1 APPENDIX

1.1 Derivation of (BetaLinear) rule:

For dense-based neural networks with *linear* activation function, for a given neural network with a layer $N : y = linear(\overline{w}.\overline{x} + \overline{b})$ and $\delta : y \bowtie \overline{n}$, the computed precondition δ' is $\overline{x} \bowtie (\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).\overline{n} - \overline{b}$.

Given, postcondition $\delta : y \bowtie \overline{n}$ and given $N : y = linear(\overline{w}.\overline{x} + \overline{b})$ We know from the weakest precondition equation $\delta' = wp(N, \delta)$.

$$\text{So, } \delta' = wp(N, \delta) \equiv wp(linear(\overline{w}.\overline{x} + \overline{b}), y \bowtie \overline{n})$$

$$\equiv linear(\overline{w}.\overline{x} + \overline{b}) \bowtie \overline{n} \equiv (\overline{w}.\overline{x} + \overline{b}) \bowtie \overline{n}$$

$$\equiv \overline{x} \bowtie (\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).\overline{n}) - \overline{b}; \text{ [From least square solution]} \tag{2.1.1}$$

$$\equiv \overline{x} \bowtie (\overline{\gamma}.\overline{n}) - \overline{b}; [\overline{\gamma} = (\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T)]$$

1.2 Derivation of (BetaRelu) rule:

For dense-based neural network with *relu* activation function, for given $N : y = relu(\overline{w}.\overline{x} + \overline{b})$ and $\delta : y \bowtie \overline{n}$, the computed precondition δ' is $\overline{x} \bowtie ((\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).\overline{n} - \overline{b}) \wedge \overline{x} \bowtie ((\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).(-\overline{b}))$.

Given, postcondition $\delta' : y \bowtie \overline{n}$ and given $N : y = relu(\overline{w}.\overline{x} + \overline{b})$ We know from the weakest precondition equation $\delta' = wp(N, \delta)$.

$$\text{So, } \delta' = wp(N, \delta) \equiv wp(relu(\overline{w}.\overline{x} + \overline{b}), y \bowtie \overline{n})$$

$$\equiv relu(\overline{w}.\overline{x} + \overline{b}) \bowtie \overline{n} \equiv max(0, \overline{w}.\overline{x} + \overline{b}) \bowtie \overline{n}$$

$$\text{We know, if } m = max(x, y) \text{ then,}$$

$$(m = x \wedge (x \geq y) \vee (m = y) \vee (y \geq x)$$

$$\text{So, } \delta' \equiv max(0, \overline{w}.\overline{x} + \overline{b}) \bowtie \overline{n}$$

$$\equiv \{0 \wedge 0 \geq (\overline{w}.\overline{x} + \overline{b})\} \vee \{(\overline{w}.\overline{x} + \overline{b}) \wedge (\overline{w}.\overline{x} + \overline{b}) \geq 0\} \bowtie \overline{n} \tag{2.1.2}$$

$$\text{From (2.1.2), } 0 \wedge 0 \geq (\overline{w}.\overline{x} + \overline{b}) \bowtie \overline{n}$$

$$\equiv 0 \leq \overline{n} \wedge 0 \geq (\overline{w}.\overline{x} + \overline{b}) \bowtie \overline{n} \equiv \overline{w}.\overline{x} + \overline{b} \bowtie \overline{n}$$

$$\equiv \overline{x} \bowtie (\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).\overline{n} - \overline{b} \tag{2.1.3}$$

$$\text{Again, } \{(\overline{w}.\overline{x} + \overline{b}) \wedge (\overline{w}.\overline{x} + \overline{b}) \geq 0\} \bowtie \overline{n}$$

$$\text{So, } \overline{w}.\overline{x} + \overline{b} \bowtie \overline{n} \equiv \overline{x} \bowtie (\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).\overline{n} - \overline{b}; \text{ [From (2.1.1)]}$$

$$\wedge \overline{w}.\overline{x} + \overline{b} \geq 0 \bowtie \overline{n} \equiv \overline{x} \bowtie (\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).(-\overline{b}) \tag{2.1.4}$$

$$\text{From (2.1.3) \& (2.1.4), } P \equiv \overline{x} \bowtie ((\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).\overline{n} - \overline{b})$$

$$\wedge \overline{x} \bowtie ((\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).(-\overline{b}))$$

1.3 Derivation of (BetaSigmoid) rule:

For a dense-based neural network with *sigmoid* activation function, for a given neural network with a single layer $N : y = sigmoid(\overline{w}.\overline{x} + \overline{b})$ and $\delta : y \bowtie \overline{n}$, The computed precondition δ' is $\overline{x} \bowtie (\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).ln(\frac{\overline{n}}{1-\overline{n}}) - \overline{b}$.

Given, postcondition $\delta : y \bowtie \overline{n}$ and given $N : y = sigmoid(\overline{w}.\overline{x} + \overline{b})$ We know from the weakest precondition equation, $\delta' = wp(N, \delta)$.

$$\text{So, } \delta' = wp(N, \delta) \equiv wp(sigmoid(\overline{w}.\overline{x} + \overline{b}), y \bowtie \overline{n})$$

$$\equiv sigmoid(\overline{w}.\overline{x} + \overline{b}) \bowtie \overline{n} \equiv \frac{1}{1 + e^{-(\overline{w}.\overline{x} + \overline{b})}} \bowtie \overline{n}$$

$$\equiv e^{-(\overline{w}.\overline{x} + \overline{b})} \bowtie \frac{1 - \overline{n}}{\overline{n}} \equiv ln(e^{-(\overline{w}.\overline{x} + \overline{b})}) \bowtie ln(\frac{1 - \overline{n}}{\overline{n}})$$

$$\equiv -(\overline{w}.\overline{x} + \overline{b}) \bowtie ln(\frac{\overline{n}}{1 - \overline{n}}) \equiv \overline{w}.\overline{x} + \overline{b} \bowtie ln(\frac{\overline{n}}{1 - \overline{n}})$$

$$\equiv \overline{x} \bowtie (\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).ln(\frac{\overline{n}}{1 - \overline{n}}) - \overline{b}; \text{ [From least square solution]}$$

$$\equiv \overline{x} \bowtie (\overline{\gamma}.ln(\frac{\overline{n}}{1 - \overline{n}}) - \overline{b}); [\overline{\gamma} = (\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T)]$$

1.4 Derivation of (BetaTanh) rule:

For dense-based neural network with *tanh* activation function, for given $N : y = tanh(\overline{w}.\overline{x} + \overline{b})$ and $\delta : y \bowtie \overline{n}$, the computed precondition δ' is $\overline{x} \bowtie ((\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).\frac{1}{2}ln(\frac{\overline{n}-1}{\overline{n}+1})) - \overline{b})$.

Given, postcondition $\delta : y \bowtie \overline{n}$ and given $N : y = tanh(\overline{w}.\overline{x} + \overline{b})$ We know from the weakest precondition equation $\delta' = wp(N, \delta)$.

$$\text{So, } \delta' = wp(N, Q) \equiv wp(tanh(\overline{w}.\overline{x} + \overline{b}), y \bowtie \overline{n})$$

$$\equiv tanh(\overline{w}.\overline{x} + \overline{b}) \bowtie \overline{n} \equiv \frac{e^{(\overline{w}.\overline{x}+\overline{b})} - e^{-(\overline{w}.\overline{x}+\overline{b})}}{e^{(\overline{w}.\overline{x}+\overline{b})} + e^{-(\overline{w}.\overline{x}+\overline{b})}} \bowtie \overline{n}$$

$$\text{Let, } z = \overline{w}.\overline{x} + \overline{b}. \text{ So, } P \equiv \frac{e^z - e^{-z}}{e^z + e^{-z}} \bowtie \overline{n}$$

$$\equiv \frac{2e^z}{-2e^{-z}} \bowtie \frac{\overline{n}+1}{\overline{n}-1} \equiv -ln(\frac{e^z}{e^{-z}}) \bowtie ln(\frac{\overline{n}+1}{\overline{n}-1})$$

$$\equiv -(ln(e^z) - ln(e^{-z})) \bowtie ln(\frac{\overline{n}+1}{\overline{n}-1}) \equiv -2z \bowtie ln(\frac{\overline{n}+1}{\overline{n}-1})$$

$$\equiv \overline{w}.\overline{x} + \overline{b} \bowtie -\frac{1}{2}.ln(\frac{\overline{n}+1}{\overline{n}-1}) \equiv \overline{w}.\overline{x} + \overline{b} \bowtie \frac{1}{2}.ln(\frac{\overline{n}-1}{\overline{n}+1})$$

$$\equiv \overline{x} \bowtie ((\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T).\frac{1}{2}ln(\frac{\overline{n}-1}{\overline{n}+1})) - \overline{b})$$

$$\equiv \overline{x} \bowtie (\overline{\gamma}.\frac{1}{2}ln(\frac{\overline{n}-1}{\overline{n}+1}) - \overline{b}); [\overline{\gamma} = (\overline{w}^T.\overline{w})^{-1}.(\overline{w}^T)]$$

1.5 Canonical Examples of Inferring Data Precondition from Derived *wp* rules

To infer data preconditions, starting from the last layer statement, we assert using the *wp* equation to determine the weakest precondition. The equation of *Dense* layer is as follows:

$$output = activation(dot(input, weight) + bias)$$

So, for given postcondition $\delta : y \leq n$, the statement S_3 can be written for output of last layer (y_3) with corresponding weight ($\overline{w_3}$) and bias($\overline{b_3}$) as,

$$y_3 = sigmoid(\overline{w_3^T}.\overline{x_3} + \overline{b_3})$$

Let us look at a canonical example of how we can apply the rules presented in Fig. 4.

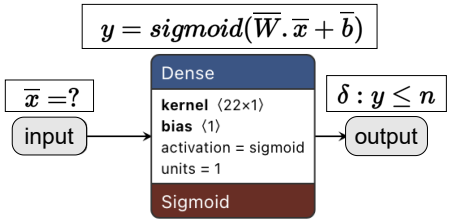

Figure 1: Example 1: Data precondition computation from given N with 1 layer and postcondition (δ)

1.5.1 ***Layer 1 Example:*** In Example 1 (Fig. 1), for given neural network (N) and postcondition (δ) as below,

$$N : a(f(\overline{x})) = sigmoid(\overline{W}.\overline{x} + \overline{b}); \delta : y \leq n$$

We can use *wp* rules over N and δ using (wpAlpha) rule to get the precondition for this single-layer neural network as follows,

$$wp(N, \delta) = wp(sigmoid(\overline{W}.\overline{x} + \overline{b}), \delta) \equiv \alpha(\delta, \beta(sigmoid(\overline{W}.\overline{x} + \overline{b})))$$

Now, we can apply (wpAlphaSigma) and (BetaSigmoid) rules consecutively to get the data precondition as follows,

$$\alpha(\delta, \beta(sigmoid(\overline{W}.\overline{x} + \overline{b}))) \equiv \beta(sigmoid(\overline{W}.\overline{x} + \overline{b}), \delta)$$

$$\equiv \beta(sigmoid(\overline{W}.\overline{x} + \overline{b}), y \leq n)$$

$$\equiv \overline{x} \leq (\overline{\gamma}.ln(\frac{n}{1-n})) - \overline{b}, \overline{\gamma} = (\overline{W^T}.\overline{W})^{-1}.(\overline{W^T})$$

Here, $\overline{x}$ is an array of data that has been fed into the Neural network, and the predicate of $\overline{x}$ denotes the precondition of the data for each feature that has been assumed by this 1-layer neural network.

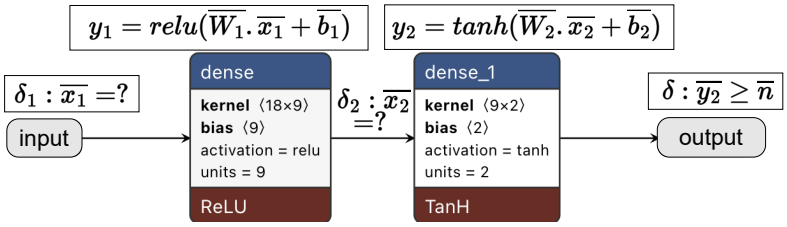

Figure 2: Example 2: Data precondition (δ_1) computation from given N with 2 layers and postcondition (δ)

1.5.2 ***Layer 2 Example:*** Let us consider, another model with different activation functions with 2 layers and multiple output classes (Example 2 in Fig. 2). Therefore, the output of this model ($\overline{y}$) is an array of n. Now, for this Example 2 Fig. 2, for given neural network (N) and postcondition (δ) as below,

$$N : a(f(\overline{x})).a(f(\overline{x})) = relu(\overline{W_1}.\overline{x_1} + \overline{b_1}).tanh(\overline{W_2}.\overline{x_2} + \overline{b_2}); \delta : \overline{y_2} \geq \overline{n}$$

We can use wp rules over N and δ using (WP), (WPALPHA) rule to get the precondition for this multiple layer neural network as follows,

$$\delta_1 = wp(N, \delta) = wp(N_0.N_1, \delta) \equiv$$
$$wp(relu(\overline{W_1}.\overline{x_1} + \overline{b_1}).tanh(\overline{W_2}.\overline{x_2} + \overline{b_2}), \delta)$$
$$\delta_1 = wp(relu(\overline{W_1}.\overline{x_1} + \overline{b_1}), \delta_2); \delta_2 = wp(tanh(\overline{W_2}.\overline{x_2} + \overline{b_2}), \delta)$$

Now, we can apply (WPALPHASIGMA) and (BETATANH) rules consecutively to get the data precondition as follows,

$$\delta_2 = wp(tanh(\overline{W_2}.\overline{x_2} + \overline{b_2}), \delta) \equiv \alpha(\delta, \beta(tanh(\overline{W_2}.\overline{x_2} + \overline{b_2})))$$
$$\equiv \beta(tanh(\overline{W_2}.\overline{x_2} + \overline{b_2}), \delta) \equiv \beta(tanh(\overline{W_2}.\overline{x_2} + \overline{b_2}), \overline{y_2} \geq \overline{n})$$
$$\equiv \overline{x_2} \leq (\overline{\gamma_2}.\frac{1}{2}ln(\frac{\overline{n}-1}{\overline{n}+1})) - \overline{b_2}, \overline{\gamma_2} = (\overline{W_2^T}.\overline{W_2})^{-1}.(\overline{W_2^T})$$

Here, $\overline{x_2}$ is an array of input that has been obtained from the first layer and fed into the second layer, and the predicate of $\overline{x_2}$ denotes the precondition of the data in layer 2, which is a postcondition of layer 1. After asserting with this postcondition, we obtain δ_1 similarly using (BETARELU) rule,

$$\delta_1 = wp(relu(\overline{W_1}.\overline{x_1} + \overline{b_1}), \overline{x_2} \leq (\overline{\gamma_2}.\frac{1}{2}ln(\frac{\overline{n}-1}{\overline{n}+1})) - \overline{b_2})$$
$$\equiv \overline{x_1} \leq (\overline{\gamma_1}.(\overline{\gamma_2}.\frac{1}{2}ln(\frac{\overline{n}-1}{\overline{n}+1})) - \overline{b_2}) - \overline{b_1}) \wedge \overline{x_1} \leq (\overline{\gamma_1}.(-\overline{b_1})),$$
$$\overline{\gamma_1} = (\overline{W_1^T}.\overline{W_1})^{-1}.(\overline{W_1^T})$$

In this step, we obtain the precondition which is an array of the data ($\overline{x_1}$) for each feature that has been assumed by this neural network with 2 layers.

1.5.3 ***Layer 3 Example:*** Let us consider an example with multiple `Dense` layers with `linear,sigmoid` activation functions. Now, for this Example 3 Fig. 3, for given neural network (N) and postcondition (δ) as below, We have `Dense` layers with `linear, linear, sigmoid` activation functions for this example. Now, for given neural network (N) and postcondition (δ),

$$N : linear(\overline{W_1}.\overline{x_1} + \overline{b_1}).linear(\overline{W_2}.\overline{x_2} + \overline{b_2}).sigmoid(\overline{W_3}.\overline{x_3} + \overline{b_3});$$
$$\delta : y_3 \geq n_1 \wedge y_3 \leq n_2$$

Our proposed technique is generalized to DNN models with multiple layers. For example, a DNN model presented in Fig. 3 has 3 layers and different activation functions. In that model, the output layer has a single class, i.e., the output value $y \in \mathbf{R}$. The given postcondition is an instance of ($\delta \wedge \delta$) and will be in the range between $[n_1, n_2]$. Now, We utilize wp rules over N and δ using (WP), (WPALPHA) rules to get the precondition for this multiple-layer neural network as follows,

$$\delta_1 = wp(N, \delta) = wp(N_0.N_1, \delta) \equiv$$
$$wp(linear(\overline{W_1}.\overline{x_1} + \overline{b_1}).linear(\overline{W_2}.\overline{x_2} + \overline{b_2}).sigmoid(\overline{W_3}.\overline{x_3} + \overline{b_3}), \delta)$$
$$\delta_1 = wp(linear(\overline{W_1}.\overline{x_1} + \overline{b_1}, \delta_2);$$
$$\delta_2 = wp(linear(\overline{W_2}.\overline{x_2} + \overline{b_2}).sigmoid(\overline{W_3}.\overline{x_3} + \overline{b_3}), \delta);$$
$$\delta_2 = wp(linear(\overline{W_2}.\overline{x_2} + \overline{b_2}), \delta_3); \delta_3 = wp(sigmoid(\overline{W_3}.\overline{x_3} + \overline{b_3}), \delta)$$

Then, we apply (WPALPHASIGMA), (WPALPHAWEDGE), (BETASIGMOID) rules consecutively to get the precondition as follows,

$$\delta_3 = wp(sigmoid(\overline{W_3}.\overline{x_3} + \overline{b_3}), \delta) \equiv \alpha(\delta_3, \beta(sigmoid(\overline{W_3}.\overline{x_3} + \overline{b_3})))$$

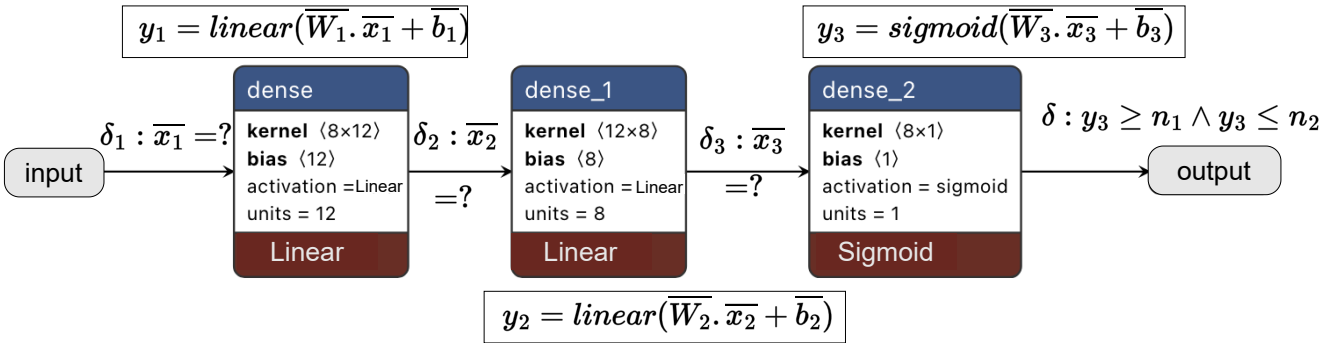

Figure 3: Data precondition (δ_1) computation from an example DNN model (N) with 3 layers and postcondition (δ)

$$\equiv \beta(sigmoid(\overline{W_3}.\overline{x_3} + \overline{b_3}), \delta_3) \equiv \beta(sigmoid(\overline{W_3}.\overline{x_3} + \overline{b_3}), y_3 \geq n_1 \wedge y_3 \leq n_2)$$
$$\equiv \overline{x_3} \geq ((\overline{\gamma_3}.ln\frac{n_1}{1-n_1}) - \overline{b_3}) \wedge \overline{x_3} \leq ((\overline{\gamma_3}.ln\frac{n_2}{1-n_2}) - \overline{b_3})$$

Here, $\overline{x_3}$ is an array of input that has been obtained from the second layer and fed into the third layer, and the predicate of $\overline{x_3}$ denotes the precondition of the data in layer 3, which is a postcondition of layer 2. Here, $\overline{\gamma_3}$ is an inverse function of the layer's weight matrix ($\overline{W_3}$). Then, we obtain δ_2 similarly using the wp rules (wpAlpha), (wpAlphaWedge), (BetaLinear) consecutively,

$$\delta_2 = wp(linear(\overline{W_2}.\overline{x_2} + \overline{b_2}), \delta_3)$$
$$\equiv \alpha(\delta_3, \beta(linear(\overline{W_2}.\overline{x_2} + \overline{b_2}))) \equiv \beta(linear(\overline{W_2}.\overline{x_2} + \overline{b_2}), \delta_3)$$
$$\equiv \beta(linear(\overline{W_2}.\overline{x_2} + \overline{b_2}), \overline{x_3} \geq ((\overline{\gamma_3}.ln\frac{n_1}{1-n_1}) - \overline{b_3}) \wedge$$
$$\overline{x_3} \leq ((\overline{\gamma_3}.ln\frac{n_2}{1-n_2}) - \overline{b_3})) \equiv \overline{x_2} \geq ((\overline{\gamma_2}.((\overline{\gamma_3}.ln\frac{n_1}{1-n_1}) - \overline{b_3})) - \overline{b_2}) \wedge$$
$$\overline{x_2} \leq ((\overline{\gamma_2}.((\overline{\gamma_3}.ln\frac{n_2}{1-n_2}) - \overline{b_3})) - \overline{b_2}), \overline{\gamma_2} = (\overline{W_2^T}.\overline{W_2})^{-1}.(\overline{W_2^T})$$

In this step, we obtain the precondition which is an array of the input ($\overline{x_2}$) that has been obtained from the first layer and fed into the second layer, and the predicate of $\overline{x_2}$ denotes the precondition of the input in layer 2, which is a postcondition of layer 1. After asserting with this postcondition, we obtain δ_1 similarly using the wp rules (wpAlpha), (wpAlphaWedge), (BetaLinear),

$$\delta_1 = wp(linear(\overline{W_1}.\overline{x_1} + \overline{b_1}), \delta_2) \equiv \alpha(\delta_2, \beta(linear(\overline{W_1}.\overline{x_1} + \overline{b_1})))$$
$$\equiv \beta(linear(\overline{W_1}.\overline{x_1} + \overline{b_1}), \delta_2)$$
$$\equiv \overline{x_1} \geq ((\overline{\gamma_1}.((\overline{\gamma_2}.((\overline{\gamma_3}.ln\frac{n_1}{1-n_1}) - \overline{b_3})) - \overline{b_2}) - \overline{b_1}) \wedge$$
$$\overline{x_1} \leq ((\overline{\gamma_1}.((\overline{\gamma_2}.((\overline{\gamma_3}.ln\frac{n_2}{1-n_2}) - \overline{b_3})) - \overline{b_2}) - \overline{b_1})$$

Finally, we obtain the precondition, which is an array of the data ($\overline{x_1}$) for each feature that this DNN has assumed with multiple layers where $\overline{\gamma_1} = (\overline{W_1^T}.\overline{W_1})^{-1}.(\overline{W_1^T})$. In our proposed technique, the entire process of data precondition inference from a DNN model is automated and generalized for other models, which is performed during the training stage.