A Appendix

A.1 Proof of Corollary 1

For a class of DNN stated in the Lemma 1, by fixing the selected pieces of all the piecewise-linear AFs, the input to each max-operator is represented by an affine function of an input image $\boldsymbol{x}$. Since max-operator is characterized by a set of inequalities, an max-operator in the DNN is written by a set of linear inequalities. Combining it with the proof of Lemma 1, $\tilde{\mathcal{Z}}^{\rm oc}$ is characterized by a set of linear inequalities.

A.2 Proof of Lemma 2

Given a real value z_t, we can always obtain $\Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\boldsymbol{x}(z_t) \leq \boldsymbol{\psi}^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}$ when applying the trained neural network on $\boldsymbol{x}(z_t)$. Then, for any $z \in \mathbb{R}$, if $\Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\boldsymbol{x}(z) \leq \boldsymbol{\psi}^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}$, $\mathcal{A}(z_t) = \mathcal{A}(z)$ and $\boldsymbol{s}(z_t) = \boldsymbol{s}(z)$. By decomposing $\boldsymbol{x}(z) = \boldsymbol{a} + \boldsymbol{b}z$, we have

$$\begin{aligned}\{\Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\boldsymbol{x}(z) \leq \boldsymbol{\psi}^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\} &= \{\Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}(\boldsymbol{a} + \boldsymbol{b}z) \leq \boldsymbol{\psi}^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\}\\ &= \{\Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\boldsymbol{b}z \leq \boldsymbol{\psi}^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))} - \Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\boldsymbol{a}\}\\ &= \{(\Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\boldsymbol{b})_k z \leq \psi_k^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))} - (\Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\boldsymbol{a})_k \ \text{ for all } k\}.\end{aligned}$$

Then, the breakpoint $z_{t+1} > z_t$ at which one node is gointg to change its status, i.e., the sign of one inequality is going to be changed, is computed as

$$z_{t+1} = \min_{k:(\Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\boldsymbol{b})_k > 0} \frac{\psi_k^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))} - (\Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\boldsymbol{a})_k}{(\Theta^{(\boldsymbol{s}(\boldsymbol{x}(z_t)))}\boldsymbol{b})_k}.$$

Hence, we have the result of Lemma 2.

A.3 The details of the algorithm.

Algorithm 2 shows the entire procedure to calculate the selective p-value in (5). First, we apply DNN on $\boldsymbol{x}^{\rm obs}$ to obtain the observed representation $\mathcal{A}(\boldsymbol{x}^{\rm obs})$. We next calculate the dimension of interest $\boldsymbol{\eta}$ which is used to identify a parametrized line in $\mathbb{R}^n$. Then, we compute $\mathcal{A}(\boldsymbol{x}(z))$ based on 3. When we have $\mathcal{A}(\boldsymbol{x}(z))$, we can easily obtain $\mathcal{Z}$ in (8) which is the key factor to compute selective p-value.

Algorithm 3 shows our solution to efficiently identify $\mathcal{A}(\boldsymbol{x}(z))$. In this algorithm, multiple *breakpoint* $z_1 < z_2 < ... < z_{|\mathcal{T}|}$ are computed one by one. Each breakpoint $z_t, t \in [|\mathcal{T}|]$, indicates a point at which the sign of one linear inequality is changed, i.e., the status of one node in the network is going to change from active to inactive or vice versa. By identifying all these breakpoints $\{z_t\}_{t \in [|\mathcal{T}|]}$, $\mathcal{A}(\boldsymbol{x}(z))$ is given by

$$\mathcal{A}(\boldsymbol{x}(z)) = \begin{cases} \mathcal{A}(\boldsymbol{x}(z_1)) & \text{if } z \in [z_1 = z_{\min}, z_2],\\ \mathcal{A}(\boldsymbol{x}(z_2)) & \text{if } z \in [z_2, z_3],\\ \quad\vdots & \\ \mathcal{A}(\boldsymbol{x}(z_{|\mathcal{T}|})) & \text{if } z \in [z_{|\mathcal{T}|}, z_{|\mathcal{T}|+1} = z_{\max}].\end{cases}$$

Choice of $[z_{\min}, z_{\max}]$. Under the normal distribution, very positive and very negative values of z does not affect the inference. Therefore, it is reasonable to consider range of values, e.g., $[-20\sigma, 20\sigma]$, where σ is the standard deviation of the sampling distribution of test statistic.

Algorithm 2 `parametric_SI`

```
Input: $x^{\text{obs}}$, $[z_{\min}, z_{\max}]$
1: Obtain $\mathcal{A}(x^{\text{obs}})$ by applying the pre-trained DNN to $x^{\text{obs}}$
2: Compute $\eta$, and then calculate $a$ and $b$ based on $x^{\text{obs}}$ and $\eta \leftarrow$ Equation (8)
3: $\mathcal{A}(x(z)) \leftarrow$ compute_solution_path$(a, b, [z_{\min}, z_{\max}])$
4: Identify truncation region $\mathcal{Z} \leftarrow \{z : \mathcal{A}(x(z)) = \mathcal{A}(x^{\text{obs}})\}$
5: $p_{\text{selective}} \leftarrow$ Equation (10)
Output: $p_{\text{selective}}$
```

Algorithm 3 `compute_solution_path`

```
Input: $a, b, [z_{\min}, z_{\max}]$
1: Initialization: $t = 1$, $z_t = z_{\min}$, $\mathcal{T} = z_t$
2: while $z_t < z_{\max}$ do
3:     Compute the corresponding data $x(z_t) = a + b z_t$ in $\mathbb{R}^n$ of $z_t$
4:     Obtain $\mathcal{A}(x(z_t))$ by applying a pre-trained DNN to $x(z_t)$
5:     Compute the next breakpoint $z_{t+1} \leftarrow$ Equation (13).
6:     $\mathcal{T} = \mathcal{T} \cup \{z_{t+1}\}$, and $t = t + 1$
7: end while
Output: $\{\mathcal{A}(x(z_t))\}_{z_t \in \mathcal{T}}$
```

A.4 Distribution of naive p-value and selective p-value when the null hypothesis is true

We demonstrate the validity of our proposed method by confirming the uniformity of p-value when the null hypothesis is true. We generated 12,000 null images $\boldsymbol{x} = (x_1, ..., x_n)$ in which $x_i \in [n] \sim \mathbb{N}(0, 1)$ for each case $n \in \{16, 64, 256, 1024\}$ and performed the experiments to check the distribution of naive p-values and selective p-values. From Figure 8, it is obvious that naive p-value does not follow uniform distribution. Therefore, it fails to control the false positive rate. The empirical distributions of selective p-value are shown in Figure 9. The results indicate our proposed method successfully control the false detection probability.

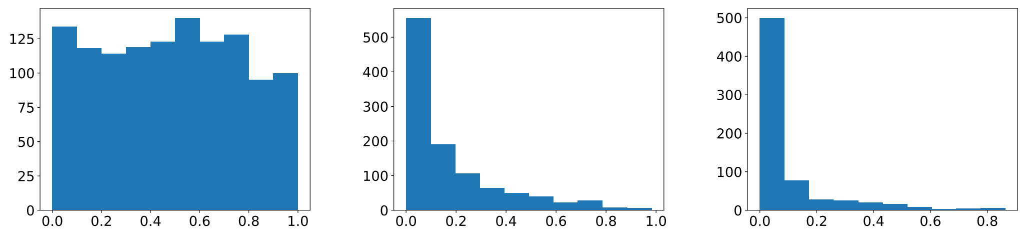
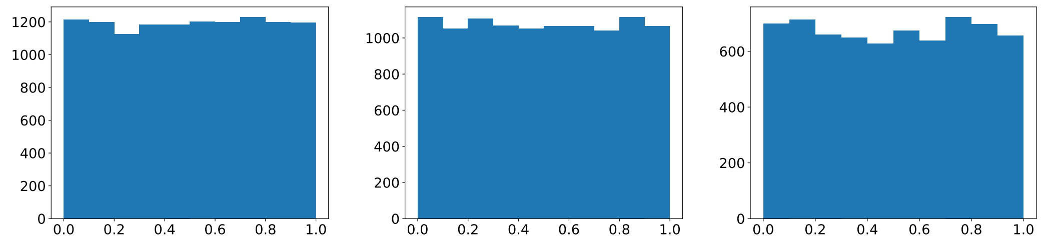

(a) $n = 16$ (b) $n = 64$ (c) $n = 256$

Figure 8: Distribution of naive p-value when the null hypothesis is true.

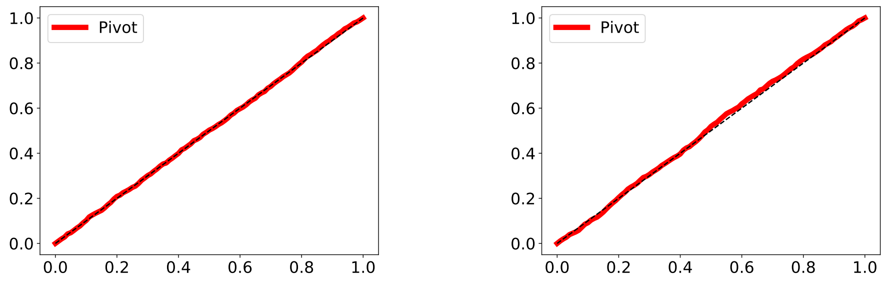
A.5 Examples of piecewise linear linear approximation for non-piecewise linear activation functions.

For simplicity, we consider a simple 3-layer neural network where the activation function at hidden layer is either sigmoid or tanh, the number of input nodes and output nodes are 8, and the number of hidden nodes is 16. Since these functions are non-piecewise linear functions, then we can use the



(a) $n = 16$ (b) $n = 64$ (c) $n = 256$

Figure 9: Distribution of selective p-value when the null hypothesis is true.



(a) Result for the piecewise linear approximation of sigmoid function.

(b) Result for the piecewise linear approximation tanh function.

Figure 10: Uniform QQ-plot of the pivot.

following approximation

$$f(x) = \text{sigmoid}(x) = \begin{cases} 0 & \text{if } x < -4, \\ \frac{1}{8}x + \frac{1}{2} & \text{if } -4 \leq x \leq 4, \\ 1 & \text{if } x > 4. \end{cases}$$

$$f(x) = \tanh(x) = \begin{cases} -1 & \text{if } x < -2, \\ \frac{1}{2}x & \text{if } -2 \leq x \leq 2, \\ 1 & \text{if } x > 2. \end{cases}$$

With the above approximations, we demonstrate that our method still can control the FPR by showing the uniform QQ-plot of the pivot, which is the p-value under the null hypothesis, in Figure 10.

In the above example, we used 3 cuts (pieces) to approximate the function. When we increase # cuts, the number of encounted intervals tend to exponentially increase. In Figure 11, we show that # encounted intervals still linearly increase in practise.

A.6 Details for experimental setup and additional results

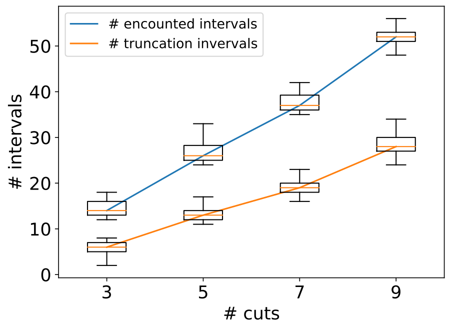
Methods for demonstration. We compare our *proposed-method* (homotopy method), which is mainly presented in the paper, with the following approaches:

• *proposed-method-oc (over-conditioning).* This is our first idea of characterizing the conditional data space by conditioning on the observed activeness and inactivess of all the nodes. The limitation of this method is low statistical because of over-conditioning.

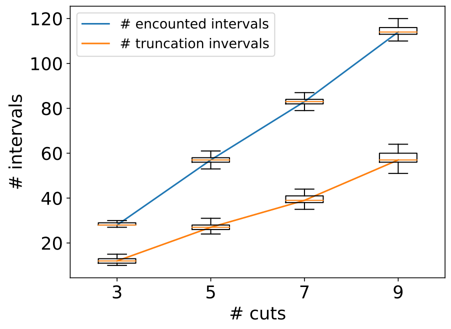
• *naive method.* This method use classical z-test to calculate the naive p-value in Equation (4).

• *permutation test.* This is also a well-known technique to compute the p-value. The permutation testing procedure is described as follows:

– Compute the observed test statistic T^{obs} by applying DNN on $\boldsymbol{x}^{\text{obs}}$



(a) # hidden node = 8



(b) # hidden node = 8

Figure 11: Demonstration of # encounted and # truncation intervals when increasing # cuts (pieces).

- For $1 \leftarrow b$ to B (B is the number of permutations which is given by user)
  + $\boldsymbol{x}^{(b)} \leftarrow \text{permute}(\boldsymbol{x}^{\text{obs}})$
  + Compute $T^{(b)}$ by applying DNN on $\boldsymbol{x}^{(b)}$
- Compute p-value as follows

$$p = \frac{1}{B} \sum_{b=1}^{B} \mathbb{1}\{|T^{\text{obs}}| \leq |T^{(b)}|\}.$$

Experimental setup. The point of SI that we only conduct statistical testing when there is at least one hypothesis discoverd by the DNNs. Therefore, the power is defined as follows:

$$\text{Power} = \frac{\#\ \text{detected}\ \&\ \text{rejected}}{\#\ \text{detected}}.$$

This definition of power is the same as Hyun et al. (2018) and Duy et al. (2020) where $\#$ detected is the number of images on which at least one hypothesis is discovered by DNN and $\#$ rejected is the number of images whose null hypothesis is rejected by our proposed method.

For the numerical experiments, we conducted the inference on DNN-driven hypothesis based on the output representation, where the pre-trained CNN has the following structure:

Layer	Output Size
Input	$\sqrt{n} \times \sqrt{n} \times 1$
Conv (3, 3, 1, 4)[a]	$\sqrt{n} \times \sqrt{n} \times 4$
Max pooling	$\sqrt{n}/2 \times \sqrt{n}/2 \times 4$
Upsampling	$\sqrt{n} \times \sqrt{n} \times 4$
Conv (3, 3, 1, 1)	$\sqrt{n} \times \sqrt{n} \times 1$
Output	$\sqrt{n} \times \sqrt{n} \times 1$

[a] Conv([filter height], [filter width], [# channel], [# filter]). The label of each pixel (object or background) is provided during the training process.

For some experimental results on real-world brain image dataset, where we conducted the inference on DNN-driven hypothesis based on the internal representation, the used pre-trained network has the following structure:

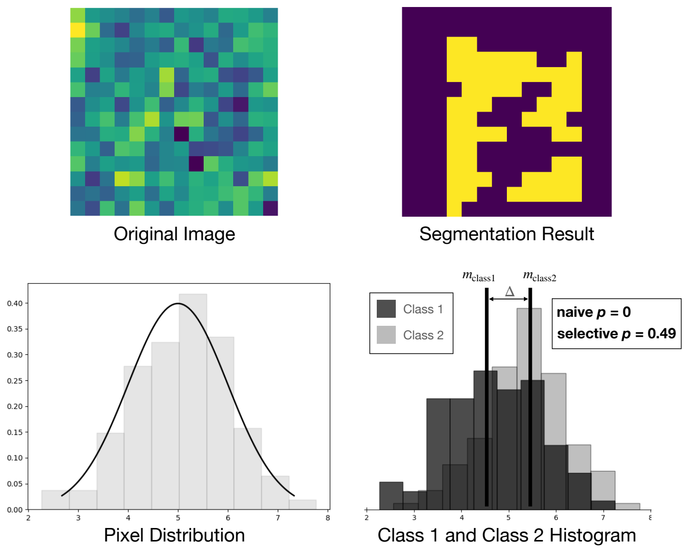


Figure 12: Demonstration of bias correction by selective inference

Layer	Output Size
Input	$\sqrt{n} \times \sqrt{n} \times 1$
Conv (3, 3, 1, 4)	$\sqrt{n} \times \sqrt{n} \times 4$
Max pooling	$\sqrt{n}/2 \times \sqrt{n}/2 \times 4$
Conv (3, 3, 1, 4)	$\sqrt{n}/2 \times \sqrt{n}/2 \times 4$
Upsampling	$\sqrt{n} \times \sqrt{n} \times 4$
Global Average Pooling	4
Output	1

For this experiment, the label of each image (tumor or no tumor) is provided for the training process.

We executed the code on Intel(R) Xeon(R) CPU E5-2687W v4 @ 3.00GHz.

Experimental results. We provide several additional results as follows:

• **Demonstration of bias correction.** Figure 12 shows the importance of SI for avoiding the selection bias. We generated a 14×14 null image in which all the pixel values are the same. Unfortunatelly, due to the noise, some pixels with high value are selected by DNN. If we use naive p-value to quantify the reliability, we reject the null hypothesis, which is incorrect. By applying our proposed method based on the concept of SI, we correctly do not reject the null hypothesis.

• **The robustness of the proposed method in terms of the FPR control.** We considered the data following Laplace distribution, skew normal distribution (skewness coeficient is 10) and t_{20} distribution. We also conducted experiments when σ^2 is also estimated from the data. In each experiment, we generated 1,200 null sequences for $n \in \{16, 64, 256, 1024\}$. We tested the FPR for both $\alpha = 0.05$ and $\alpha = 0.1$. The FPR results are shown in Figure 13.

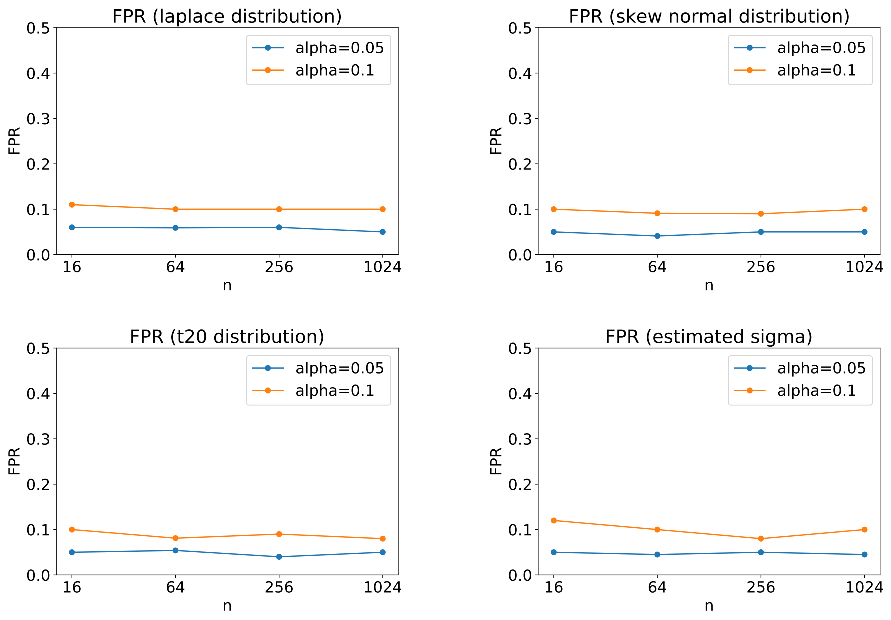


Figure 13: False positive rate of the proposed method when data is non-normal or σ^2 is unknown.