary function, the softmax activation function is typically the last layer in a network, revealing confidence of classification. Unlike the binary activation function, the softmax function is infinitely differentiable, which leads to an exact and approximate Taylor series given in (22). The 0^{th}, 1^{st}, and d^{th} order terms are given in (91) to (25) to give a sense of the iterative pattern. This form is the most general softmax expression, although a common simpler expression relates the only the i^th input to the i^th output, given in (93). In this special case, the weight matrix is a square identity matrix and the biases are all zeros. The Taylor series states and polynomial coefficients can then be simplified to elementwise exponentiation, instead of vectorwise exponentiation, given in (94). The polynomial coefficients for this special case are given in (95) and note that the coefficients are independent of the state variation.

$$a_j^0 = f_j(x_i = 0) = \frac{1}{\sum_{k=1}^m e^{\sum_{i=1}^m w_{ki}x_i + b_k}} e^{b_j} \tag{91}$$

$$A_{ji}^1 = \frac{\partial f_j}{\partial x_i}\Big|_{x_i=0} = \frac{1}{\sum_{k=1}^m e^{\sum_{i=1}^m w_{ki}x_i + b_k}} e^{b_j} w_{ji} \tag{92}$$

$$f(x_i = 0) = \frac{1}{\sum_{k=1}^m e^{x_k}} e^{x_i} \tag{93}$$

$$y_i = a_i^0 + A_i^1 x_i + \frac{1}{2!}A_i^2 x_i^2 + \cdots \tag{94}$$

$$a_j^0 = A_i^1 = A_i^d = \frac{1}{\sum_{k=1}^m e^{x_k}} \tag{95}$$

7) Perceptron Layer with Probabilistic Activation: Probabilistic activation functions come in many different types of which the Gaussian or radial basis function is the most popular function, given in Table I. This function is nonlinear and infinitely differentiable with a singularity at $\boldsymbol{x} = \boldsymbol{c}_i$. Much like the softmax, the Gaussian activation function is part of the exponential family of expressions, which leads to an exact and approximate Taylor series given in (22). The 0^{th} to 4^{th} order terms are explicitly given in (96) to (100) to give a sense of the iterative pattern. A table of α_t^D and s_t^D up to the third term and fourth degree are listed in Table XII and Table XIII. The most general terms, α_t^D and s_t^D, for any term and degree definition are in (28) and (29).

$$\begin{aligned} a_j^0 &= f_j(x_i = 0) \\ &= e^{-\beta_j ||\boldsymbol{c}_j||} \end{aligned} \tag{96}$$

$$\begin{aligned} A_{ji}^1 &= \frac{\partial f_j}{\partial x_i}\Big|_{x_i=0} \\ &= e^{-\beta_j ||\boldsymbol{c}_j||} \frac{\beta_j c_{ji}}{||\boldsymbol{c}_j||} \\ &= e^{-\beta_j ||\boldsymbol{c}_j||} \alpha_1^1 s_1^1 \end{aligned} \tag{97}$$

$$\begin{aligned} A_{ji_1i_2}^2 &= \frac{\partial}{\partial x_{i_2}}\Big[\frac{\partial f_j}{\partial x_{i_1}}\Big]\Big|_{x_{i_1},x_{i_2}=0} \\ &= e^{-\beta_j ||\boldsymbol{c}_j||}\Big[\prod_{k=1}^{2} \frac{\beta_j c_{ji_k}}{||\boldsymbol{c}_j||}\Big(1 + \frac{1}{\beta_j ||\boldsymbol{c}_j||}\Big) + \frac{-\beta_j}{||\boldsymbol{c}_j||}\Big] \\ &= e^{-\beta_j ||\boldsymbol{c}_j||}(\alpha_1^2 s_1^2 + \alpha_2^2 s_2^2) \end{aligned} \tag{98}$$

$$\begin{aligned} A_{ji_1i_2i_3}^3 &= \frac{\partial}{\partial x_{i_3}}\Big[\frac{\partial}{\partial x_{i_2}}\Big[\frac{\partial f_j}{\partial x_{i_1}}\Big]\Big]\Big|_{x_{i_1},x_{i_2},x_{i_3}=0} \\ &= e^{-\beta_j ||\boldsymbol{c}_j||}\Big[\prod_{k=1}^{3} \frac{\beta_j c_{ji_k}}{||\boldsymbol{c}_j||}\Big[\Big(1 + \frac{2}{\beta_j ||\boldsymbol{c}_j||}\Big)\Big(1 + \frac{1}{\beta_j ||\boldsymbol{c}_j||}\Big) + \Big(\frac{-1}{\beta_j ||\boldsymbol{c}_j||}\Big)^2\Big] + \sum_{k=1}^{3} \frac{\beta_j c_{ji_k}}{||\boldsymbol{c}_j||}\Big(\frac{-\beta_j}{||\boldsymbol{c}_j||}\Big)\Big(1 + \frac{1}{\beta_j ||\boldsymbol{c}_j||}\Big) \\ &= e^{-\beta_j ||\boldsymbol{c}_j||}(\alpha_1^3 s_1^3 + \alpha_2^3 s_2^3) \end{aligned} \tag{99}$$

$$\begin{aligned} A_{ji_1i_2i_3i_4}^4 &= \frac{\partial}{\partial x_{i_4}}\Big[\frac{\partial}{\partial x_{i_3}}\Big[\frac{\partial}{\partial x_{i_2}}\Big[\frac{\partial f_j}{\partial x_{i_1}}\Big]\Big]\Big]\Big|_{x_{i_1},x_{i_2},x_{i_3},x_{i_4}=0} \\ &= e^{-\beta_j ||\boldsymbol{c}_j||}\Big[\prod_{k=1}^{4} \frac{\beta_j c_{ji_k}}{||\boldsymbol{c}_j||}\Big[\Big(1 + \frac{3}{\beta_j ||\boldsymbol{c}_j||}\Big)\Big[\Big(1 + \frac{2}{\beta_j ||\boldsymbol{c}_j||}\Big)\Big(1 + \frac{1}{\beta_j ||\boldsymbol{c}_j||}\Big) + \Big(\frac{-1}{\beta_j ||\boldsymbol{c}_j||}\Big)^2\Big] + \Big(\frac{-3}{\beta_j ||\boldsymbol{c}_j||}\Big)^2\Big(1 + \frac{2}{\beta_j ||\boldsymbol{c}_j||}\Big) \\ &\quad + \sum_{k=1}^{4}\sum_{l=k+1}^{4} \frac{\beta_j c_{ji_k}}{||\boldsymbol{c}_j||}\Big(\frac{-\beta_j}{||\boldsymbol{c}_j||}\Big)\Big[\Big(1 + \frac{2}{\beta_j ||\boldsymbol{c}_j||}\Big)\Big(1 + \frac{1}{\beta_j ||\boldsymbol{c}_j||}\Big) + \Big(\frac{-1}{\beta_j ||\boldsymbol{c}_j||}\Big)^2\Big] + 3\Big(\frac{-\beta_j}{||\boldsymbol{c}_j||}\Big)^2\Big(1 + \frac{1}{\beta_j ||\boldsymbol{c}_j||}\Big) \\ &= e^{-\beta_j ||\boldsymbol{c}_j||}(\alpha_1^4 s_1^4 + \alpha_2^4 s_2^4 + \alpha_3^4 s_3^4) \end{aligned} \tag{100}$$

terms $\rightarrow$ degree $\downarrow$	$t = 1$	$t = 2$	$t = 3$
$D = 1$	$\alpha_1^1 = 1$		
$D = 2$	$\alpha_1^2 = (1 + \frac{1}{\beta_j \|\|\boldsymbol{c}_j\|\|})\alpha_1^1 + \frac{d\alpha_1^1}{dz\|_0}$	$\alpha_2^2 = \frac{-\beta_j}{\|\|\boldsymbol{c}_j\|\|}\alpha_1^1$	
$D = 3$	$\alpha_1^3 = (1 + \frac{2}{\beta_j \|\|\boldsymbol{c}_j\|\|})\alpha_1^2 + \frac{d\alpha_1^2}{dz\|_0}$	$\alpha_2^3 = \frac{-\beta_j}{\|\|\boldsymbol{c}_j\|\|}\alpha_1^2$	
$D = 4$	$\alpha_1^4 = (1 + \frac{3}{\beta_j \|\|\boldsymbol{c}_j\|\|})\alpha_1^3 + \frac{d\alpha_1^3}{dz\|_0}$	$\alpha_2^4 = \frac{-\beta_j}{\|\|\boldsymbol{c}_j\|\|}\alpha_1^3$	$\alpha_3^4 = 3(\frac{-\beta_j}{\|\|\boldsymbol{c}_j\|\|})^2\alpha_1^2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
D	$\alpha_1^D = (1 + \frac{D-1}{\beta_j \|\|\boldsymbol{c}_j\|\|})\alpha_1^{D-1} + \frac{d\alpha_1^{D-1}}{dz\|_0}$	$\alpha_2^D = \frac{-\beta_j}{\|\|\boldsymbol{c}_j\|\|}\alpha_1^{D-1}$	$\alpha_3^D = 3(\frac{-\beta_j}{\|\|\boldsymbol{c}_j\|\|})^2\alpha_1^{D-2}$

TABLE XII
DEFINITION OF SUBEXPRESSIONS FOR α_t^D

terms $\rightarrow$ degree $\downarrow$	$t=1$	$t=2$	$t=3$
$D=1$	$s_1^1 = \prod_{k=1}^{1} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|}$		
$D=2$	$s_1^2 = \prod_{k=1}^{2} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|}$	$s_2^2 = \prod_{l=1}^{0}(\sum_{k=l}^{2} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|})$	
$D=3$	$s_1^3 = \prod_{k=1}^{3} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|}$	$s_2^3 = \prod_{l=1}^{1}(\sum_{k=l}^{3} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|})$	
$D=4$	$s_1^4 = \prod_{k=1}^{4} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|}$	$s_2^4 = \prod_{l=1}^{2}(\sum_{k=l}^{4} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|})$	$s_3^4 = \prod_{l=1}^{0}(\sum_{k=l}^{4} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
D	$s_1^D = \prod_{k=1}^{D} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|}$	$s_2^{D-2} = \prod_{l=1}^{D-2}(\sum_{k=l}^{D} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|})$	$s_3^D = \prod_{l=1}^{D-4}(\sum_{k=l}^{D} \frac{\beta_j c_{ji_k}}{\|\|\boldsymbol{c_j}\|\|})$

TABLE XIII
DEFINITION OF SUBEXPRESSIONS FOR s_t^D

8) Vanilla Recurrent Layer with Binary Activation: The subsequent derivatives that populate a Taylor expansion of the recurrent layer are identical to feed-forward derivatives with the exchange of the weight matrix and state representation. The only coefficient term to appear in the Taylor expansion is the 0^{th} order derivative evaluated at the origin, given in (101).

$$y_j = f_j(z_l = 0) = a_j^0 = \begin{cases} 1, & \text{if } \sum_{i=1}^{n+m} v_{jl} z_l + b_j \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{101}$$

9) Vanilla Recurrent Layer with ReLU Activation: The Taylor approximation series truncates after the 1^{st} order term. The 0^{th} and 1^{st} order terms are given in (102) and (103). Much like the binary expression, the Taylor approximation yields an exact representation to the original function expression.

$$a_j^0 = \begin{cases} b_j, & \text{if } \sum_{l=1}^{n+m} v_{jl} z_l + b_j \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{102}$$

$$A_{jl}^1 = \begin{cases} v_{jl}, & \text{if } \sum_{l=1}^{n+m} v_{jl} z_l + b_j \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{103}$$

10) Vanilla Recurrent Layer with Sigmoid Activation: The 0^{th} to 2^{nd} order terms are given in (104) to (106) to give a sense of the iterative pattern. The higher order polynomial coefficient tensors are iteratively derived with immediately previous polynomial coefficient tensor, given in (107). These equations fully define the polynomial coefficient tensors for any arbitrary order term to populate the entire Taylor approximation series.

$$a_j^0 = f_j(z_l = 0) = \sigma(b_j) \tag{104}$$

$$A_{jl}^1 = \frac{\partial f_j}{\partial z_l}\bigg|_{z_l=0} = v_{jl}\sigma(b_j)(1-\sigma(b_j)) (105)$$

$$A_{jl_1l_2}^2 = \frac{\partial}{\partial z_{l_2}}[\frac{\partial f_j}{\partial z_{l_1}}]\bigg|_{z_{l_1},x_{l_2}=0} = v_{jl_1}v_{jl_2}(1-2\sigma(b_j))\sigma(b_j)(1-\sigma(b_j)) (106)$$

$$A_{jl_1\cdots l_d}^d = \frac{\partial A_{jl_1\cdots l_{d-1}}^{d-1}}{\partial \sigma} v_{jl_d}\sigma(b_j)(1-\sigma(b_j)) \tag{107}$$

11) Vanilla Recurrent Layer with Tanh Activation: The 0^{th} to 2^{nd} and d^{th} order terms are given in (108) to (111) to give a sense of the iterative pattern. These equations fully define the polynomial coefficient tensors for any arbitrary order term to populate the entire Taylor approximation series.

$$a_j^0 = f_j(z_l = 0) = \sigma(b_j) \tag{108}$$

$$A_{jl}^1 = \frac{\partial f_j}{\partial z_l}\bigg|_{z_l=0} = v_{jl}(1-\sigma^2(b_j)) (109)$$

$$A_{jl_1l_2}^2 = \frac{\partial}{\partial z_{l_2}}[\frac{\partial f_j}{\partial z_{l_1}}]\bigg|_{z_{l_1},z_{l_2}=0} = v_{jl_1}v_{jl_2}(-2\sigma(b_j))(1-\sigma^2(b_j)) (110)$$

$$A_{jl_1\cdots l_d}^d = \frac{\partial A_{jl_1\cdots l_{d-1}}^{d-1}}{\partial \sigma} v_{jl_d}(1-\sigma^2(b_j)) \tag{111}$$

C. Multilayer Approximation

The coefficient matrices a_j^0 and A_{jl}^1 in (125) are given in (112) and (113). The coefficient matrices a_l^0, A_{li}^1, $A_{li_1i_2}^2$, to $A_{li_1\cdots i_d}^d$ from (126) are given in (114) to (117). The Taylor series expansion of the network with the sequential layers prior to distributing coefficient tensors is given in (118).

$$a_j^0 = f_j(0) = b_j^O \tag{112}$$

$$A_{jl}^1 = \frac{\partial f_j}{\partial z_l}\Big|_{z_l=0} = w_{jl}^O (113)$$

$$a_l^0 = f_l(0) = \sigma(b_l^I) \tag{114}$$

$$A_{li_1}^1 = \frac{\partial f_l}{\partial x_{i_1}}\Big|_{x_{i_1}=0} = w_{li_1}^I(1-\sigma(b_l^I)^2)(115)$$

$$A_{li_1i_2}^2 = \frac{\partial}{\partial x_{i_2}}[\frac{\partial f_l}{\partial x_{i_1}}]\Big|_{x_{i_1},x_{i_2}=0} = w_{li_1}^I w_{li_2}^I(-2\sigma(b_l^I)+2\sigma(b_l^I)^3)(116)$$

$$A_{li_1\cdots i_d}^d = \frac{\partial A_{li_1\cdots i_{d-1}}^d}{\partial \sigma} w_{li_d}^I w(1-\sigma(b_l^I)^2) \tag{117}$$

$$y_j = b_j^O + \sum_l w_{jl}^O[\sigma(b_l^I) + \sum_{i_1} w_{li_1}^I(1-\sigma(b_l^I)^2)x_{i_1} + \frac{1}{2!}\sum_{i_1}\sum_{i_2} w_{li_1}^I w_{li_2}^I(-2\sigma(b_l^I)+2\sigma(b_l^I)^3)x_{i_1i_2}^{\otimes 2} + \cdots + R(x_i)] \tag{118}$$

The coefficient matrices a_j^0 to $A_{ji_1\cdots i_d}^d$ in (130) are given in (119) and (124).

$$a_j^0 = f_j(0) = \sum_l w_{jl}^O \sigma(b_l^I) + b_j^O \tag{119}$$

$$A_{ji_1}^1 = \frac{\partial f_j}{\partial x_{i_1}}\Big|_{x_i=0} = \sum_l w_{jl}^O w_{li_1}^I(1-\sigma(b_l^I)^2)(120)$$

$$A_{ji_1i_2}^2 = \frac{\partial}{\partial x_{i_2}}[\frac{\partial f_j}{\partial x_{i_1}}]\Big|_{x_i=0} = \sum_l w_{jl}^O w_{li_1}^I w_{li_2}^I(-2\sigma(b_l^I)+2\sigma(b_l^I)^3)(121)$$

$$A_{ji_1i_2i_3}^3 = \frac{\partial}{\partial x_{i_3}}[\frac{\partial}{\partial x_{i_2}}[\frac{\partial f_j}{\partial x_{i_1}}]]\Big|_{x_i=0} = \sum_l w_{jl}^O w_{li_1}^I w_{li_2}^I w_{li_3}^I(-2+8\sigma(b_l^I)^2-6\sigma(b_l^I)^4)(122)$$

$$A_{ji_1i_2i_3i_4}^4 = \frac{\partial}{\partial x_{i_4}}[\frac{\partial}{\partial x_{i_3}}[\frac{\partial}{\partial x_{i_2}}[\frac{\partial f_j}{\partial x_{i_1}}]]]\Big|_{x_i=0} = \sum_l w_{jl}^O w_{li_1}^I w_{li_2}^I w_{li_3}^I w_{li_4}^I(16\sigma(b_l^I)-40\sigma(b_l^I)^3+24\sigma(b_l^I)^5)(123)$$

$$A_{ji_1\cdots i_d}^d = \frac{\partial A_{ji_1\cdots i_{d-1}}^d}{\partial \sigma} w_{li_d}^I w(1-\sigma(b_l^I)^2) \tag{124}$$

D. Derivation Validation

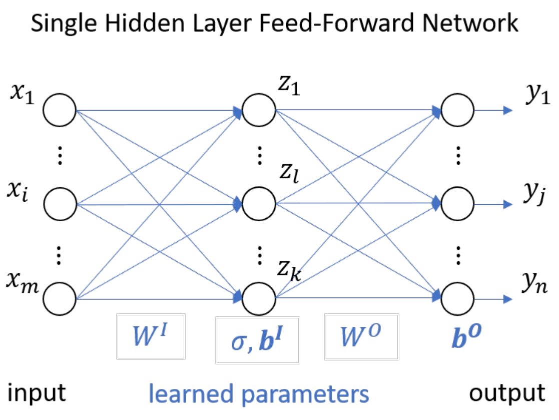

Fig. 10. A single hidden layer neural network configuration labeling input, intermediate state, output, and learned parameters

A popular choice of a single hidden layer neural network configuration includes a hyperbolic tangent hidden layer and a linear output perceptron layer. To validate the overall Taylor series expansion of the two layers, a separate Taylor series expansion is derived from a function mapping the input directly to the output. The Taylor series expansion of each individual layer of this multi-layer neural network is separated into (125) and (126). The overall Taylor series expansion of the two individual layers is given in (127). By substituting the coefficient matrices from (112) to (116) [Appendix C] into the combined Taylor expansion in (127) and after distributing the coefficient tensors, the Taylor series expansion of the network with the sequential layers is given in (128).

$$y_j = f_j(z_l) = a_j^0 + \sum_l A_{jl}^1 z_l \tag{125}$$

$$z_l = f_l(x_i) = a_l^0 + \sum_{i_1} A_{li_1}^1 x_{i_1} + \frac{1}{2!}\sum_{i_1}\sum_{i_2} A_{li_1i_2}^2 x_{i_1i_2}^{\otimes 2} + \cdots + R(x_i) \tag{126}$$

$$y_j = f_j(f_l(x_i)) = a_j^0 + \sum_l A_{jl}^1[a_l^0 + \sum_{i_1} A_{li_1}^1 x_{i_1} + \frac{1}{2!}\sum_{i_1}\sum_{i_2} A_{li_1i_2}^2 x_{i_1i_2}^{\otimes 2} + \cdots + R(x_i)] \tag{127}$$

$$\begin{aligned} y_j =& b_j^O + \sum_l w_{jl}^O \sigma(b_l^I) \\ &+ \sum_l \sum_{i_1} w_{jl}^O w_{li_1}^I (1 - \sigma(b_l^I)^2) x_{i_1} \\ &+ \frac{1}{2!}\sum_l\sum_{i_1}\sum_{i_2} w_{jl}^O w_{li_1}^I w_{li_2}^I (-2\sigma(b_l^I) + 2\sigma(b_l^I)^3) x_{i_1i_2}^{\otimes 2} \\ &+ \cdots + R(x_i) \end{aligned} \tag{128}$$

To derive the Taylor approximation of the multilayer network directly, the single hidden layer neural network nonlinear function of this system is explicitly given in (129). The Taylor expansion directly of the explicit nonlinear input to output mapping is given in (130). The coefficient matrices a_j^0 to $A_{ji_1\cdots i_d}^d$ in (130) are given in (119) and (124) [Appendix C]. The Taylor series expansion of the two sequential individual layers and the overall multilayer network are identical, seen by comparing (128) and (131).

$$y_j = f_j(x_i) = \sum_l w_{jl}^O \sigma(\sum_i w_{li}^I x_i + b_l^I) + b_j^O \tag{129}$$

$$y_j = f_j(x_i) = a_j^0 + \sum_{i_1} A_{ji_1}^1 x_{i_1} + \frac{1}{2!}\sum_{i_1}\sum_{i_2} A_{ji_1i_2}^2 x_{i_1i_2}^{\otimes 2} + \cdots + R(x_i) \tag{130}$$

$$\begin{aligned} y_j =& \sum_l w_{jl}^O \sigma(b_l^I) + b_j^O \\ &+ \sum_{i_1} \sum_l w_{jl}^O w_{li_1}^I (1 - \sigma(b_l^I)^2) x_{i_1} \\ &+ \frac{1}{2!}\sum_{i_1}\sum_{i_2}\sum_l w_{jl}^O w_{li_1}^I w_{li_2}^I (-2\sigma(b_l^I) + 2\sigma(b_l^I)^3) x_{i_1i_2}^{\otimes 2} \\ &+ \cdots + R(x_i) \end{aligned} \tag{131}$$

E. FLOPS Derivation

1) Polynomial Directly from Data: The entire process of finding the least-squares solution involves a sequence of large matrix operations. The entire expression is reiterated here for convenience

$$A_p = Y(X^{\otimes d})^T((X^{\otimes d})(X^{\otimes d})^T)^{-1}$$

This expression can be broken down into the following sequence of steps with their associated flops.

$$\mathcal{O}((X^{\otimes d})(X^{\otimes d})^T) = (2t-1)\tfrac{m^{2d}}{d!} \tag{132}$$

$$\mathcal{O}(((X^{\otimes d})(X^{\otimes d})^T)^{-1}) = \tfrac{m^{3d}}{d!} \tag{133}$$

$$\mathcal{O}(Y(X^{\otimes d})) = (2t-1)n\tfrac{m^d}{d!} \tag{134}$$

$$\mathcal{O}(Y(X^{\otimes d})((X^{\otimes d})(X^{\otimes d})^T)^{-1}) = (2t-1)\tfrac{m^{2d}}{d!} + \tfrac{m^{3d}}{d!} + (2t-1)n\tfrac{m^d}{d!} \tag{135}$$

$$\approx t\tfrac{m^d}{d!} + \tfrac{m^{3d}}{d!} \tag{136}$$

2) Polynomial from Neural Network: In calculating the polynomial coefficients, each polynomial term of ascending degree must be calculated given in (137), where y is the matrix output, $A^0, A^1, \cdots, A^d$ are the polynomial coefficients from NN parameters, a are the multinomial coefficients, x is the matrix input, and d is the degree of the polynomial expression.

$$y = \begin{bmatrix} A^0 \; A^1 \cdots A^d \end{bmatrix} (a \odot \begin{bmatrix} 1 \\ x \\ \vdots \\ x^{\otimes d} \end{bmatrix}) \tag{137}$$

To derive the general form, flops are calculated for each polynomial coefficient matrix in ascending order, where m is the input state size, n is the output state size, and k is the neural network's number of neurons. The flops from calculating A^0 is given in .

$$\mathcal{O}(A^0_j = \sum_l w^o_{jl}\sigma(b^I_l) + b^o_j) = nk \tag{138}$$

The flops from calculating A^1 is given in (139).

$$\mathcal{O}(A^1_{ji_1} = \sum_l w^o_{jl} w^I_{li_1}(1 - \sigma(b^I_l)^2) = nkm \tag{139}$$

The flops from calculating A^2 is given in (140).

$$\mathcal{O}(A^2_{ji_1i_2} = \sum_l -2w^o_{jl} w^I_{li_1} w^I_{li_2}(\sigma(b^I_l) - \sigma(b^I_l)^3)) = nkm^2 \tag{140}$$

One can infer the sum of flops from lower order terms that the d^{th} term is given in (141).

$$\mathcal{O}(A^d) = nkm^d \tag{141}$$

The total flops to calculate all terms is given in (142).

$$\mathcal{O}(\begin{bmatrix} A^0 \; A^1 \cdots A^d \end{bmatrix}) = nk + nkm + nkm^2 + \cdots + nkm^d \tag{142}$$

If $m > 1$, this sum does not converge but diverges. The dominant term is thus the last, largest term nkm^d.