Appendix for:
Efficient Neural Architecture Search via Parameters Sharing

A. Details on Penn Treebank Experiments

Computations in an RNN Cell. We think of the cell at time step t as a DAG with N computational nodes, indexed by $\mathbf{h}_1^{(t)}$, $\mathbf{h}_2^{(t)}$, ... $\mathbf{h}_N^{(t)}$. Node $\mathbf{h}_1^{(t)}$ receives two inputs: 1) the RNN signal $\mathbf{x}^{(t)}$ at its current time step; and 2) the output $\mathbf{h}_D^{(t-1)}$ from the cell at the previous time step. The following computations are performed:

$$\mathbf{c}_1^{(t)} \leftarrow \text{sigmoid}\left(\mathbf{x}^{(t)} \cdot \mathbf{W}^{(\mathbf{x},\mathbf{c})} + \mathbf{h}_N^{(t-1)} \cdot, \mathbf{W}_0^{(\mathbf{c})}\right) \quad (2)$$

$$\mathbf{h}_1^{(t)} \leftarrow \mathbf{c}_1^{(t)} \otimes f_1\left(\mathbf{x}^{(t)} \cdot \mathbf{W}^{(\mathbf{x},\mathbf{h})} + \mathbf{h}_N^{(t-1)} \cdot \mathbf{W}_1^{(\mathbf{h})}\right) + (1 - \mathbf{c}_1^{(t)}) \otimes \mathbf{h}_N^{(t-1)}, \quad (3)$$

where f_1 is an activation function that the controller will decide. For $\ell = 2, 3, ..., N$, node $\mathbf{h}_\ell$ receives its input from a layer $j_\ell \in \{\mathbf{h}_1, ..., \mathbf{h}_{\ell-1}\}$, which is specified by the controller, and then performs the following computations:

$$\mathbf{c}_\ell^{(t)} \leftarrow \text{sigmoid}\left(\mathbf{h}_{j_\ell}^{(t)} \cdot \mathbf{W}_{\ell,j_\ell}^{(\mathbf{c})}\right) \quad (4)$$

$$\mathbf{h}_\ell^{(t)} \leftarrow \mathbf{c}_\ell^{(t)} \otimes f_\ell\left(\mathbf{h}_{j_\ell}^{(t)} \cdot \mathbf{W}_{\ell,j_\ell}^{(\mathbf{h})}\right) + (1 - \mathbf{c}_\ell^{(t)}) \otimes \mathbf{h}_{j_\ell}^{(t)}. \quad (5)$$

Therefore, the shared parameters ω among different recurrent cells consist of all the matrices $\mathbf{W}^{(\mathbf{x},\mathbf{c})}$, $\mathbf{W}^{(\mathbf{x},\mathbf{h})}$, $\mathbf{W}_{\ell,j}^{(\mathbf{c})}$, $\mathbf{W}_{\ell,j}^{(\mathbf{h})}$, word embeddings, and the softmax weights if they are not tied with the word embeddings. The controller decides the connection j_ℓ and the activation function f_ℓ for each $\ell \in \{2, 3, ..., N\}$. The layers that are never selected by any subsequent layers are averaged and sent to a softmax head, or to higher recurrent layers.

Parameters Initialization. Our controller's parameters θ are initialized uniformly in $[-0.1, 0.1]$. We find that for Penn Treebank, ENAS quite insensitive to its initialization than for CIFAR-10. Meanwhile, the shared parameters ω are initialized uniformly in $[-0.025, 0.025]$ during architecture search, and $[-0.04, 0.04]$ when we train a fixed architecture recommended by the controller.

Stabilizing the Updates of ω. To stabilize the updates of ω, during the architectures search phase, a layer of batch normalization (Ioffe & Szegedy, 2015) is added immediately after the average of these layers, before the average are sent out of the cell as its output. When a fixed cell is sampled by the controller, we find that we can remove the batch normalization layer without any loss in performance.

B. Details on CIFAR-10 Experiments

We find the following tricks crucial for achieving good performance with ENAS. Standard NAS (Zoph & Le, 2017; Zoph et al., 2018) rely on these and other tricks as well.

Structure of Convolutional Layers. Each convolution in our model is applied in the order of relu-conv-batchnorm (Ioffe & Szegedy, 2015; He et al., 2016b). Additionally, in our micro search space, each depthwise separable convolution is applied twice (Zoph et al., 2018).

Stabilizing Stochastic Skip Connections. If a layer receives skip connections from multiple layers before it, then these layers' outputs are concatenated in their depth dimension, and then a convolution of filter size 1×1 (followed by a batch normalization layer and a ReLU layer) is performed to ensure that the number of output channels does not change between different architectures. When a fixed architecture is sampled, we find that one can remove these batch normalization layers to save computing time and parameters of the final model, without sacrificing significant performance.

Global Average Pooling. After the final convolutional layer, we average all the activations of each channel and then pass them to the Softmax layer. This trick was introduced by (Lin et al., 2013), with the purpose of reducing the number of parameters in the dense connection to the Softmax layer to avoid overfitting.

The last two tricks are extremely important, since the gradient updates of the shared parameters ω, as described in Eqn 1, have very high variance. In fact, we find that without these two tricks, the training of ENAS is very unstable.