Appendix

A. BayesNAS Algorithm Derivation

A.1. Algorithm Derivation

In this subsection, we explain the detailed algorithm of updating hyper-parameters for the abstracted single motif as shown in Figure 1e. The proposition about optimization objective will be illustrated firstly.

Proposition 2 *Suppose the likelihood of the architecture parameters of a neural network* $\mathbf{w}$ *could be formulated as one exponential family distribution* $p(\mathbf{Y} \mid \mathbf{w}, \mathbf{X}, \mathbf{s}) \sim \exp\left(-E_D(\mathbf{Y}; \mathit{Net}(\mathbf{X}; \mathbf{w}); \mathbf{s})\right)$*, where* $\mathcal{D} = (\mathbf{X}, \mathbf{Y})$ *is the given dataset,* $\mathbf{s}$ *stands for the uncertainty and* $E_D(*)$ *represents the* energy function *over data. The sparse prior with super Gaussian distribution for each architecture parameter has been defined in equation 11. The unknown architecture parameter of the network* $\mathbf{w}$ *and hyperparameter* $\mathbf{s}$ *can be approximately obtained by solving the following optimization problem*

$$\min_{\mathbf{w},\mathbf{s}} \mathcal{L}(\mathbf{w}, \mathbf{s}) \tag{A.1.1}$$

specially, for the architecture parameter $\mathbf{w}_{\mathbf{jk}}^{\mathbf{o}'}$ *which is associated with one operation of the edge* e_{jk} $(j < k)$*, the optimization problem could be reformulated as:*

$$\begin{aligned}\mathcal{L}(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}) &= \mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'\,*})\mathbf{w}_{jk}^{o'} + 2\mathbf{w}_{jk}^{o'}\left[\mathbf{g}(\mathbf{w}_{jk}^{o'\,*}) - \mathbf{H}(\mathbf{w}_{jk}^{o'\,*})\mathbf{w}_{jk}^{o'\,*}\right] + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'\,-1}\mathbf{w}_{jk}^{o'} \\ &+ \log|\mathbf{s}_{jk}^{o'}| + \log|\mathbf{H}(\mathbf{w}_{jk}^{o'\,*}) + \mathbf{s}_{jk}^{o'\,-1}| - 2\log b(\mathbf{w}_{jk}^{o'\,*})\end{aligned} \tag{A.1.2}$$

where $\mathbf{w}_{jk}^{o'\,*}$ *is arbitrary, and*

$$\mathbf{g}(\mathbf{w}_{jk}^{o'\,*}) \triangleq \nabla E_D(\mathbf{w}_{jk}^{o'})|_{\mathbf{w}_{jk}^{o'\,*}}, \mathbf{H}(\mathbf{w}_{jk}^{o'\,*}) \triangleq \nabla\nabla E_D(\mathbf{w}_{jk}^{o'})|_{\mathbf{w}_{jk}^{o'\,*}}$$

and

$$b(\mathbf{w}_{jk}^{o'\,*}) \triangleq \exp\left\{-\left(\frac{1}{2}\mathbf{w}_{jk}^{o'\,*}\mathbf{H}(\mathbf{w}_{jk}^{o'\,*})\mathbf{w}_{jk}^{o'\,*} - \mathbf{w}_{jk}^{o'\,*}\mathbf{g}(\mathbf{w}_{jk}^{o'\,*}) + E_D(\mathbf{w}_{jk}^{o'\,*})\right)\right\}$$

It should also be noted that $\mathbf{s}_{jk}^{o'}$ represents the uncertainty of $\mathbf{w}_{jk}^{o'}$ without considering the dependency between edge $e_{jk}^{o'}$ and $\sum_{i<j} e_{ij}^{o}$, where o' and o stands for one possible operation in corresponding edges.

Proof *Given the likelihood with exponential family distribution*

$$p(\mathbf{Y} \mid \mathbf{w}_{jk}^{o'}, \mathbf{X}, \mathbf{s}_{jk}^{o'}) \sim \exp\left(-E_D(\mathbf{Y}; \mathit{Net}(\mathbf{X}; \mathbf{w}_{jk}^{o'}); \mathbf{s}_{jk}^{o'})\right)$$

as explained in equation 5, we define the prior of $w_{jk}^{o'}$ *with Gaussian distribution*

$$p(w_{jk}^{o'}) = \mathcal{N}(w_{jk}^{o'}|0, \mathbf{s}_{jk}^{o'})$$

The marginal likelihood could be calculated as:

$$p(\mathbf{Y} \mid \mathbf{w}_{jk}^{o'})\mathcal{N}(\mathbf{w}_{jk}^{o'}|\mathbf{0}, \mathbf{s}_{jk}^{o'})d\mathbf{w}_{jk}^{o'} = \int \exp\{-E_D(\mathbf{w}_{jk}^{o'})\}\mathcal{N}(\mathbf{w}_{jk}^{o'}|\mathbf{0}, \mathbf{s}_{jk}^{o'})d\mathbf{w}_{jk}^{o'} \tag{A.1.3}$$

Typically, this integral is intractable or has no analytical solution.

The mean and covariance can be fixed if the family is Gaussian. Performing a Taylor series expansion around some point $\mathbf{w}_{jk}^{o'\,*}$*,* $E_D(\mathbf{w}_{jk}^{o'})$ *can be approximated as*

$$E_D(\mathbf{w}_{jk}^{o'}) \approx E_D(\mathbf{w}_{jk}^{o'\,*}) + (\mathbf{w}_{jk}^{o'} - \mathbf{w}_{jk}^{o'\,*})\mathbf{g}(\mathbf{w}_{jk}^{o'\,*}) + \frac{1}{2}(\mathbf{w}_{jk}^{o'} - \mathbf{w}_{jk}^{o'\,*})\mathbf{H}(\mathbf{w}_{jk}^{o'\,*})(\mathbf{w}_{jk}^{o'} - \mathbf{w}_{jk}^{o'\,*}) \tag{A.1.4}$$

where $\mathbf{g}(\cdot)$ is the gradient and $\mathbf{H}(\cdot)$ is the Hessian of the energy function E

$$\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) \triangleq \nabla E_D(\mathbf{w}_{jk}^{o'})|_{\mathbf{w}_{jk}^{o'^*}} \tag{A.1.5a}$$

$$\mathbf{H}(\mathbf{w}_{jk}^{o'^*}) \triangleq \nabla\nabla E_D(\mathbf{w}_{jk}^{o'})|_{\mathbf{w}_{jk}^{o'^*}} \tag{A.1.5b}$$

To derive the cost function in equation A.1.2, we introduce the posterior mean and covariance:

$$\mathbf{m}_{jk}^{o'} = C_{jk}^{o'} \cdot \left[\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) + \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'^*}\right], \tag{A.1.6a}$$

$$C_{jk}^{o'} = \left[\mathbf{s}_{jk}^{o'^{-1}} + \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\right]^{-1}. \tag{A.1.6b}$$

Then define the following quantities

$$b(\mathbf{w}_{jk}^{o'^*}) \triangleq \exp\left\{-\left(\frac{1}{2}\mathbf{w}_{jk}^{o'^*}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'^*} - \mathbf{w}_{jk}^{o'^*}\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) + E_D(\mathbf{w}_{jk}^{o'^*})\right)\right\}, \tag{A.1.7a}$$

$$c(\mathbf{w}_{jk}^{o'^*}) \triangleq \exp\left\{\frac{1}{2}\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*})\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*})\right\}, \tag{A.1.7b}$$

$$d(\mathbf{w}_{jk}^{o'^*}) \triangleq \sqrt{|\mathbf{H}(\mathbf{w}_{jk}^{o'^*})|}, \tag{A.1.7c}$$

$$\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*}) \triangleq \mathbf{g}(\mathbf{w}_{jk}^{o'^*}) - \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'^*}. \tag{A.1.7d}$$

Now the approximated likelihood $p(\mathbf{Y}|\mathbf{w}_{jk}^{o'})$ is a exponential of quadratic, then Gaussian,

$$\begin{aligned}
&p(\mathbf{Y}|\mathbf{w}_{jk}^{o'}) \\
=&\exp\{-E_D(\mathbf{w}_{jk}^{o'})\} \\
\approx&\exp\left\{-\left(\frac{1}{2}(\boldsymbol{W}-\boldsymbol{W}^*)^\top\mathbf{H}(\boldsymbol{W}^*)(\boldsymbol{W}-\boldsymbol{W}^*) + (\boldsymbol{W}-\boldsymbol{W}^*)^\top\mathbf{g}(\boldsymbol{W}^*) + E_D(\mathbf{w}_{jk}^{o'^*})\right)\right\} \\
=&\exp\left\{-\left(\frac{1}{2}\mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + \mathbf{w}_{jk}^{o'}\left[\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) - \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'^*}\right]\right)\right\} \\
&\cdot\exp\left\{-\left(\frac{1}{2}\mathbf{w}_{jk}^{o'^*}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'^*} - \mathbf{w}_{jk}^{o'^*}\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) + E_D(\mathbf{w}_{jk}^{o'^*})\right)\right\} \\
=&b(\mathbf{w}_{jk}^{o'^*})\cdot\exp\left\{-\left(\frac{1}{2}\mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + \mathbf{w}_{jk}^{o'}\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*})\right)\right\} \\
&\cdot\exp\left\{\frac{1}{2}\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*})\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*}) - \frac{1}{2}\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*})\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*})\right\} \\
=&b(\mathbf{w}_{jk}^{o'^*})\cdot c(\mathbf{w}_{jk}^{o'^*}) \\
&\cdot\exp\left\{-\left(\frac{1}{2}\mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + \mathbf{w}_{jk}^{o'}\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*}) + \frac{1}{2}\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*})\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*})\right)\right\} \\
=&(2\pi)^{M/2}b(\mathbf{w}_{jk}^{o'^*})c(\mathbf{w}_{jk}^{o'^*})d(\mathbf{w}_{jk}^{o'^*})\cdot\mathcal{N}(\mathbf{w}_{jk}^{o'}|\hat{\mathbf{w}_{jk}^{o'^*}}, \mathbf{H}^{-1}(\mathbf{w}_{jk}^{o'^*})) \\
\triangleq&A(\mathbf{w}_{jk}^{o'^*})\cdot\mathcal{N}(\mathbf{w}_{jk}^{o'}|\hat{\mathbf{w}_{jk}^{o'^*}}, \mathbf{H}^{-1}(\mathbf{w}_{jk}^{o'^*})),
\end{aligned} \tag{A.1.8}$$

where

$$\begin{aligned}
A(\mathbf{w}_{jk}^{o'^*}) &= (2\pi)^{M/2}b(\mathbf{w}_{jk}^{o'^*})c(\mathbf{w}_{jk}^{o'^*})d(\mathbf{w}_{jk}^{o'^*}), \\
\hat{\mathbf{w}_{jk}^{o'^*}} &= -\mathbf{H}^{-1}(\mathbf{w}_{jk}^{o'^*})\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*}) = \mathbf{w}_{jk}^{o'^*} - \mathbf{H}^{-1}(\mathbf{w}_{jk}^{o'^*})\mathbf{g}(\mathbf{w}_{jk}^{o'^*}).
\end{aligned}$$

We can write the approximate marginal likelihood as

$$\begin{aligned}
&A(\mathbf{w}_{jk}^{o'^*}) \int \mathcal{N}(\mathbf{w}_{jk}^{o'}|\hat{\mathbf{w}_{jk}^{o'}}^*, \mathbf{H}^{-1}(\mathbf{w}_{jk}^{o'^*})) \cdot \mathcal{N}(\mathbf{w}_{jk}^{o'}|\mathbf{0}, \mathbf{s}_{jk}^{o'}) d\mathbf{w}_{jk}^{o'} \\
=&b(\mathbf{w}_{jk}^{o'^*}) \cdot \int \exp\left\{-\left(\frac{1}{2}\mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + \mathbf{w}_{jk}^{o'}\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*})\right)\right\} \mathcal{N}(\mathbf{w}_{jk}^{o'}|\mathbf{0}, \mathbf{s}_{jk}^{o'}) d\mathbf{w}_{jk}^{o'} \\
=&\frac{b(\mathbf{w}_{jk}^{o'^*})}{(2\pi)^{1/2}|\mathbf{s}_{jk}^{o'}|^{1/2}} \int \exp\{-\hat{E}(\mathbf{w}_{jk}^{o'})\} d\mathbf{w}_{jk}^{o'},
\end{aligned} \tag{A.1.9}$$

where

$$\hat{E}(\mathbf{w}_{jk}^{o'}) = \frac{1}{2}\mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + \mathbf{w}_{jk}^{o'}\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*}) + \frac{1}{2}\mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'^{-1}}\mathbf{w}_{jk}^{o'}. \tag{A.1.10}$$

Equivalently, we get

$$\hat{E}(\mathbf{w}_{jk}^{o'}) = \frac{1}{2}(\mathbf{w}_{jk}^{o'} - \mathbf{m}_{jk}^{o'})(C_{jk}^{o'})^{-1}(\mathbf{w}_{jk}^{o'} - \mathbf{m}_{jk}^{o'}) + \hat{E}(\mathbf{Y}), \tag{A.1.11}$$

where $\mathbf{m}_{jk}^{o'}$ and $C_{jk}^{o'}$ are given in equation A.1.6. From equation A.1.6a and equation A.1.6b, the data-dependent term can be re-expressed as

$$\begin{aligned}
\hat{E}(\mathbf{Y}) =&\frac{1}{2}\mathbf{m}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{m}_{jk}^{o'} + \mathbf{m}_{jk}^{o'}\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) + \frac{1}{2}\mathbf{m}_{jk}^{o'}\mathbf{s}_{jk}^{o'^{-1}}\mathbf{m}_{jk}^{o'} \\
=&\min_{\mathbf{w}_{jk}^{o'}}\left[\frac{1}{2}\mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + \mathbf{w}_{jk}^{o'}\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*}) + \frac{1}{2}\mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'^{-1}}\mathbf{w}_{jk}^{o'}\right] \\
=&\min_{\mathbf{w}_{jk}^{o'}}\left[\frac{1}{2}\mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + \mathbf{w}_{jk}^{o'}\left(\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) - \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'^*}\right) + \frac{1}{2}\mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'^{-1}}\mathbf{w}_{jk}^{o'}\right].
\end{aligned} \tag{A.1.12}$$

Using equation A.1.11, we can evaluate the integral in equation A.1.9 to obtain

$$\int \exp\left\{-\hat{E}(\mathbf{w}_{jk}^{o'})\right\} d\mathbf{w}_{jk}^{o'} = \exp\left\{-\hat{E}(\mathbf{Y})\right\} 2\pi|C_{jk}^{o'}|^{1/2}. \tag{A.1.13}$$

Applying a $-2\log(\cdot)$ transformation to equation A.1.9, we have

$$\begin{aligned}
&-2\log\left[\frac{b(\mathbf{w}_{jk}^{o'^*})}{(2\pi)^{1/2}|\mathbf{s}_{jk}^{o'}|^{1/2}} \int \exp\{-\hat{E}(\mathbf{w}_{jk}^{o'})\} d\mathbf{w}_{jk}^{o'}\right] \\
\propto&-2\log b(\mathbf{w}_{jk}^{o'^*}) + \hat{E}(\mathbf{Y}) + \\
&\log|\mathbf{s}_{jk}^{o'}| + \log|\mathbf{H}(\mathbf{w}_{jk}^{o'^*}) + \mathbf{s}_{jk}^{o'^{-1}}| \\
\propto&\mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + 2\mathbf{w}_{jk}^{o'}\hat{\mathbf{g}}(\mathbf{w}_{jk}^{o'^*}) + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'^{-1}}\mathbf{w}_{jk}^{o'} \\
&+\log|\mathbf{s}_{jk}^{o'}| + \log|\mathbf{H}(\mathbf{w}_{jk}^{o'^*}) + \mathbf{s}_{jk}^{o'^{-1}}| - 2\log b(\mathbf{w}_{jk}^{o'^*}).
\end{aligned} \tag{A.1.14}$$

Therefore we get the following cost function to be minimised in equation A.1.2 over $\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}$,

$$\begin{aligned}
&\mathcal{L}(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}) = \mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + 2\mathbf{w}_{jk}^{o'}\left[\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) - \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'^*}\right] + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'^{-1}}\mathbf{w}_{jk}^{o'} \\
&+\log|\mathbf{s}_{jk}^{o'}| + \log|\mathbf{H}(\mathbf{w}_{jk}^{o'^*}) + \mathbf{s}_{jk}^{o'^{-1}}| - 2\log b(\mathbf{w}_{jk}^{o'^*}).
\end{aligned}$$

It can be easily found that the first line of $\mathcal{L}$ is quadratic programming with ℓ_2 regularizer. The second line is all about the hyperparameter $\mathbf{s}_{jk}^{o'}$.

Once the estimation on $\mathbf{w}_{jk}^{o'}$ and $\mathbf{s}_{jk}^{o'}$ are obtained, the cost function is alternatively optimised. The new estimated $\mathbf{w}_{jk}^{o'}$ can substitute $\mathbf{w}_{jk}^{o'^*}$ and repeat the estimation iteratively.

■

We note that in equation A.1.5, $\mathbf{w}_{jk}^{o'}{}^{*}$ may not be the mode (i.e., the lowest energy state), which means the gradient term $\mathbf{g}$ may not be zero. Therefore the selection of $\mathbf{w}_{jk}^{o'}(1)^{*}$ remains to be problematic. We give the following Corollary to address this issue, which is more general.

Corollary 1 *Suppose*

$$\mathbf{w}_{jk}^{o'}{}^{*} = \arg\min_{\mathbf{w}_{jk}^{o'}} E_D(\mathbf{w}_{jk}^{o'}) + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'}{}^{-1}\mathbf{w}_{jk}^{o'}, \tag{A.1.15}$$

we define a new cost function

$$\hat{\mathcal{L}}(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}) \triangleq E_D(\mathbf{w}_{jk}^{o'}) + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'}{}^{-1}\mathbf{w}_{jk}^{o'} + \log|\mathbf{s}_{jk}^{o'}| + \log|\mathbf{H}(\mathbf{w}_{jk}^{o'}{}^{*}) + \mathbf{s}_{jk}^{o'}{}^{-1}| - 2\log b(\mathbf{w}_{jk}^{o'}{}^{*}). \tag{A.1.16}$$

Instead of minimising $\mathcal{L}(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'})$, we can solve the following optimization problem to get $\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}$,

$$\min_{\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'},} \hat{\mathcal{L}}(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}).$$

Proof *Since the likelihood is*

$$p(\mathbf{Y}|\mathbf{w}_{jk}^{o'}) = \exp\{-E_D(\mathbf{w}_{jk}^{o'})\},$$

then

$$\min_{\mathbf{w}_{jk}^{o'}} E_D(\mathbf{w}_{jk}^{o'}) + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'}{}^{-1}\mathbf{w}_{jk}^{o'}$$

is exactly the regularised maximum likelihood estimation *with ℓ_2 type regulariser.*

We look at the first part of $\mathcal{L}(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'})$ in equation A.1.2, and define them as

$$\mathcal{L}_0(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}) \triangleq \mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'}{}^{*})\mathbf{w}_{jk}^{o'} + 2\mathbf{w}_{jk}^{o'}\left[\mathbf{g}(\mathbf{w}_{jk}^{o'}{}^{*}) - \mathbf{H}(\mathbf{w}_{jk}^{o'}{}^{*})\mathbf{w}_{jk}^{o'}{}^{*}\right] + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'}{}^{-1}\mathbf{w}_{jk}^{o'},$$

then

$$\begin{aligned}
&\min_{\mathbf{w}_{jk}^{o'}} \mathcal{L}_0(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}) \\
=&\min_{\mathbf{w}_{jk}^{o'}} \frac{1}{2}(\boldsymbol{W} - \boldsymbol{W}^*)^\top \mathbf{H}(\boldsymbol{W}^*)(\boldsymbol{W} - \boldsymbol{W}^*) + (\boldsymbol{W} - \boldsymbol{W}^*)^\top \mathbf{g}(\boldsymbol{W}^*) + E_D(\mathbf{w}_{jk}^{o'}{}^{*}) + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'}{}^{-1}\mathbf{w}_{jk}^{o'} \\
\approx&\min_{\mathbf{w}_{jk}^{o'}} E_D(\mathbf{w}_{jk}^{o'}) + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'}{}^{-1}\mathbf{w}_{jk}^{o'}
\end{aligned} \tag{A.1.17}$$

where, given equation A.1.5,

$$\begin{aligned}
\mathbf{g}(\mathbf{w}_{jk}^{o'}) &= \nabla E_D(\mathbf{w}_{jk}^{o'}) \\
\mathbf{H}(\mathbf{w}_{jk}^{o'}) &= \nabla\nabla E_D(\mathbf{w}_{jk}^{o'}).
\end{aligned} \tag{A.1.18}$$

Such quadratic approximation to $E_D(\mathbf{w}_{jk}^{o'}) + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'}{}^{-1}\mathbf{w}_{jk}^{o'}$ is actually the same as the approximation procedure in Trust-Region Methods where a region is defined around the current iterate within which they trust the model to be an adequate representation of the objective function (Nocedal & Wright, 2006, pp.65).

To obtain each step, we seek a solution of the subproblem at iteration t

$$\begin{aligned}
&\min_{\mathbf{w}_{jk}^{o'}} E_D(\mathbf{w}_{jk}^{o'}(t-1)) + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'}(t-1)^{-1}\mathbf{w}_{jk}^{o'} \\
=&\min_{\mathbf{w}_{jk}^{o'}} \frac{1}{2}(\mathbf{w}_{jk}^{o'} - \mathbf{w}_{jk}^{o'}(t-1))\mathbf{H}(\mathbf{w}_{jk}^{o'}(t-1))(\mathbf{w}_{jk}^{o'} - \mathbf{w}_{jk}^{o'}(t-1)) + (\mathbf{w}_{jk}^{o'} - \mathbf{w}_{jk}^{o'}(t-1))\mathbf{g}(\mathbf{w}_{jk}^{o'}(t-1)) \\
&+ \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'}(t-1)^{-1}\mathbf{w}_{jk}^{o'}
\end{aligned} \tag{A.1.19}$$

Suppose

$$\mathbf{w}_{jk}^{o'}{}^{*} = \arg\min_{\mathbf{w}_{jk}^{o'}} E_D(\mathbf{w}_{jk}^{o'}) + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'}{}^{-1}\mathbf{w}_{jk}^{o'},$$

then inject $\mathbf{w}_{jk}^{o'^*}$ *into* $\min_{\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}} \mathcal{L}(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'})$*, we can optimise equation A.1.16 instead of equation A.1.2, i.e.,* $\min_{\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}} \hat{\mathcal{L}}(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'})$.

■

A.2. Algorithm for Proxyless Tasks

In this Section, we propose iterative optimization algorithms to estimate $\mathbf{w}_{jk}^{o'}$ and $\mathbf{s}_{jk}^{o'}$ alternatively.

A.2.1. Optimization for architecture parameter $\mathbf{w}_{jk}^{o'}$ and switch $\mathbf{s}_{jk}^{o'}$

We first target for the estimation of unknown parameter $\mathbf{w}_{jk}^{o'}$ and hyperparameter $\mathbf{s}_{jk}^{o'}$. In the sequel, we show that the stated program can be formulated as a *convex-concave procedure* (CCCP) for $\mathbf{w}_{jk}^{o'}$ and $\mathbf{s}_{jk}^{o'}$.

Proposition 3 *The following programme*

$$\min_{\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}} \mathcal{L}(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'})$$

with the cost function defined as

$$\begin{aligned}\mathcal{L}(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}) \triangleq\ & \mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + 2\mathbf{w}_{jk}^{o'}\left[\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) - \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'^*}\right] + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'^{-1}}\mathbf{w}_{jk}^{o'} \\ & + \log|\mathbf{s}_{jk}^{o'}| + \log|\mathbf{s}_{jk}^{o'^{-1}} + \mathbf{H}(\mathbf{w}_{jk}^{o'^*})|\end{aligned} \tag{A.2.1}$$

can be formulated as a convex-concave procedure (CCCP), where $\mathbf{w}_{jk}^{o'^*}$ *can be arbitrary real vector.*

Proof Fact on convexity: *the function*

$$\begin{aligned}u\left(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}\right) =& \mathbf{w}_{jk}^{o'}\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'} + 2\mathbf{w}_{jk}^{o'}\left[\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) - \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'^*}\right] + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'^{-1}}\mathbf{w}_{jk}^{o'} \\ \propto& (\mathbf{w}_{jk}^{o'} - \mathbf{w}_{jk}^{o'^*})\mathbf{H}(\mathbf{w}_{jk}^{o'^*})(\mathbf{w}_{jk}^{o'} - \mathbf{w}_{jk}^{o'^*}) + 2\mathbf{w}_{jk}^{o'}\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) + \mathbf{w}_{jk}^{o'}\mathbf{s}_{jk}^{o'^{-1}}\mathbf{w}_{jk}^{o'}\end{aligned} \tag{A.2.2}$$

is convex jointly in $\mathbf{w}_{jk}^{o'}$, $\mathbf{s}_{jk}^{o'}$ *due to the fact that* $f(\mathbf{x}, Y) = \mathbf{x}\mathbf{Y}^{-1}\mathbf{x}$ *is jointly convex in* $\mathbf{x}$, $\mathbf{Y}$ *(see, (Boyd & Vandenberghe, 2004, p.76)). Hence* u *as a sum of convex functions is convex.*

Fact on concavity: *the function*

$$v(\mathbf{s}_{jk}^{o'}) = \log|\mathbf{s}_{jk}^{o'}| + \log|\mathbf{s}_{jk}^{o'^{-1}} + \mathbf{H}(\mathbf{w}_{jk}^{o'^*})| \tag{A.2.3}$$

is jointly concave in $\mathbf{s}_{jk}^{o'}$, $\mathbf{\Pi}$*. We exploit the properties of the determinant of a matrix*

$$|A_{22}||A_{11} - A_{12}A_{22}^{-1}A_{21}| = \left|\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22}\end{pmatrix}\right| = |A_{11}||A_{22} - A_{21}A_{11}^{-1}A_{12}|.$$

Then we have

$$\begin{aligned}v(\mathbf{s}_{jk}^{o'}) &= \log|\mathbf{s}_{jk}^{o'}| + \log|\mathbf{s}_{jk}^{o'^{-1}} + \mathbf{H}(\mathbf{w}_{jk}^{o'^*})| \\ &= \log\left(|\mathbf{s}_{jk}^{o'}||\mathbf{s}_{jk}^{o'^{-1}} + \mathbf{H}(\mathbf{w}_{jk}^{o'^*})|\right) \\ &= \log\left|\begin{pmatrix}\mathbf{H}(\mathbf{w}_{jk}^{o'^*}) & \\ & -\mathbf{s}_{jk}^{o'}\end{pmatrix}\right| \\ &= \log\left|\mathbf{s}_{jk}^{o'} + \mathbf{H}^{-1}(\mathbf{w}_{jk}^{o'^*})\right| + \log\left|\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\right|\end{aligned} \tag{A.2.4}$$

which is a log*-determinant of an affine function of semidefinite matrices* $\mathbf{\Pi}$, $\mathbf{s}_{jk}^{o'}$ *and hence concave.*

Therefore, we can derive the iterative algorithm solving the CCCP. We have the following iterative convex optimization program by calculating the gradient of concave part.

$$\mathbf{w}_{jk}^{o'}(t) = \arg\min_{\mathbf{w}_{jk}^{o'}} u(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}(t-1), \mathbf{H}(\mathbf{w}_{jk}^{o'^*})), \tag{A.2.5}$$

$$\mathbf{s}_{jk}^{o'}(t) = \arg\min_{\mathbf{s}_{jk}^{o'} \succeq \mathbf{0}} u(\mathbf{w}_{jk}^{o'}, \mathbf{s}_{jk}^{o'}(t-1), \mathbf{H}(\mathbf{w}_{jk}^{o'^*})) + \nabla_{\mathbf{s}_{jk}^{o'}} v(\mathbf{s}_{jk}^{o'}(t-1), \mathbf{H}(\mathbf{w}_{jk}^{o'^*}))\mathbf{s}_{jk}^{o'}(t-1). \tag{A.2.6}$$

■

A.2.2. Derivation of iterative reweighted ℓ_1 algorithm

Using basic principles in convex analysis, we then obtain the following analytic form for the negative gradient of $v(\mathbf{s}_{jk}^{o'})$ at $\mathbf{s}_{jk}^{o'}$ is (using chain rule):

$$\begin{aligned}
&\nabla_{\mathbf{s}_{jk}^{o'}} v\left(\mathbf{s}_{jk}^{o'}, \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\right)|_{\mathbf{s}_{jk}^{o'}=\mathbf{s}_{jk}^{o'}(t-1)} \\
=&\nabla_{\mathbf{s}_{jk}^{o'}} \left(\log|\mathbf{s}_{jk}^{o'^{-1}} + \mathbf{H}(\mathbf{w}_{jk}^{o'^*})| + \log|\mathbf{s}_{jk}^{o'}|\right)|_{\mathbf{s}_{jk}^{o'}=\mathbf{s}_{jk}^{o'}(t-1)} \\
=&-\frac{((\mathbf{s}_{jk}^{o'}(t-1))^{-1} + \mathbf{H}(\mathbf{w}_{jk}^{o'^*}))^{-1}}{(\mathbf{s}_{jk}^{o'}(t-1))^2} + \frac{1}{\mathbf{s}_{jk}^{o'}(t-1)}
\end{aligned} \tag{A.2.7}$$

Combined with equation A.1.6b, we denote a new hyper-parameter $w_{ij}^{o'}$ as following:

$$\begin{aligned}
\omega_{jk}^{o'}(t) =&\sqrt{-\frac{((\mathbf{s}_{jk}^{o'}(t-1))^{-1} + \mathbf{H}(\mathbf{w}_{jk}^{o'}(t)))^{-1}}{(\mathbf{s}_{jk}^{o'}(t-1))^2} + \frac{1}{\mathbf{s}_{jk}^{o'}(t-1)}} \\
=&\frac{\sqrt{\mathbf{s}_{jk}^{o'}(t-1) - \mathrm{C}_{jk}^{o'}(t)}}{\mathbf{s}_{jk}^{o'}(t-1)}
\end{aligned} \tag{A.2.8}$$

Therefore, the iterative procedures equation A.2.5 and equation A.2.6 for $\mathbf{w}_{jk}^{o'}(t)$ and $\mathbf{s}_{jk}^{o'}(t)$ can be formulated as

$$\begin{aligned}
&\left[\mathbf{w}_{jk}^{o'}(t), \mathbf{s}_{jk}^{o'}(t)\right] \\
=&\arg\min_{\mathbf{s}_{jk}^{o'}(t)\succeq\mathbf{0}, \mathbf{w}_{jk}^{o'}(t)} (\mathbf{w}_{jk}^{o'}(t) - \mathbf{w}_{jk}^{o'^*})\mathbf{H}(\mathbf{w}_{jk}^{o'^*})(\mathbf{w}_{jk}^{o'}(t) - \mathbf{w}_{jk}^{o'^*}) \\
&+ 2\mathbf{w}_{jk}^{o'}(t)\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) + \left(\frac{\mathbf{w}_{jk}^{o'}(t)^2}{\mathbf{s}_{jk}^{o'}(t-1)} + \omega_{jk}^{o'}(t)^2\mathbf{s}_{jk}^{o'}(t-1)\right) \\
=&\arg\min_{\mathbf{w}_{jk}^{o'}(t)} \mathbf{w}_{jk}^{o'}(t)\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'}(t) \\
&+ 2\mathbf{w}_{jk}^{o'}(t)\left(\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) - \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'}(t)^*\right) + \left(\frac{\mathbf{w}_{jk}^{o'}(t)^2}{\mathbf{s}_{jk}^{o'}(t)} + \omega_{jk}^{o'}(t)^2\mathbf{s}_{jk}^{o'}(t)\right).
\end{aligned} \tag{A.2.9}$$

Or in the compact form

$$\begin{aligned}
&\left[\mathbf{w}_{jk}^{o'}(t), \mathbf{s}_{jk}^{o'}(t)\right] \\
=&\arg\min_{\mathbf{w}_{jk}^{o'}(t)} \mathbf{w}_{jk}^{o'}(t)\mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'}(t) + 2\mathbf{w}_{jk}^{o'}(t)\left(\mathbf{g}(\mathbf{w}_{jk}^{o'^*}) - \mathbf{H}(\mathbf{w}_{jk}^{o'^*})\mathbf{w}_{jk}^{o'}(t)^*\right) \\
&+ \mathbf{w}_{jk}^{o'}(t)^2\mathbf{s}_{jk}^{o'}(t)^{-1} + \omega_{jk}^{o'}(t)^2\mathbf{s}_{jk}^{o'}(t).
\end{aligned} \tag{A.2.10}$$

Since

$$\frac{\mathbf{w}_{jk}^{o'}(t)^2}{\mathbf{s}_{jk}^{o'}(t)} + \omega_{jk}^{o'}(t)^2 \mathbf{s}_{jk}^{o'}(t) \geq 2 \left| \sqrt{\omega_{jk}^{o'}(t)^2} \cdot \mathbf{w}_{jk}^{o'}(t) \right|,$$

the optimal $\mathbf{s}_{jk}^{o'}(t)$ can be obtained as:

$$\mathbf{s}_{jk}^{o'}(t) = \frac{|\mathbf{w}_{jk}^{o'}(t)|}{\sqrt{\omega_{jk}^{o'}(t)^2}}, \forall i. \tag{A.2.11}$$

$\mathbf{w}_{jk}^{o'}(t)$ can be obtained as follows

$$\mathbf{w}_{jk}^{o'}(t) = \arg\min_{\mathbf{w}_{jk}^{o'}} \frac{1}{2} \mathbf{w}_{jk}^{o'}(t) \mathbf{H}(\mathbf{w}_{jk}^{o'}(t)^*) \mathbf{w}_{jk}^{o'}(t) + \mathbf{w}_{jk}^{o'}(t) \left(\mathbf{g}(\mathbf{w}_{jk}^{o'}(t)^*) - \mathbf{H}(\mathbf{w}_{jk}^{o'}(t)^*) \mathbf{w}_{jk}^{o'}(t)^* \right) + \|\omega_{jk}^{o'} \cdot \mathbf{w}_{jk}^{o'}(t)\|_{\ell_1}. \tag{A.2.12}$$

We can then inject this into equation A.2.11, which yields

$$\mathbf{s}_{jk}^{o'}(t) = \frac{\mathbf{w}_{jk}^{o'}(t)}{\omega_{jk}^{o'}(t)} \tag{A.2.13}$$

The update rules for $\mathbf{s}_{jk}^{o'}$ without considering the dependency has been explained above. However, as illustrated in Sec .4, the dependency between a node and its predecessors should not be disregarded. It means the dependency between edge $e_{jk}^{o'}$ and $\sum_{i<j} e_{ij}^{o}$ should be taken into consideration, then Gaussian prior could be defined as equation 5 and equation 11:

$$p(\mathbf{w} \mid \mathbf{s}) = \prod_{j<k} \prod_{o \in \mathcal{O}} \prod_{o' \in \mathcal{O}} p(w_{jk}^{o'} \sum_{i<j} w_{jk}^{o}) = \prod_{j<k} \prod_{o \in \mathcal{O}} \prod_{o' \in \mathcal{O}} \mathcal{N}(w_{jk}^{o'} \sum_{i<j} w_{ij}^{o} | 0, \mathbf{s}_{jk}^{o'})$$

based on this prior, the uncertainty of $\mathbf{w}_{jk}^{o'}(t)$ should be computed as:

$$\gamma_{ij}^{o'}(t) = \left(\frac{1}{\sum_{i<j} \sum_{o \in \mathcal{O}} \mathbf{s}_{ij}^{o}(t)} + \frac{1}{\mathbf{s}_{jk}^{o'}(t)} \right)^{-1} \tag{A.2.14}$$

As we found the expression of equation A.2.8, $\omega_{jk}^{o'}(t)$ is function of $\mathbf{s}_{jk}^{o'}(t-1)$, therefore $\mathbf{s}_{jk}^{o'}(t)$ is function of $\mathbf{s}_{jk}^{o'}(t-1)$ and $\mathbf{w}_{jk}^{o'}(t)$. We notice that the update for $\mathbf{w}_{jk}^{o'}(t)$ is to use the *lasso* or ℓ_1*-regularised* regression type optimization. The pseudo code is summarised in Algorithm 1.

A.3. Algorithm for Proxy Tasks

Our algorithm can be easily transferred to the scenario of proxy tasks to find the cell. Suppose a network is assembled by stacking O different kinds of cells together, such as $\aleph_1$ normal cells and $\aleph_O$ reduction cells in (Liu et al., 2019b). Then optimal O cells are required to be designed in a NAS task. As explained before, we design a switch $\mathbf{s}$ for each architecture parameter w to determine the "on-off" of the corresponding edge in our method. In order to find such optimal cells, we propose that switches on the same position of the identical kind of cells should also be same. Based on this, the architecture parameters could be divided into different groups. The general grouped architecture parameters are given as follows:

$$\mathbf{w}_{jk}^{o'}(t) = \Big[\underbrace{\mathbf{w}_{jk,1}^{o'1}(t), \ldots, \mathbf{w}_{jk,1}^{o'\aleph_1}(t)}_{\aleph_1 \textbf{ elements}} \Big| \ \ldots \ \Big| \underbrace{\mathbf{w}_{jk,O}^{o'1}(t), \ldots, \mathbf{w}_{jk,O}^{o'\aleph_O}(t)}_{\aleph_O \textbf{ elements}} \Big]. \tag{A.3.6}$$

Similar to equation A.2.11, if the group o is consist of $\aleph_o$ elements, where $o = 1, \ldots, O$, the optimal $s_{jk,o}^{o'}$ can be obtained as:

$$\sum_{i=1}^{\aleph_o} \left(\frac{\mathbf{w}_{jk,o}^{o'\top}(t) \mathbf{w}_{jk,o}^{o'}(t)}{s_{jk,o}^{o'}(t)} + \omega_{jk,o}^{o'i}(t)^2 s_{jk,o}^{o'}(t) \right) \geq 2 \left\| \sqrt{\sum_{i=1}^{\aleph_o} \omega_{jk,o}^{o'i}(t)^2} \cdot \mathbf{w}_{jk,o}^{o'}(t) \right\|_{\ell_2}, \tag{A.3.7}$$

Algorithm 2 The proposed Algorithm is transferable for cell selection on proxy tasks.

Initialization: $\boldsymbol{\gamma}_{jk}^{o'}(0), \boldsymbol{\omega}_{jk}^{o'}(0), \mathbf{w}(0) = \mathbf{1}$; sparsity intensity $\lambda_w^o \in \mathbb{R}^+$; $\lambda = 0.01$; cost function $\mathcal{L}_D$ in equation 16
Iteration:
for $t = 1$ to $T_{\max}$ **do**
1. Maximum likelihood with regularization:

$$\min_{\mathcal{W},\mathbf{w}} E_D(\cdot) + \lambda_w \sum_g \sum_{j<k} \sum_{o' \in \mathcal{O}} \|\omega_{jk,g}^{o'}(t)\mathbf{w}_{jk,g}^{o'}\|_2 + \lambda \|\mathcal{W}\|_2^2 \tag{A.3.1}$$

2. Compute Hessian for $\mathbf{w}$ (equation C.2.2, C.3.1, C.3.2)
3. Update variables associated with $\mathbf{w}$
while $g \in (1, O); i < j < k; o, o' \in \mathcal{O}$ **do**

$$C_{jk}^{o'}(t) = \left(\frac{1}{\gamma_{jk}^{o'}(t-1)} + \mathbf{H}_{jk}^{o'}(t) \right)^{-1} \tag{A.3.2}$$

$$\omega_{jk}^{o'}(t) \text{ is given by } A.3.9 \text{ and } A.3.11 \tag{A.3.3}$$

$$\mathbf{s}_{jk}^{o'}(t) \text{ is given by } A.3.8 \text{ and } A.3.10 \tag{A.3.4}$$

$$\gamma_{jk}^{o'}(t) \text{ is given by 6 or 8}, \gamma_{jk}^{o'}(t) = \Big[\underbrace{\gamma_{jk,1}^{o'}(t), \ldots, \gamma_{jk,1}^{o'}(t)}_{\aleph_1 \text{ elements}} \Big| \ \ldots \ \Big| \underbrace{\gamma_{jk,O}^{o'}(t), \ldots, \gamma_{jk,O}^{o'}(t)}_{\aleph_O \text{ elements}} \Big] \tag{A.3.5}$$

end while
4. Prune the architecture if the entropy $\frac{\ln(2\pi e \gamma_{jk}^{o'})}{2} \leq 0$
5. Fix $\mathbf{w} = \mathbf{1}$, train the pruned net in the standard way
end for

then

$$s_{jk,o}^{o'}(t) = \frac{\left\| \mathbf{w}_{jk,o}^{o'}(t) \right\|_{\ell_2}}{\sqrt{\sum_{i=1}^{\aleph_o} \omega_{jk,o}^{o'i}(t)^2}}, \forall i. \tag{A.3.8}$$

The calculation of $\omega_{jk,g}^{o'}$ for group o is:

$$\omega_{jk,o}^{o'}(t) = \sqrt{\sum_{i=1}^{\aleph_o} \frac{\sqrt{\gamma_{jk,o}^{o'i}(t-1) - \mathrm{C}_{jk,o}^{o'i}(t)}}{\gamma_{jk,o}^{o'i}(t-1)^2}} \tag{A.3.9}$$

and both $\mathbf{s}$ and ω for the different elements in identity group should keep the same:

$$\mathbf{s}_{jk}^{o'}(t) = \Big[\underbrace{\mathbf{s}_{jk,1}^{o'}(t), \ldots, \mathbf{s}_{jk,1}^{o'}(t)}_{\aleph_1 \text{ elements}} \Big| \ \ldots \ \Big| \underbrace{\mathbf{s}_{jk,O}^{o'}(t), \ldots, \mathbf{s}_{jk,O}^{o'}(t)}_{\aleph_O \text{ elements}} \Big] \tag{A.3.10}$$

$$\omega_{jk}^{o'}(t) = \left[\underbrace{\omega_{jk,1}^{o'}(t), \ldots, \omega_{jk,1}^{o'}(t)}_{\aleph_1 \text{ elements}} \Big| \ \ldots \ \Big| \underbrace{\omega_{jk,O}^{o'}(t), \ldots, \omega_{jk,O}^{o'}(t)}_{\aleph_O \text{ elements}} \right] \tag{A.3.11}$$

It should be noted that the detailed derivation procedures can be referred to A.1 and A.2. The pseudo code is summarised in Algorithm 2.

B. Structural Bayesian Deep Compression

In addition to applying the proposed Bayesian approach to address NAS problem, we also explore the possibility of our method on network structural compression problem. In this section, we extend to compress deep neural networks by proposing a series of generic and easily implemented reweighted group Lasso algorithms to solve maximization of marginal

Algorithm 3 The proposed Algorithm is transferable for neural network compression.

Initialization: Initialization: $\forall l = 1, \ldots, L, \omega^l(0), \gamma^l(0) = 1$; $\lambda^l \in \mathbb{R}^+$;
Iteration:
for $t = 1$ to $T_{\max}$ **do**
1. Maximum likelihood with regularization:

$$\min_{\mathcal{W}} E_D(\cdot) + \sum_{l=1}^{L} \lambda^l R(\omega^l(t) \circ \mathcal{W}^l) \tag{B.0.1}$$

2. Compute the Hessian for fully connected layer and convolutional layers as Appendix. C.1 and C.2 respectively.
3. Update hyper-parameters:
{Update() specifies how to update parameters, detailed update rules are in Table S3}

$$\gamma^l(t) \leftarrow \text{Update}(\omega^l(t-1), \mathcal{W}^l(t)), \Gamma^l(t) = [\gamma^l(t)] \tag{B.0.2}$$

$$C^l(t) \leftarrow \left((\Gamma^l(t))^{-1} + \text{diag}(\text{H}^l(t))\right)^{-1}, \tag{B.0.3}$$

$$\alpha^l(t) \leftarrow -\frac{\text{C}^l(t)}{\gamma^l(t)^2} + \frac{1}{\gamma^l(t)}\{\text{element-wise division}\} \tag{B.0.4}$$

$$\omega^l(t) \leftarrow \text{Update}(\alpha^l(t)) \tag{B.0.5}$$

4. Prune the unimportant connections.
end for

likelihood $\int p(\mathbf{Y} \mid \boldsymbol{\mathcal{W}}) p(\boldsymbol{\mathcal{W}}) d\boldsymbol{\mathcal{W}}$ where $p(\boldsymbol{\mathcal{W}})$ can be specified as various sparse structured priors over network weights as shown in Table S3. The proposed Algorithm is generic for the weights in fully connected and convolutional neural networks. The training algorithm is iteratively indexed by t. Each iteration contains several epochs. Within each iteration t, there are three parts, The first part is simply a reweighted group Lasso type optimization. In the regularization terms $R(\omega^l \circ \mathcal{W}^l)$, each weight of layer l is scaled by a factor (ω^l). The update of (ω^l) needs the Hessian of each $\boldsymbol{\mathcal{W}}^l$. The Hessian of each layer can be computed recursively given the Hessian in the next layer through backward passing. The hyper-parameters γ^l, C^l and ω^l will be updated every iteration t with $T_{\max}$ being the maximal iterations. α^l is an introduced intermediate variable during the update process. The pseudo code is summarized in Algorithm 3.

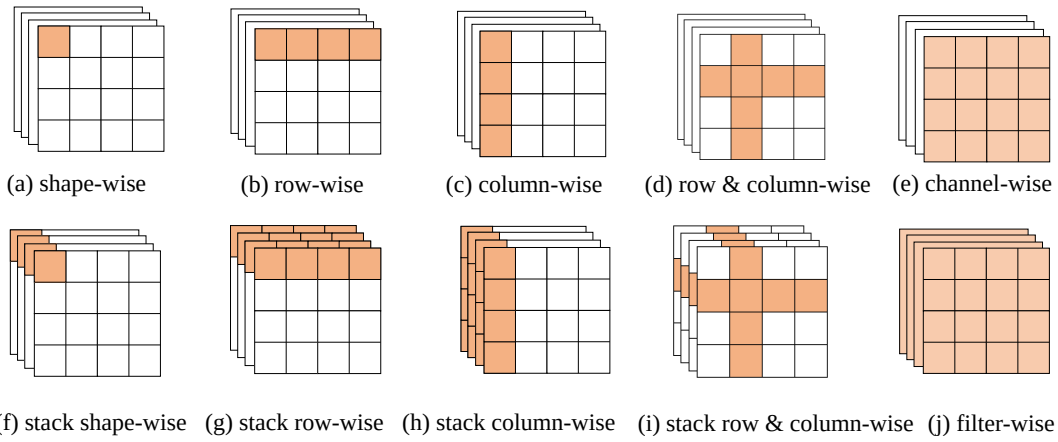
B.1. Structured Sparse Prior for Fully Connected and Convolutional Networks

For weight in the l-th convolutional layer $W^l \in \mathbb{R}^{N_l \times C_l \times m_l \times k_l}$, some examples of the structured sparsity are shown in Fig.S5. The corresponding sparse prior is given in the second column of Table S3. It should be noted that the prior for the weight in fc layer could also be represented as this table with $m_l = k_l = 1$, N_l and C_l stands for the size of input feature and output feature respectively.

B.2. Experiments

B.2.1. LeNet-300-100 and LeNet-5 on MNIST

We first perform LeNet-300-100 and LeNet-5 on MNIST dataset (LeCun, 1998). For LeNet-300-100, we apply shape-wise, row-wise and column-wise regularization as shown in Fig. S5(a), S5(b) and S5(c), for the 2D weight matrices. The hyper-parameters γ, ω and α are updated every ten epochs for a total of $T_{\max} = 10$ loops. The learned structure is $465 - 37 - 90$ with 1.54% test error and 0.04 FLOPS (Molchanov et al., 2017b). Comparison with other methods can be found in Table S4. For LeNet-5, we apply shape-wise and filter-wise regularization for the conv layer as shown in Fig. S5(a) and S5(j); row-wise and column-wise regularization for fc layer as shown in Fig. S5(b) and S5(c). The learned structure is $5 - 10 - 65 - 25$ with 1.00% test error and 0.57 FLOPS. Comparison with other methods can be found in Table S5.



(a) shape-wise (b) row-wise (c) column-wise (d) row & column-wise (e) channel-wise

(f) stack shape-wise (g) stack row-wise (h) stack column-wise (i) stack row & column-wise (j) filter-wise

Figure S5. some examples of structured sparsity for the 3D filters in Conv layer with extensions of (Wen et al., 2016). Coloured squares mean the weights to be pruned. It should be noted that the FC layer can be easily enforced by (a)-(e).

B.2.2. RESNET-18 ON CIFAR-10

We also evaluate our algorithm on Cifar10 dataset using ResNet-18 as initialized backbone (He et al., 2016). In addition to the input conv layer and output fc layer, the other 16 conv layers are separated into 8 blocks with 2 layers each. We apply shape-wise and filter-wise regularization to the conv layer as shown in Fig. S5(a) and S5(j); row-wise and column-wise regularization to the fc layer as shown in Fig. S5(b) and S5(c). The result is given in Table S6. It can be found that two Conv layers in block 4 are pruned away which shows the potential of our method to reduce the number of layers.

C. Efficient Hessian Computation

C.1. Compute the Hessian of FC Layer

The mathematical operation in a fully-connected layer could be formulated as:

$$h_j^o = W_{ij}^o a_i, \quad a_i = \sigma(h_i) \tag{C.1.1}$$

where h_i is the pre-activation value for node i and a_i is the activation value. $\sigma()$ is the element-wise activation function. $\mathcal{W}_{ij}^o$ stands for the weight matrix associated with operation o in edge e_{ij}^o. In (Botev et al., 2017), a recursive method is proposed to compute the Hessian $\mathbf{H}$ for $\mathcal{W}_{ij}^o$:

$$\mathbf{H}_{ij}^o = a_i \cdot (a_i)^\top \otimes H_j^o \tag{C.1.2}$$

where $\otimes$ stands for Kronecker product; The pre-activation Hessian H_j^o is known and could be used to compute the pre-activation Hessian recursively for the previous layer:

$$H_i = B_i(\mathcal{W}_{ij}^o)^\top H_j^o \mathcal{W}_{ij}^o B_i + D_i, \quad B_i = \mathrm{diag}(\sigma'(h_i)), \quad D_i = \mathrm{diag}(\sigma''(h_i) \circ \frac{\partial L}{\partial a_i}) \tag{C.1.3}$$

In order to reduce computation complexity, the original pre-activation Hessian H and Hessian $\mathbf{H}$ in Eq C.1.2-C.1.3 are replaced with their diagonal values for recursive computation. Thus the matrix multiplication could be reduced to vector multiplication. The hessian calculation process could be reformulated as:

$$\mathbf{H}_{ij}^o = a_i^2 \otimes H_j^o \tag{C.1.4}$$

$$H_i = B_i^2 \circ (((\mathcal{W}_{ij}^o)^\top)^2 H_j^o) + D_i, \quad B_i = \sigma'(H_i), \quad D_i = \sigma''(H_i) \circ \frac{\partial L}{\partial a_i} \tag{C.1.5}$$

Table S3. Hyper-parameter update rule in Algorithm3 for CNN

Category	sparse prior	$R^l(\omega \circ \boldsymbol{\mathcal{W}})$	ω^l	γ^l
(a) Shape-wise	$\prod_{c_l}\prod_{m_l}\prod_{k_l}\mathcal{N}(\mathbf{0},\gamma_{c_l,m_l,k_l}\mathbf{I}_{n_l})$	$\sum_{c_l=1}^{C_l}\sum_{m_l=1}^{M_l}\sum_{k_l=1}^{K_l}\|\omega^l_{:,c_l,m_l,k_l}\circ\boldsymbol{\mathcal{W}}^l_{:,c_l,m_l,k_l}\|_2$	$\omega^l_o=\sqrt{\sum_{c_l}\sum_{m_l}\sum_{k_l}\|\alpha^l_{:,c_l,m_l,k_l}\|}$ $\omega^l_{:,c_l,m_l,k_l}=\omega^l_o\cdot\mathbf{I}^l_{:,c_l,m_l,k_l}$	$\gamma^l_o=\frac{\|\boldsymbol{\mathcal{W}}^l_{:,c_l,m_l,k_l}\|_2}{\omega^{l\,(t-1)}_{:,c_l,m_l,k_l}}$ $\gamma^l_{:,c_l,m_l,k_l}=\gamma^l_o\cdot\mathbf{I}^l_{:,c_l,m_l,k_l}$
(b) Row-wise	$\prod_{c_l}\prod_{m_l}\mathcal{N}(\mathbf{0},\gamma_{c_l,m_l}\mathbf{I}_{n_lk_l})$	$\sum_{c_l=1}^{C_l}\sum_{m_l=1}^{M_l}\|\omega_{:,c_l,m_l,:}\circ\boldsymbol{\mathcal{W}}^l_{:,c_l,m_l,:}\|_2$	$\omega^l_o=\sqrt{\sum_{c_l}\sum_{m_l}\|\alpha^l_{:,c_l,m_l,:}\|}$ $\omega^l_{:,c_l,m_l,:}=\omega^l_o\cdot\mathbf{I}^l_{:,c_l,m_l,:}$	$\gamma^l_o=\frac{\|\boldsymbol{\mathcal{W}}^l_{:,c_l,m_l,:}\|_2}{\omega^{l\,(t-1)}_{:,c_l,m_l,:}}$ $\gamma^l_{:,c_l,m_l,:}=\gamma^l_o\cdot\mathbf{I}^l_{:,c_l,m_l,:}$
(c) Column-wise	$\prod_{c_l}\prod_{k_l}\mathcal{N}(\mathbf{0},\gamma_{c_l,k_l}\mathbf{I}_{n_lm_l})$	$\sum_{c_l=1}^{C_l}\sum_{k_l=1}^{K_l}\|\omega^l_{:,c_l,:,k_l}\circ\boldsymbol{\mathcal{W}}^l_{:,c_l,:,k_l}\|_2$	$\omega^l_o=\sqrt{\sum_{c_l}\sum_{k_l}\|\alpha^l_{:,c_l,:,k_l}\|}$ $\omega^l_{:,c_l,:,k_l}=\omega^l_o\cdot\mathbf{I}^l_{:,c_l,:,k_l}$	$\gamma^l_o=\frac{\|\boldsymbol{\mathcal{W}}^l_{:,c_l,:,k_l}\|_2}{\omega^{l\,(t-1)}_{:,c_l,:,k_l}}$ $\gamma^l_{:,c_l,:,k_l}=\gamma^l_o\cdot\mathbf{I}^l_{:,c_l,:,k_l}$
(d) Row & column-wise	$\prod_{c_l}\prod_{m_lk_l}\mathcal{N}(\mathbf{0},\gamma_{c_l,m_lk_l}\mathbf{I}_{n_l})$	$\bar{W}^l=[\boldsymbol{\mathcal{W}}^l_{:,c_l,m_l,:},\boldsymbol{\mathcal{W}}^l_{:,c_l,:,k_l}]$ $\bar{\omega}^l=[\omega^l_{:,c_l,m_l,:},\omega^l_{:,c_l,:,k_l}]$ $\sum_{c_l=1}^{C_l}\sum\|\bar{\omega}^l\circ\bar{W}^l\|_2$	$\omega^l_o=\sqrt{\sum_{c_l}\sum_{m_l}\|\alpha^l_{:,c_l,m_l,:}\|+\sum_{c_l}\sum_{k_l}\|\alpha^l_{:,c_l,:,k_l}\|}$ $\omega^l_{:,c_l,m_l,:}=\omega^l_o\cdot\mathbf{I}^l_{:,c_l,m_l,:}$ $\omega^l_{:,c_l,:,k_l}=\omega^l_o\cdot\mathbf{I}^l_{:,c_l,:,k_l}$	$\gamma^l_o=\frac{\|\bar{W}^l\|_2}{\bar{\omega}^{l\,(t-1)}}$ $\gamma^l_{:,c_l,m_l,:}=\gamma^l_o\cdot\mathbf{I}^l_{:,c_l,m_l,:}$ $\gamma^l_{:,c_l,:,k_l}=\gamma^l_o\cdot\mathbf{I}^l_{:,c_l,:,k_l}$
(e) Channel-wise	$\prod_{c_l}\mathcal{N}(\mathbf{0},\gamma_{c_l}\mathbf{I}_{n_lm_lk_l})$	$\sum_{c_l=1}^{C_l}\|\omega^l_{:,c_l,:,:}\circ\boldsymbol{\mathcal{W}}^l_{:,c_l,:,:}\|_2$	$\omega^l_o=\sqrt{\sum_{c_l}\|\alpha^l_{:,c_l,:,:}\|}$ $\omega^l_{:,c_l,:,:}=\omega^l_o\cdot\mathbf{I}^l_{:,c_l,:,:}$	$\gamma^l_o=\frac{\|\boldsymbol{\mathcal{W}}^l_{:,c_l,:,:}\|_2}{\omega^{l\,(t-1)}_{:,c_l,:,:}}$ $\gamma^l_{:,c_l,:,:}=\gamma^l_o\cdot\mathbf{I}^l_{:,c_l,:,:}$
(f) Group shape-wise	$\prod_{m_l}\prod_{k_l}\mathcal{N}(\mathbf{0},\gamma_{m_l,k_l}\mathbf{I}_{n_lc_l})$	$\sum_{m_l=1}^{M_l}\sum_{k_l=1}^{K_l}\|\omega^l_{:,:,m_l,k_l}\circ\boldsymbol{\mathcal{W}}^l_{:,:,m_l,k_l}\|_2$	$\omega^l_o=\sqrt{\sum_{m_l}\sum_{k_l}\|\alpha^l_{:,:,m_l,k_l}\|}$ $\omega^l_{:,:,m_l,k_l}=\omega^l_o\cdot\mathbf{I}^l_{:,:,m_l,k_l}$	$\gamma^l_o=\frac{\|\boldsymbol{\mathcal{W}}^l_{:,:,m_l,k_l}\|_2}{\omega^{l\,(t-1)}_{:,:,m_l,k_l}}$ $\gamma^l_{:,:,m_l,k_l}=\gamma^l_o\cdot\mathbf{I}^l_{:,:,m_l,k_l}$
(g) Group row-wise	$\prod_{m_l}\mathcal{N}(\mathbf{0},\gamma_{m_l}\mathbf{I}_{n_lc_lk_l})$	$\sum_{m_l=1}^{M_l}\|\omega^l_{:,:,m_l,:}\circ\boldsymbol{\mathcal{W}}^l_{:,:,m_l,:}\|_2$	$\omega^l_o=\sqrt{\sum_{m_l}\|\alpha^l_{:,:,m_l,:}\|}$ $\omega^l_{:,:,m_l,:}=\omega^l_o\cdot\mathbf{I}^l_{:,:,m_l,:}$	$\gamma^l_o=\frac{\|\boldsymbol{\mathcal{W}}^l_{:,:,m_l,:}\|_2}{\omega^{l\,(t-1)}_{:,:,m_l,:}}$ $\gamma^l_{:,:,m_l,:}=\gamma^l_o\cdot\mathbf{I}^l_{:,:,m_l,:}$
(h) Group column-wise	$\prod_{k_l}\mathcal{N}(\mathbf{0},\gamma_{k_l}\mathbf{I}_{n_lc_lm_l})$	$\sum_{k_l=1}^{K_l}\|\omega^l_{:,:,:,k_l}\circ\boldsymbol{\mathcal{W}}^l_{:,:,:,k_l}\|_2$	$\omega^l_o=\sqrt{\sum_{k_l}\|\alpha^l_{:,:,:,k_l}\|}$ $\omega^l_{:,:,:,k_l}=\omega^l_o\cdot\mathbf{I}^l_{:,:,:,k_l}$	$\gamma^l_o=\frac{\|\boldsymbol{\mathcal{W}}^l_{:,:,:,k_l}\|_2}{\omega^{l\,(t-1)}_{:,:,:,k_l}}$ $\gamma^l_{:,:,:,k_l}=\gamma^l_o\cdot\mathbf{I}^l_{:,:,:,k_l}$
(i) Group row & column-wise	$\prod_{m_lk_l}\mathcal{N}(\mathbf{0},\gamma_{m_lk_l}\mathbf{I}_{n_lc_l})$	$\bar{W}^l=[\boldsymbol{\mathcal{W}}^l_{:,:,m_l,:},\boldsymbol{\mathcal{W}}^l_{:,:,:,k_l}]$ $\bar{\omega}^l=[\omega^l_{:,:,m_l,:},\omega^l_{:,:,:,k_l}]$ $\sum\|\bar{\omega}^l\circ\bar{W}^l\|_2$	$\omega^l_o=\sqrt{\sum_{m_l}\|\alpha^l_{:,:,m_l,:}\|+\sum_{k_l}\|\alpha^l_{:,:,:,k_l}\|}$ $\omega^l_{:,:,m_l,:}=\omega^l_o\cdot\mathbf{I}^l_{:,:,m_l,:}$ $\omega^l_{:,:,:,k_l}=\omega^l_o\cdot\mathbf{I}^l_{:,:,:,k_l}$	$\gamma^l_o=\frac{\|\bar{W}^l\|_2}{\bar{\omega}^{l\,(t-1)}}$ $\gamma^l_{:,:,m_l,:}=\gamma^l_o\cdot\mathbf{I}^l_{:,:,m_l,:}$ $\gamma^l_{:,:,:,k_l}=\gamma^l_o\cdot\mathbf{I}^l_{:,:,:,k_l}$
(j) Filter-wise	$\prod_{n_l}\mathcal{N}(\mathbf{0},\gamma_{n_l}\mathbf{I}_{c_lm_lk_l})$	$\sum_{n_l=1}^{N_l}\|\omega^l_{n_l,:,:,:}\circ\boldsymbol{\mathcal{W}}^l_{n_l,:,:,:}\|_2$	$\omega^l_o=\sqrt{\sum_{n_l}\|\alpha^l_{n_l,:,:,:}\|}$ $\omega^l_{n_l,:,:,:}=\omega^l_o\cdot\mathbf{I}^l_{n_l,:,:,:}$	$\gamma^l_o=\frac{\|\boldsymbol{\mathcal{W}}^l_{n_l,:,:,:}\|_2}{\omega^{l\,(t-1)}_{n_l,:,:,:}}$ $\gamma^l_{n_l,:,:,:}=\gamma^l_o\cdot\mathbf{I}^l_{n_l,:,:,:}$

Table S4. Comparison of the learned architecture with other methods using LeNet-300-100 on MNIST dataset

Method	Pruned Architecture	Error Rate (%)	FLOPs (M)
Baseline	784-300-100	1.39	0.53
SBP ((Neklyudov et al., 2017))	245-160-55	1.60	0.10
BC-GNJ ((Louizos et al., 2017))	278-98-13	1.80	0.06
BC-GHS ((Louizos et al., 2017))	311-86-14	1.80	0.06
Practical ℓ_0((Louizos et al., 2018))	219-214-100	1.40	0.14
Practical ℓ_0 ((Louizos et al., 2018))	266-88-33	1.80	0.05
Proposed method	465-37-90	1.54	0.04

Where diag() means the operation to extract the diagonal values of input variable. If we compute Hessian with the approximate method as Eq C.1.4-C.1.5, the multiply accumulate operation (MACs) for the pre-activatiion Hessian H and Hessian $\mathbf{H}$ could be reduced from $n(2m^2+2n^2+4mn+3m-1)$ to $n(2+4m)$ with $W\in\mathbb{R}^{n\times m}$. (e.g. with $n=100, m=100$, the original method requires 107.97 MMACs compared with only 0.04 MMACs for the approximate method.)

C.2. Compute the Hessian of Conv Layer

Although the Hessian of weight matrix has been widely used in second-order optimization techniques to speed up the training process (LeCun et al., 1990; Amari, 1998), it still remains infeasible to calculate explicit Hessian directly due to the intensive computation burden (Martens & Grosse, 2015; Botev et al., 2017). Moreover, as most of current deep neural networks include plenty of Convolutional (Conv) layers, it further increases the difficulty of calculation due to the indirect convolution operation. Inspired by the Hessian calculation methods for Fully Connected (FC) layers as shown in (Botev et al., 2017), we propose a recursive and efficient method to compute the Hessian of Conv layers by converting Conv layers to FC layers (Ma & Lu, 2017). Therefore Hessian of the resulting equivalent FC layer is ready to be obtained. The detailed calculation procedures are explained in the following:

Suppose a convolution operation o is selected between node i and j ($i<j$). The corresponding input vector, weight

Table S5. Comparison of the learned architecture with other methods using LeNet-5 on MNIST dataset

Method	Pruned Architecture	Error Rate (%)	FLOPs (M)
Baseline	20-50-800-500	0.83	8.85
SBP ((Neklyudov et al., 2017))	3-18-284-283	0.90	0.69
BC-GNJ ((Louizos et al., 2017))	8-13-88-13	1.00	1.09
BC-GHS ((Louizos et al., 2017))	5-10-76-16	1.00	0.57
Practical ℓ_0((Louizos et al., 2018))	20-25-45-462	0.90	4.49
Practical ℓ_0 ((Louizos et al., 2018))	9-18-65-25	1.00	1.55
Proposed method	5-10-65-25	1.00	0.57

Table S6. Sparsity for each layer in ResNet-18 on Cifar10 dataset

conv1	conv2-5	conv6-9	conv10-13	con14-17	FC layer	Total	Test error
22.80%	10.06%	9.24%	20.64%	1.43%	10.33%	2.96%	6.58% (baseline)
	22.87%	5.45%	15.04%	0.19%			
	20.05%	4.94%	10.77%	0			6.23% (our method)
	12.99%	10.73%	4.61%	0			

and output vector of this edge are denoted as $B_i \in \mathbb{R}^{b\times C_i\times H_i\times W_i}$, $\mathcal{W}_{ij}^o \in \mathbb{R}^{C_j^o\times C_i\times m_{ij}^o\times k_{ij}^o}$ and $B_j^o \in \mathbb{R}^{b\times C_j^o\times H_j^o\times W_j^o}$ respectively, where b is the batch size, C_i, H_i, W_i are the size of input channel, height and width; C_j^o is the size of output channel, $m_{ij}^o \times k_{ij}^o$ is the kernel dimension; H_j^o and W_j^o are the size of output height and width.

As in (Ma & Lu, 2017), B_i is converted to two dimensional matrix for FC layer, with dimension $(bH_j^oW_j^o) \times (C_im_{ij}^ok_{ij}^o)$. Similarly, the dimension of B_j^o is changed from $b \times C_j^o \times H_j^o \times W_j^o$ to $(bH_j^oW_j^o) \times C_j^o$. The dimension of $\mathcal{W}_{ij}^o$ is changed to $\mathbb{R}^{(C_im_{ij}^ok_{ij}^o)\times C_j^o}$. The input vector, output vector and weight for the FC layer are denoted as M_i, M_j^o and $\mathcal{W}_{ij}^{oM}$.

Secondly, M_i, M_j^o and H_j^{oM} are decomposed into a total of $bH_j^oW_j^o$ row vectors with $(M_i)^n \in \mathbb{R}^{C_im_{ij}^ok_{ij}^o}$, $(M_j^o)^n \in \mathbb{R}^{C_j^o}$ and $(H_j^{oM})^n \in \mathbb{R}^{C_j^o}$ $(n = 1, \ldots, bH_j^oW_j^o)$ respectively. It is easy to understand that $(M_i)^n$, $(M_j^o)^n$ could be regarded as the input vector and output vector of a FC layer with weight matrix $\mathcal{W}_{ij}^{oM}$. Then we can obtain the Hessian $\mathbf{H}_{ij}^o$ for $\mathcal{W}_{ij}^o$ as follows:

$$(\mathbf{H}_{ij}^{oM})^n = (M_i)^n \cdot (M_i)^{n\top} \otimes (H_j^{oM})^n \tag{C.2.1}$$

$(H_j^{oM})^n$ is the pre-activation Hessian which could be computed recursively. With $(H_j^{oM})^n$ known, the pre-activation Hessian for $(M_i)^n$ could be calculated as:

$$\begin{aligned}
(H_i^M)^n &= (B_i)^n \mathcal{W}_{ij}^{oM\top} (H_j^{oM})^n \mathcal{W}_{ij}^{oM} (B_i)^n + (D_i)^n \\
(B_i)^n &= \operatorname{diag}\left(\sigma'((h_i)^n)\right) \\
(D_i)^n &= \operatorname{diag}\left(\sigma''((h_i)^n)\frac{\partial L}{\partial (M_i)^n}\right)
\end{aligned}$$

where $(h_i)^n$ is the pre-activation value for FC layer and L means the loss function. The pre-activation Hessian H_i^M could be obtained after concatenating all $(H_i^M)^n$ as

$$H_i^M = [\operatorname{diag}((H_i^M)^1); \ldots; \operatorname{diag}((H_i^M)^{bH_j^oW_j^o})] \tag{C.2.2}$$

the Hessian $\mathbf{H}_{ij}^{oM}$ for $\mathcal{W}_{ij}^{oM}$ can be obtained as:

$$\mathbf{H}_{ij}^{oM} = \frac{1}{bH_j^oW_j^o} \sum_{n=1}^{bH_j^oW_j^o} (\mathbf{H}_{ij}^{oM})^n \tag{C.2.3}$$

It should be noted that as pre-activation Hessian is a recursive variable for convolutional layer and Hessian will be used for updating hyper-parameters which will be introduced later, both H_i^M and $\mathbf{H}_{ij}^{oM}$ should be converted back to conv type before imparting to next layer with dimension $\mathbb{R}^{b\times C_i\times H_i\times W_i}$ and $\mathbb{R}^{C_j^o\times C_i\times m_{ij}^o\times k_{ij}^o}$.

As analyzed in Sec C.1, the Hessian calculation may cost a lot of time and resource. In order to address this problem, we propose the following approximate method:

$$\mathbf{H}_{ij}^{oM} = \mathbb{E}(M_i)^2 \otimes \mathbb{E}(H_j^{oM}) \tag{C.2.4}$$

$$H_i^M = \underbrace{\left[\hat{H}_i^M; \ldots; \hat{H}_i^M\right]}_{(bH_{\text{out}}W_{\text{out}})}, \quad \hat{H}_i^M = (B_i)^2 \circ ((\mathcal{W}_{ij}^{oM})^2 \mathbb{E}(H_j^{oM})^\top) + D_i \tag{C.2.5}$$

where $B_i = \mathbb{E}(\sigma'(h_i)), \quad D_i = \mathbb{E}(\sigma''(h_i) \circ \frac{\partial L}{\partial M_i})$; $\mathbb{E}()$ returns a vector which stands for the mean value of input variable along feature map. An approximate pre-activation Hessian is calculated without decomposition of input variables, which saves more than $bH_j^o W_j^o$ times multiply-accumulate operations (MACs).

C.3. Compute the Hessian of Architecture Parameter

After we have the computation method for the Hessian of a Convolutional layer, we need to consider the Hessian of an architecture parameter. Now the output from node i to j under operation o becomes $w_{ij}^o B_i$, where w_{ij}^o is the architecture parameter and B_i stands for the input vector.

Inspired by (Botev et al., 2017), the Hessian for w_{ij}^o could be computed recursively as $\mathbf{H}_{ij}^o = \mathbb{E}(\sum (B_i)^2 H_j)$, where H_j is supposed to be the known pre-activation Hessian for B_j and H_i is the pre-activation Hessian for B_i.

$$H_i = \sum_{o \in \mathcal{O}} (w_{ij}^o)^2 H_j. \tag{C.3.1}$$

Since B_i and H_j are independent of each other, the Hessian $\mathbf{H}_{ij}^o$ could also be calculated more efficiently:

$$\mathbf{H}_{ij}^o = (\mathbb{E}(|B_i|)^2 H_j \tag{C.3.2}$$

where $\mathbb{E}$ will return the mean.

D. Detailed Settings of Experiments

D.1. Architecture Search on CIFAR-10

Data Pre-processing and Augmentation Techniques We employ the following techniques in our experiments: centrally padding the training images to 40×40 and then randomly cropping them back to 32×32; randomly flipping the training images horizontally; normalizing the training and validation images by subtracting the channel mean and dividing by the channel standard deviation.

Implementation Details of Operations The operations include: 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated convolutions, 3×3 max pooling, 3×3 average pooling, and skip connection. All operations are of stride one (excluded the ones adjacent to the input nodes in the reduction cell, which are of stride two) and the convolved feature maps are padded to preserve their spatial resolution. Convolutions are applied in the order of BN-ReLU-Conv and the depthwise separable convolution is always applied twice (Zoph et al., 2018; Real et al., 2019; Liu et al., 2018a; 2019b).

Detailed Training Settings The network parameters are optimized using momentum SGD, with initial learning rate $\eta_{\boldsymbol{\theta}} = 0.1$, momentum 0.9, and weight decay 1×10^{-4}. The batch size employed is 16 and the initial number of channels is 16.

D.2. Architecture evaluation on CIFAR-10

Additional Enhancement Techniques Following existing works (Zoph et al., 2018; Liu et al., 2018a; Pham et al., 2018; Real et al., 2019; Liu et al., 2019b), we employ the following additional enhancements: cutout (DeVries & Taylor, 2017).

D.3. Architecture transferability evaluation on CIFAR-10

Detailed Training Settings The network is trained with batch size 128, SGD optimizer with weight decay 3×10^{-4}, momentum 0.9 and initial learning rate 0.1, which is decayed using cosine annealing.

D.4. Cells for $\lambda_w^o = 0.007$

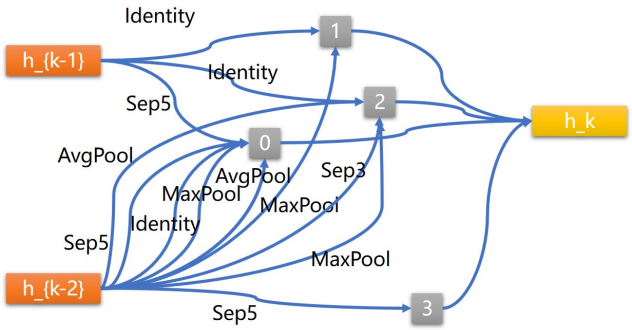


Figure S6. Normal cell found by BayesNAS with $\lambda_w^o = 0.007$.

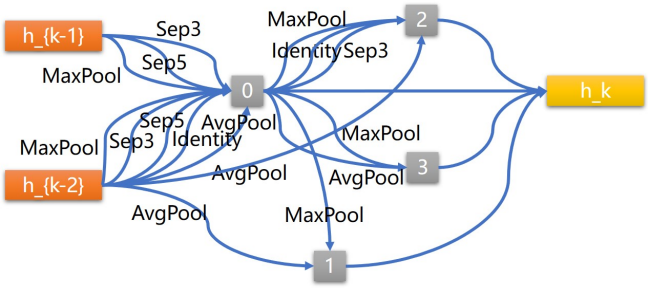


Figure S7. Reduction cell found by BayesNAS with $\lambda_w^o = 0.007$.