`vanilla` Recurrent Neural-Network: back-propagation derivation

Harsha Vardhan

April 27, 2022

Abstract

This document contains derivation of the gradients for a vanilla recurrent neural-network, using back-propagation (or reverse-mode differentiation). For the implementation of the neural-network see the accompanying notebooks.

1 Network Architecture

A Recurrent Neural-Network can take many different architectures, and here we are considering a common many-to-many architecture. Also, for a brief description of other common RNN architectures, see section-4.

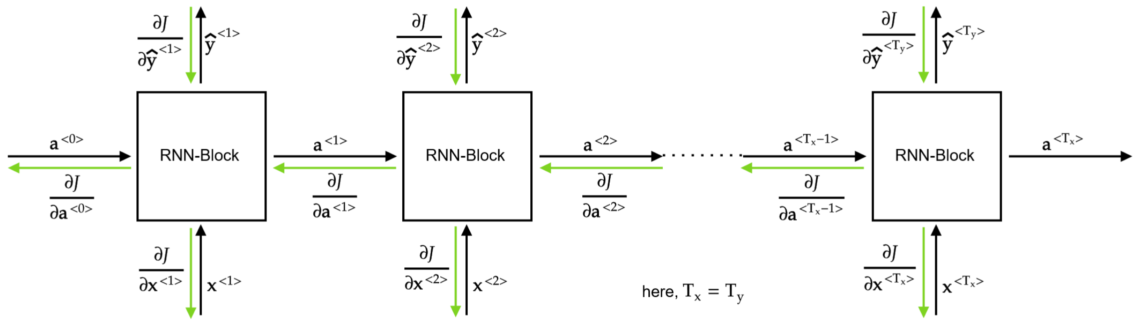

Figure 1: a recurrent neural network with many-to-many architecture, such that $T_x = T_y$. Where, T_x is the number of input time-steps, and T_y is the number of output time-steps. Also, here 'RNN-Block' would be the block for vanilla RNN; see section-1.1 for more information.

1.1 `vanilla` RNN Block

A RNN-block could be of many types, such as the `vanilla`-block, LSTM-block, GRU-block, etc. In this section we define the `vanilla` RNN-Block, since in this article we are dealing with a vanilla RNN.

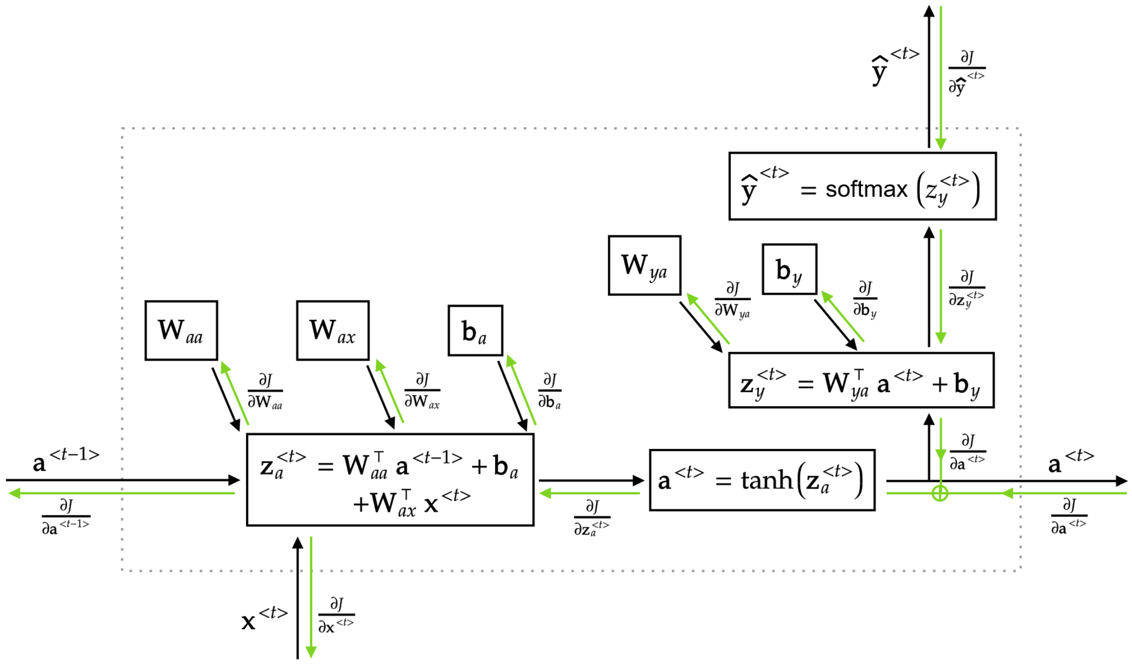

Figure 2: a `vanilla` RNN-block along with the back-propagation routes. Note that the gradient $\frac{\partial J}{\partial \mathbf{a}^{\langle t \rangle}}$ come from two directions - the $(t+1)^{th}$ time-step and from t^{th} time-steps' output - both of which get added.

2 Forward Propagation

Given $\mathbf{a}^{\langle t-1 \rangle}$ from the $(t-1)^{th}$ time-step, the equations for forward propagation through the t^{th} time-step, are as follows:

$$\begin{aligned}
\mathbf{z}_a^{\langle t \rangle} &= \mathbf{W}_{aa}^{\intercal}\, \mathbf{a}^{\langle t-1 \rangle} + \mathbf{W}_{ax}^{\intercal}\, \mathbf{x}^{\langle t \rangle} + \mathbf{b}_a \\
\mathbf{a}^{\langle t \rangle} &= \tanh(\mathbf{z}_a^{\langle t \rangle}) \\
\mathbf{z}_y^{\langle t \rangle} &= \mathbf{W}_{ya}^{\intercal}\, \mathbf{a}^{\langle t \rangle} + \mathbf{b}_y \\
\hat{\mathbf{y}}^{\langle t \rangle} &= \text{softmax}(\mathbf{z}_y^{\langle t \rangle})
\end{aligned}$$

where,

$\mathbf{x}^{\langle t \rangle}$	a $(n_x, 1)$ dimensional input-vector, at the t^{th} time-step
$\mathbf{W}_{ax}$	a (n_x, n_a) dimensional weights-matrix
$\mathbf{W}_{aa}$	a (n_a, n_a) dimensional weights-matrix
$\mathbf{W}_{ya}$	a (n_a, n_y) dimensional weights-matrix
$\mathbf{b}_a$	a $(n_a, 1)$ dimensional bias-vector
$\mathbf{b}_y$	a $(n_y, 1)$ dimensional bias-vector

Notation: in the above equations, the sub-script for matrices and vectors must be interpreted as follows:

- Let $\mathbf{A}_{pq}$ be a matrix. Then the sub-script 'p' denotes that the matrix is used for computing some p-like quantity, and the sub-script 'q' denotes that the matrix is multiplied by some q-like quantity.

- Let $\mathbf{a}_q$ be a vector. Then the sub-script 'q' denotes that the vector is used for computing some q-like quantity.

3 Optimization: gradient-descent

The optimization is performed according to the following equations:

$$\mathbf{W}_{aa} := \mathbf{W}_{aa} - \alpha \sum_{t=1}^{\mathrm{T}_x} \nabla^{\langle t \rangle}_{\mathrm{W}_{aa}} J$$

$$\mathbf{W}_{ax} := \mathbf{W}_{ax} - \alpha \sum_{t=1}^{\mathrm{T}_x} \nabla^{\langle t \rangle}_{\mathrm{W}_{ax}} J$$

$$\mathbf{W}_{ya} := \mathbf{W}_{ya} - \alpha \sum_{t=1}^{\mathrm{T}_x} \nabla^{\langle t \rangle}_{\mathrm{W}_{ya}} J$$

$$\mathbf{b}_{a} := \mathbf{b}_{a} - \alpha \sum_{t=1}^{\mathrm{T}_x} \nabla^{\langle t \rangle}_{\mathrm{b}_{a}} J$$

$$\mathbf{b}_{y} := \mathbf{b}_{y} - \alpha \sum_{t=1}^{\mathrm{T}_x} \nabla^{\langle t \rangle}_{\mathrm{b}_{y}} J$$

Notice that we are summing the gradients over all the time-steps before updating the parameters.

3.1 Back-propagation

The gradients in the above equations, for any particular time-step t, are derived using back-propagation as follows:

3.1.1 Computing $\frac{\partial J}{\partial \hat{\mathbf{y}}^{\langle t \rangle}}$

Since, $\hat{\mathbf{y}}^{\langle t \rangle}$ is computed using the softmax() activation-function, the loss J is computed using the cross-entropy loss. Also, let $\mathbf{y}^{\langle t \rangle}$ be the output-label corresponding to the t^{th} time-step. Then,

$$\begin{aligned}\frac{\partial J}{\partial \hat{\mathbf{y}}^{\langle t \rangle}} &= \begin{bmatrix} \frac{\partial J}{\partial \hat{y}_1^{\langle t \rangle}} & \frac{\partial J}{\partial \hat{y}_2^{\langle t \rangle}} & \cdots & \frac{\partial J}{\partial \hat{y}_{n_y}^{\langle t \rangle}} \end{bmatrix} \\ &= \begin{bmatrix} \frac{y_1^{\langle t \rangle}}{\hat{y}_1^{\langle t \rangle}} & \frac{y_2^{\langle t \rangle}}{\hat{y}_2^{\langle t \rangle}} & \cdots & \frac{y_{n_y}^{\langle t \rangle}}{\hat{y}_{n_y}^{\langle t \rangle}} \end{bmatrix}\end{aligned}$$

3.1.2 Computing $\frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}}$

$$\frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}} = \frac{\partial J}{\partial \hat{\mathbf{y}}^{\langle t \rangle}} \frac{\partial \hat{\mathbf{y}}^{\langle t \rangle}}{\partial \mathbf{z}_y^{\langle t \rangle}}$$

For the derivative of the softmax() function, i.e. $\frac{\partial \hat{\mathbf{y}}^{\langle t \rangle}}{\partial \mathbf{z}_y^{\langle t \rangle}}$, see `..\notes\softmax-function.ipynb`.

3.1.3 Computing $\frac{\partial J}{\partial \mathbf{W}_{ya}}$, $\frac{\partial J}{\partial \mathbf{b}_y}$, and $\frac{\partial J}{\partial \mathbf{a}^{\langle t \rangle}}$

$$\mathrm{d}J = \mathrm{tr}\left(\frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}} \mathrm{d}\mathbf{z}_y^{\langle t \rangle}\right)$$

where, $\mathrm{d}\mathbf{z}_y^{\langle t \rangle}$ can be expanded as follows:

$$\begin{aligned}\mathrm{d}\mathbf{z}_y^{\langle t \rangle} &= \mathrm{d}\left(\mathbf{W}_{ya}^{\intercal}\, \mathbf{a}^{\langle t \rangle} + \mathbf{b}_y\right) \\ &= \mathrm{d}\mathbf{W}_{ya}^{\intercal}\, \mathbf{a}^{\langle t \rangle} + \mathbf{W}_{ya}^{\intercal}\, \mathrm{d}\mathbf{a}^{\langle t \rangle} + \mathrm{d}\mathbf{b}_y\end{aligned}$$

when differentiating w.r.t. $\mathbf{W}_{ya}$, we have $\mathrm{d}\mathbf{a}^{\langle t \rangle} = 0$, and $\mathrm{d}\mathbf{b}_y = 0$. So,

$$\begin{aligned}\mathrm{d}J &= \mathrm{tr}\left(\frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}} \mathrm{d}\mathbf{W}_{ya}^{\intercal} \mathbf{a}^{\langle t \rangle}\right) \\ &= \mathrm{tr}\left(\left(\mathbf{a}^{\langle t \rangle}\right)^{\intercal} \mathrm{d}\mathbf{W}_{ya} \left(\frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}}\right)^{\intercal}\right) \\ &= \mathrm{tr}\left(\left(\frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}}\right)^{\intercal} \left(\mathbf{a}^{\langle t \rangle}\right)^{\intercal} \mathrm{d}\mathbf{W}_{ya}\right) \\ \implies \frac{\partial J}{\partial \mathbf{W}_{ya}} &= \left(\frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}}\right)^{\intercal} \left(\mathbf{a}^{\langle t \rangle}\right)^{\intercal}\end{aligned}$$

when differentiating w.r.t. $\mathbf{b}_y$, we have $\mathrm{d}\mathbf{a}^{\langle t \rangle} = 0$, and $\mathrm{d}\mathbf{W}_{ya} = 0$. So,

$$\begin{aligned}\mathrm{d}J &= \mathrm{tr}\left(\frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}} \mathrm{d}\mathbf{b}_y\right) \\ \implies \frac{\partial J}{\partial \mathbf{b}_y} &= \frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}}\end{aligned}$$

when differentiating w.r.t. $\mathbf{a}^{\langle t \rangle}$, we have $\mathrm{d}\mathbf{W}_{ya} = 0$, and $\mathrm{d}\mathbf{b}_y = 0$. So,

$$\begin{aligned}\mathrm{d}J &= \mathrm{tr}\left(\frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}} \mathbf{W}_{ya}^{\intercal} \mathrm{d}\mathbf{a}^{\langle t \rangle}\right) \\ \implies \frac{\partial J}{\partial \mathbf{a}^{\langle t \rangle}} &= \frac{\partial J}{\partial \mathbf{z}_y^{\langle t \rangle}} \mathbf{W}_{ya}^{\intercal}\end{aligned}$$

3.1.4 Computing $\frac{\partial J}{\partial \mathbf{z}_a^{\langle t \rangle}}$

$$\frac{\partial J}{\partial \mathbf{z}_a^{\langle t \rangle}} = \frac{\partial J}{\partial \mathbf{a}^{\langle t \rangle}} \frac{\partial \mathbf{a}^{\langle t \rangle}}{\partial \mathbf{z}_a^{\langle t \rangle}}$$

In the above expression, we have

- $\frac{\partial J}{\partial \mathbf{a}^{\langle t \rangle}}$ is computed by taking the sum of the gradient propagation from $\hat{\mathbf{y}}^{\langle t \rangle}$ with the gradient propagating from the $(t+1)^{th}$ time-step. Note that when $t = \mathrm{T}_x$, the gradient from the $(t+1)^{th}$ time-step will be zero.

- $\frac{\partial \mathbf{a}^{\langle t \rangle}}{\partial \mathbf{z}_a^{\langle t \rangle}}$ is the derivative across the tanh() function. For the derivation of this derivative see `..\notes\hyperbolic-tangent-function.ipynb`.

3.1.5 Computing $\frac{\partial J}{\partial \mathbf{W}_{aa}}$, $\frac{\partial J}{\partial \mathbf{W}_{ax}}$, $\frac{\partial J}{\partial \mathbf{b}_a}$, and $\frac{\partial J}{\partial \mathbf{a}^{\langle t-1\rangle}}$

$$\mathrm{d}J = \operatorname{tr}\left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\mathrm{d}\mathbf{z}_a^{\langle t\rangle}\right)$$

where, $\mathrm{d}\mathbf{z}_a^{\langle t\rangle}$ can be expanded as follows:

$$\begin{aligned}
\mathrm{d}\mathbf{z}_a^{\langle t\rangle} &= \mathrm{d}\left(\mathbf{W}_{aa}^\intercal\,\mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{ax}^\intercal\,\mathbf{x}^{\langle t\rangle} + \mathbf{b}_a\right)\\
&= \mathrm{d}\mathbf{W}_{aa}^\intercal\,\mathbf{a}^{\langle t-1\rangle} + \mathbf{W}_{aa}^\intercal\,\mathrm{d}\mathbf{a}^{\langle t-1\rangle} + \mathrm{d}\mathbf{W}_{ax}^\intercal\,\mathbf{x}^{\langle t\rangle} + \mathrm{d}\mathbf{b}_a
\end{aligned}$$

when differentiating w.r.t. $\mathbf{W}_{aa}$, we have $\mathrm{d}\mathbf{a}^{\langle t-1\rangle} = 0$, $\mathrm{d}\mathbf{W}_{ax} = 0$, and $\mathrm{d}\mathbf{b}_a = 0$. So,

$$\begin{aligned}
\mathrm{d}J &= \operatorname{tr}\left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\mathrm{d}\mathbf{W}_{aa}^\intercal\,\mathbf{a}^{\langle t-1\rangle}\right)\\
&= \operatorname{tr}\left(\left(\mathbf{a}^{\langle t-1\rangle}\right)^\intercal \mathrm{d}\mathbf{W}_{aa}\,\left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\right)^\intercal\right)\\
&= \operatorname{tr}\left(\left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\right)^\intercal \left(\mathbf{a}^{\langle t-1\rangle}\right)^\intercal \mathrm{d}\mathbf{W}_{aa}\right)\\
\Longrightarrow \frac{\partial J}{\partial \mathbf{W}_{aa}} &= \left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\right)^\intercal \left(\mathbf{a}^{\langle t-1\rangle}\right)^\intercal
\end{aligned}$$

when differentiating w.r.t. $\mathbf{W}_{ax}$, we have $\mathrm{d}\mathbf{a}^{\langle t-1\rangle} = 0$, $\mathrm{d}\mathbf{W}_{aa} = 0$, and $\mathrm{d}\mathbf{b}_a = 0$. So,

$$\begin{aligned}
\mathrm{d}J &= \operatorname{tr}\left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\mathrm{d}\mathbf{W}_{ax}^\intercal\,\mathbf{x}^{\langle t\rangle}\right)\\
&= \operatorname{tr}\left(\left(\mathbf{x}^{\langle t\rangle}\right)^\intercal \mathrm{d}\mathbf{W}_{ax}\,\left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\right)^\intercal\right)\\
&= \operatorname{tr}\left(\left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\right)^\intercal \left(\mathbf{x}^{\langle t\rangle}\right)^\intercal \mathrm{d}\mathbf{W}_{ax}\right)\\
\Longrightarrow \frac{\partial J}{\partial \mathbf{W}_{ax}} &= \left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\right)^\intercal \left(\mathbf{x}^{\langle t\rangle}\right)^\intercal
\end{aligned}$$

when differentiating w.r.t. $\mathbf{b}_a$, we have $\mathrm{d}\mathbf{W}_{ax} = 0$, $\mathrm{d}\mathbf{W}_{aa} = 0$, and $\mathrm{d}\mathbf{a}^{\langle t-1\rangle} = 0$. So,

$$\begin{aligned}
\mathrm{d}J &= \operatorname{tr}\left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\mathrm{d}\mathbf{b}_a\right)\\
\Longrightarrow \frac{\partial J}{\partial \mathbf{b}_a} &= \frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}
\end{aligned}$$

when differentiating w.r.t. $\mathbf{a}^{\langle t-1\rangle}$, we have $\mathrm{d}\mathbf{W}_{ax} = 0$, $\mathrm{d}\mathbf{W}_{aa} = 0$, and $\mathrm{d}\mathbf{b}_a = 0$. So,

$$\begin{aligned}
\mathrm{d}J &= \operatorname{tr}\left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\mathbf{W}_{aa}^\intercal\,\mathrm{d}\mathbf{a}^{\langle t-1\rangle}\right)\\
\Longrightarrow \frac{\partial J}{\partial \mathbf{a}^{\langle t-1\rangle}} &= \frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\mathbf{W}_{aa}^\intercal
\end{aligned}$$

3.2 Gradient or Jacobian?

In the above derivations, we have used the numerator layout while performing matrix-derivatives. One of the consequences of this decision is that the derivatives that we have computed are in-fact jacobians and not gradients. Fortunately, gradients are just transpose of jacobians. So, based on our derivations the gradients would be the following:

$$\nabla^{\langle t\rangle}_{\mathrm{W}_{aa}} J = \left(\frac{\partial J}{\partial \mathbf{W}_{aa}}\right)^{\mathsf{T}} = \mathbf{a}^{\langle t-1\rangle}\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}$$

$$\nabla^{\langle t\rangle}_{\mathrm{W}_{ax}} J = \left(\frac{\partial J}{\partial \mathbf{W}_{ax}}\right)^{\mathsf{T}} = \mathbf{x}^{\langle t\rangle}\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}$$

$$\nabla^{\langle t\rangle}_{\mathrm{W}_{ya}} J = \left(\frac{\partial J}{\partial \mathbf{W}_{ya}}\right)^{\mathsf{T}} = \mathbf{a}^{\langle t\rangle}\frac{\partial J}{\partial \mathbf{z}_y^{\langle t\rangle}}$$

$$\nabla^{\langle t\rangle}_{\mathrm{b}_{a}} J = \left(\frac{\partial J}{\partial \mathbf{b}_{a}}\right)^{\mathsf{T}} = \left(\frac{\partial J}{\partial \mathbf{z}_a^{\langle t\rangle}}\right)^{\mathsf{T}}$$

$$\nabla^{\langle t\rangle}_{\mathrm{b}_{y}} J = \left(\frac{\partial J}{\partial \mathbf{b}_{y}}\right)^{\mathsf{T}} = \left(\frac{\partial J}{\partial \mathbf{z}_y^{\langle t\rangle}}\right)^{\mathsf{T}}$$

4 Appendix-I

The following are some of the common architectures for a Recurrent Neural-Network:

Many-to-many: such that $\mathrm{T}_x \neq \mathrm{T}_y$

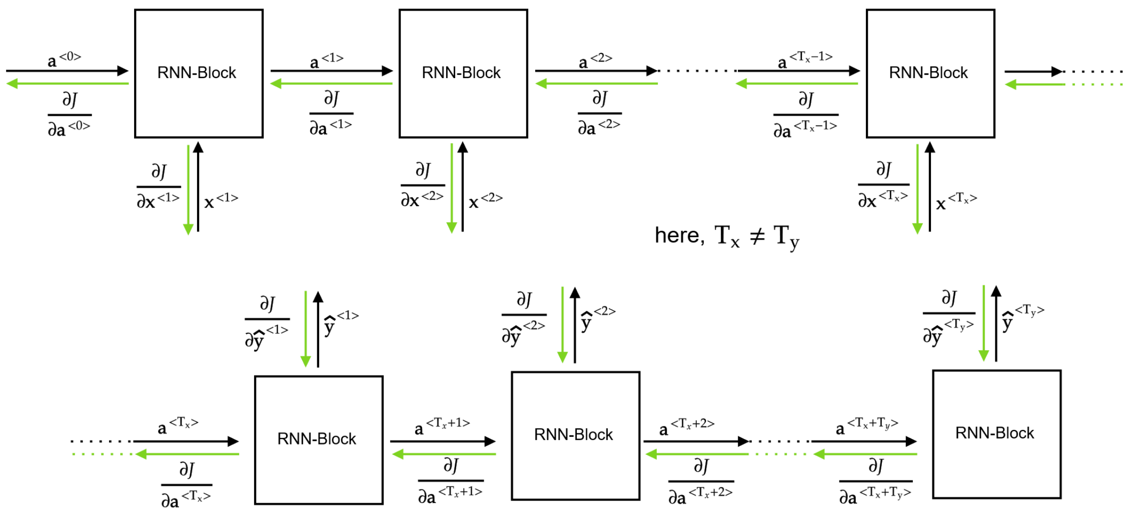

Figure 3: a recurrent neural network with many-to-many architecture, such that $\mathrm{T}_x \neq \mathrm{T}_y$. Where, T_x is the number of input time-steps, and T_y is the number of output time-steps.

Many-to-one: This is a special case for the many-to-many architecture above, obtained by setting $\mathrm{T}_y = 1$, i.e.

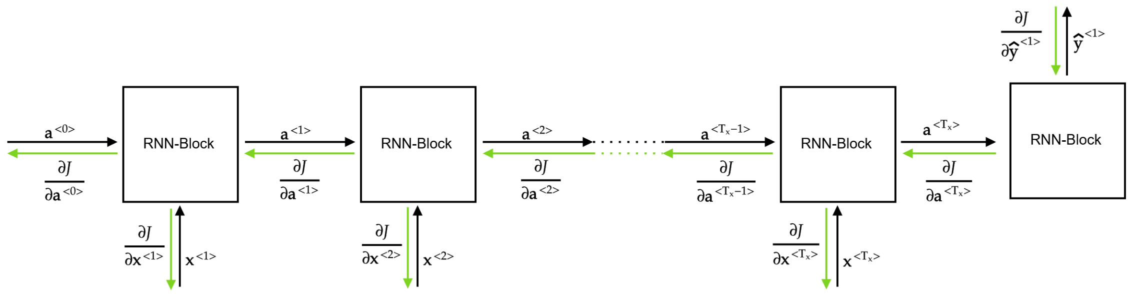

Figure 4: a recurrent neural network with many-to-one architecture.

One-to-many: sometimes, the activation's $\mathbf{a}^{\langle t \rangle}$ are also passed from one time-step to the next.

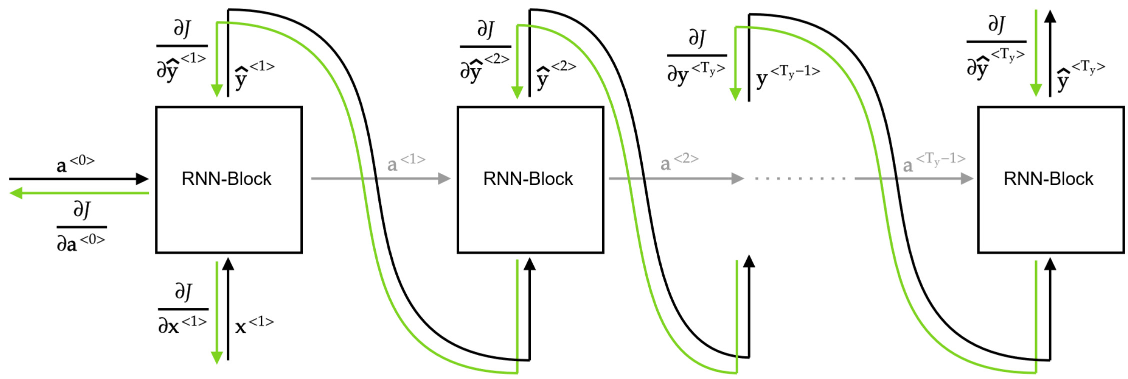

Figure 5: a recurrent neural network with one-to-many architecture. The muted-arrows in the figure denote optional connections between time-steps.

One-to-one: This is a special case for the many-to-many architecture (see section-1), obtained by setting $\mathrm{T}_x = \mathrm{T}_y = 1$, i.e.

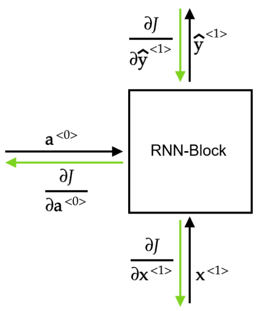

Figure 6: a recurrent neural network with one-to-one architecture. Note that this is essentially a dense neural-network with a single hidden-layer with a tanh() activation function.

4.1 Additional Architectures

Deep-RNN: a network obtained by stacking l many-to-many RNN's

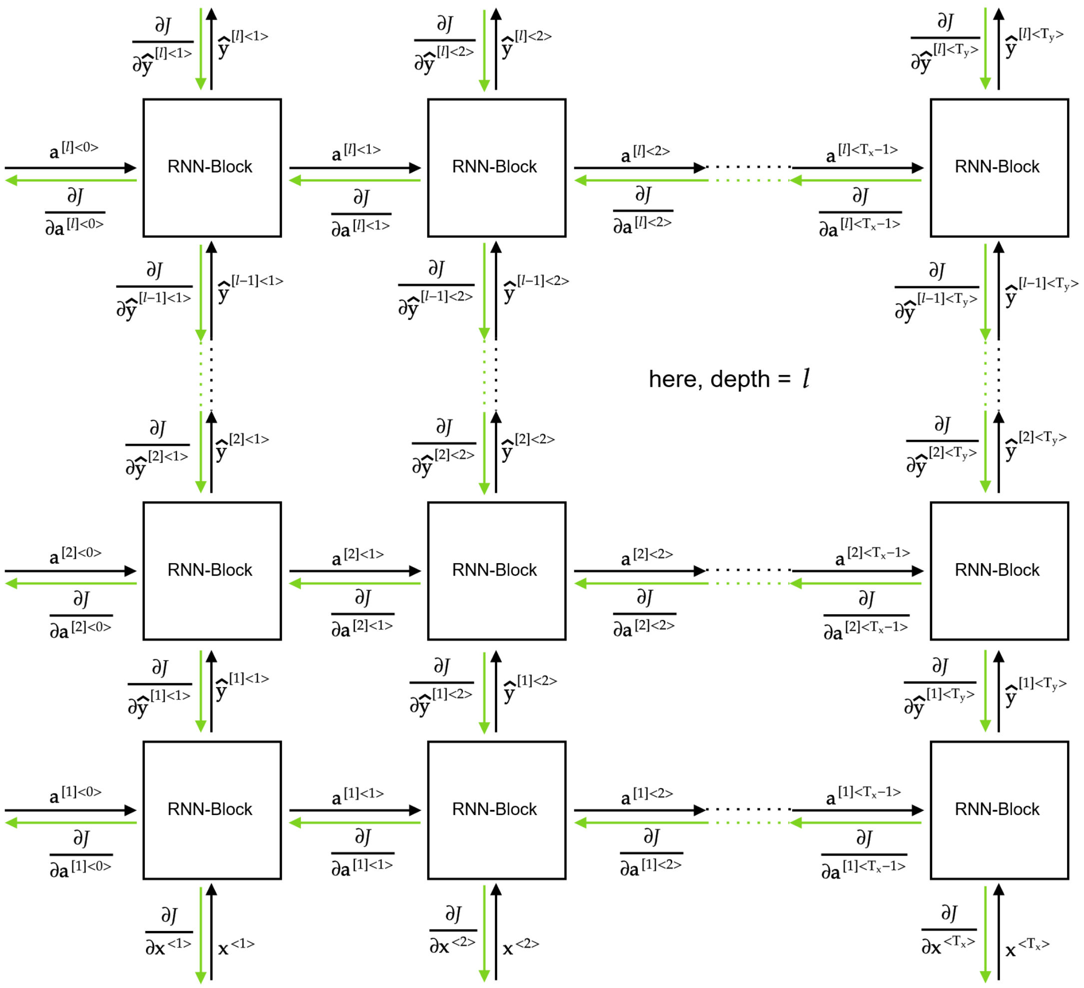

Figure 7: a unidirectional deep recurrent neural-network with a network-depth of l.

Bi-RNN: here, at any time-step t, the affine vector $\mathbf{z}^{\langle t \rangle}$ is computed as follows:

$$\mathbf{z}^{\langle t \rangle} = \mathbf{W}_{ya^f}\, {}^{f}\mathbf{a}^{\langle t \rangle} + \mathbf{W}_{ya^b}\, {}^{b}\mathbf{a}^{\langle \mathrm{T}_x - t \rangle} + \mathbf{b}_y$$

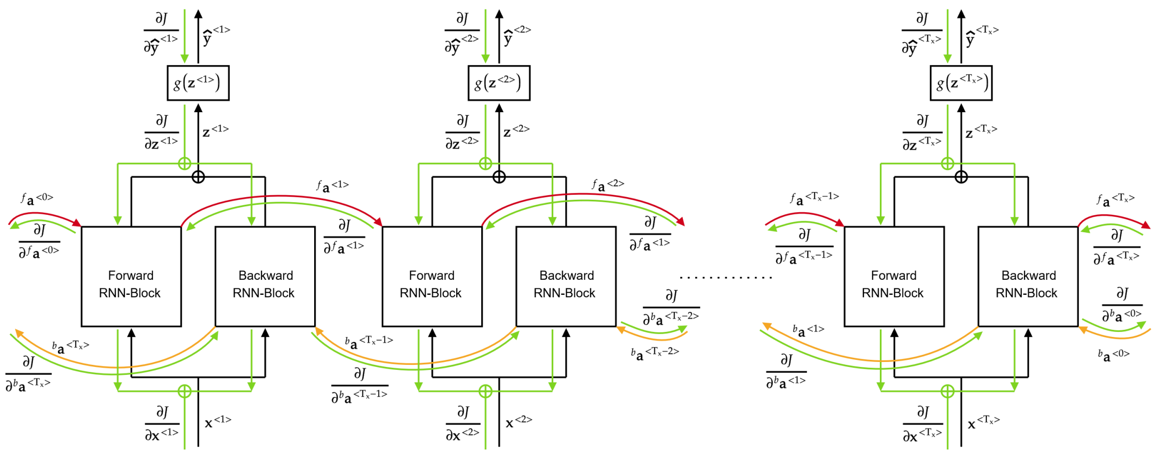

Figure 8: a bi-directional recurrent-neural network with a one-to-one architecture. Here, $g()$ is the activation function.

Deep Bi-RNN: similar to Deep-RNN's, these are obtained by stacking multiple one-to-one Bi-RNN's as layers.