RNN Architecture Learning with Sparse Regularization

Jesse Dodge♣ **Roy Schwartz**♠♢ **Hao Peng**♠ **Noah A. Smith**♠♢
♣Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA, USA
♢Allen Institute for Artificial Intelligence, Seattle, WA, USA
♠Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA, USA
jessed@cs.cmu.edu roys@allenai.org {hapeng,nasmith}@cs.washington.edu

Abstract

Neural models for NLP typically use large numbers of parameters to reach state-of-the-art performance, which can lead to excessive memory usage and increased runtime. We present a structure learning method for learning sparse, parameter-efficient NLP models. Our method applies group lasso to rational RNNs (Peng et al., 2018), a family of models that is closely connected to weighted finite-state automata (WFSAs). We take advantage of rational RNNs' natural grouping of the weights, so the group lasso penalty directly removes WFSA states, substantially reducing the number of parameters in the model. Our experiments on a number of sentiment analysis datasets, using both GloVe and BERT embeddings, show that our approach learns neural structures which have fewer parameters without sacrificing performance relative to parameter-rich baselines. Our method also highlights the interpretable properties of rational RNNs. We show that sparsifying such models makes them easier to visualize, and we present models that rely exclusively on as few as three WFSAs after pruning more than 90% of the weights. We publicly release our code.[1]

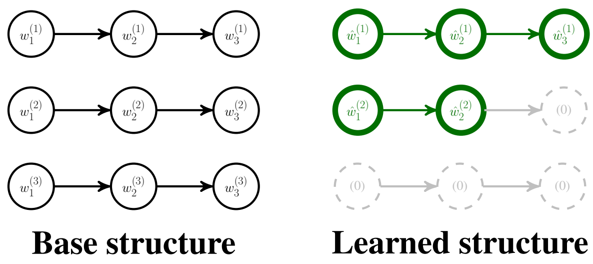

Figure 1: Our approach learns a sparse structure (right hand side) of a base rational RNN (left hand side) where each hidden unit corresponds to a WFSA (in this example, three hidden units, represented by the three rows). Grayed-out, dashed states are removed from the model, while retained states are marked in **bold green**.

1 Introduction

State-of-the-art neural models for NLP are heavily parameterized, requiring hundreds of millions (Devlin et al., 2019) and even billions (Radford et al., 2019) of parameters. While over-parameterized models can sometimes be easier to train (Livni et al., 2014), they may also introduce memory problems on small devices and lead to increased carbon emission (Strubell et al., 2019; Schwartz et al., 2019).

In feature-based NLP, structured-sparse regularization, in particular the group lasso (Yuan and Lin, 2006), has been proposed as a method to reduce model size while preserving performance (Martins et al., 2011). But, in neural NLP, some of the most widely used models—LSTMs (Hochreiter and Schmidhuber, 1997) and GRUs (Cho et al., 2014)—do not have an obvious, intuitive notion of "structure" in their parameters (other than, perhaps, division into layers), so the use of structured sparsity at first may appear incongruous.

In this paper we show that group lasso can be successfully applied to neural NLP models. We focus on a family of neural models for which the hidden state exhibits a natural structure: rational RNNs (Peng et al., 2018). In a rational RNN, the value of each hidden dimension is the score of a weighted finite-state automaton (WFSA) on (a prefix of) the input vector sequence. This property offers a natural grouping of the transition function parameters for each WFSA. As shown by Schwartz et al. (2018) and Peng et al. (2018), a variety of state-of-the-art neural architectures are rational (Lei et al., 2017; Bradbury et al., 2017; Foerster et al., 2017, *inter alia*), so learning parameter-efficient rational RNNs is of practical value. We

[1] https://github.com/dodgejesse/sparsifying_regularizers_for_RRNNs

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 1179–1184,
Hong Kong, China, November 3–7, 2019.

also take advantage of the natural interpretation of rational RNNs as "soft" patterns (Schwartz et al., 2018).

We apply a group lasso penalty to the WFSA parameters of rational RNNs during training, where each group is comprised of the parameters associated with one state in one WFSA (Fig. 1; §2). This penalty pushes the parameters in some groups to zero, effectively eliminating them, and making the WFSA smaller. When all of the states for a given WFSA are eliminated, the WFSA is removed entirely, so this approach can be viewed as learning the number of WFSAs (i.e., the RNN hidden dimension) as well as their size. We then retain the sparse structure, which results in a much smaller model in terms of parameters.

We experiment with four text classification benchmarks (§3), using both GloVe and BERT embeddings. As we increase the regularization strength, we end up with smaller models. These models have a better tradeoff between the number of parameters and model performance compared to setting the number of WFSAs and their lengths by hand or using hyperparameter search. In almost all cases, our approach results in models with fewer parameters and similar or better performance compared to our baselines.

In contrast to neural architecture search (Jozefowicz et al., 2015; Zoph and Le, 2017), which can take several GPU years to learn an appropriate neural architecture, our approach requires only two training runs: one to learn the structure, and the other to estimate its parameters. Other approaches either ignore the structure of the model and only look at the value of individual weights (Liu et al., 2019; LeCun et al., 1990; Lee et al., 2019; Frankle and Carbin, 2019) or only use high-level structures like the number of layers of the network (Wen et al., 2016; Scardapane et al., 2017; Gordon et al., 2018).

Finally, our approach touches on another appealing property of rational RNNs—their interpretability. Each WFSA captures a "soft" version of patterns like "such a great X", and can be visualized as such (Schwartz et al., 2018). By retaining a small number of WFSAs, model structures learned using our method can be visualized succinctly. In §4 we show that some of our sentiment analysis models rely exclusively on as few as *three* WFSAs.[2]

[2]That is, a rational RNN with hidden size 3.

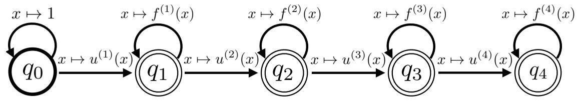

Figure 2: A 4-gram WFSA, from which we derive the rational RNN (§2). The rational RNN's hidden states correspond to a set of WFSAs, each separately parameterized. We apply group lasso to each WFSA.

2 Method

We describe the proposed method. At a high level, we follow the standard practice for using ℓ_1 regularization for sparsification (Wen et al., 2016):

1. Fit a model on the training data, with the group lasso regularizer added to the loss during training (the parameters associated with one state comprise one group).
2. After convergence, eliminate the states whose parameters are zero.
3. Finetune the resulting, smaller model, by minimizing the unregularized loss with respect to its parameters.

In this work, we assume a single layer rational RNN, but our approach is equally applicable to multi-layer models. For clarity of the discussion, we start with a one-dimensional rational RNN (i.e., one based on a single WFSA only). We then generalize to the d-dimensional case (computing the scores of d WFSAs in parallel).

Rational recurrent networks Following Peng et al. (2018), we parameterize the transition functions of WFSAs with neural networks, such that each transition (main path or self loop) defines a weighted function over the input word vector. We consider a 5-state WFSA, diagrammed in Fig. 2.

A path starts at q_0; at least four tokens must be consumed to reach q_4, and in this sense it captures 4-gram "soft" patterns (Peng et al., 2018; Schwartz et al., 2018). In addition to q_4, we also designate q_1, q_2, and q_3 as final states, allowing for the interpolation between patterns of different lengths.[3] The self-loop transitions over q_1, q_2, q_3, and q_4 aim to allow, but downweight, nonconsecutive patterns, as the self-loop transition functions yield values between 0 and 1 (using a sigmoid function). The recurrent function is equivalent to applying the Forward dynamic programming algorithm (Baum and Petrie, 1966).

[3]We found this to be more stable than using only q_4.

Promoting sparsity with group lasso We aim to learn a sparse rational model with fewer WFSA states. This can be achieved by penalizing the parameters associated with a given state, specifically the parameters associated with *entering* that state, by a transition from another state or a self-loop on that state. For example, the parameters of the WFSA in Fig. 2 (excluding the word embedding parameters) are assigned to four nonoverlapping groups, one for each non-starting state.

During training, the regularization term will push all parameters toward zero, and some will converge close to zero.[4] After convergence, we remove groups for which the ℓ_2 norm falls below ϵ.[5] The resulting smaller model is then finetuned by continuing training without regularizing. With our linear-structured WFSA, zeroing out the group associated with a state in the middle effectively makes later states inaccessible. While our approach offers no guarantee to remove states from the end first (thus leaving no unreachable states), in our experiments it always did so.

d-dimensional Case To construct a rational RNN with d WFSAs (a d-dimensional model), we stack d one-dimensional models, each of them separately parameterized. The parameters of a d-dimensional rational model derived from the WFSA in Fig. 2 are organized into $4d$ groups, four for each dimension. Since there is no direct interaction between different dimensions (e.g., through an affine transformation), group lasso sparsifies each dimension/WFSA independently. Hence the resulting rational RNN can consist of WFSAs of different sizes, the number of which could be smaller than d if any of the WFSAs have all states eliminated.

One can treat the numbers and sizes of WFSAs as hyperparameters (Oncina et al., 1993; Ron et al., 1994; de la Higuera, 2010; Schwartz et al., 2018). By eliminating states from WFSAs with group lasso, we learn the WFSA structure *while* fitting the models' parameters, reducing the number of training cycles by reducing the number of tunable hyperparameters.

[4]There are optimization methods that achieve "strong" sparsity (Parikh and Boyd, 2013), where some parameters are exactly set to zero during training. Recent work has shown these approaches can converge in nonconvex settings (Reddi et al., 2016), but our experiments found them to be unstable.

[5]We use 0.1. This threshold was lightly tuned in preliminary experiments on the validation set and found to reliably remove those parameters which converged around zero without removing others.

3 Experiments

We run sentiment analysis experiments. We train the rational RNN models (§2) with group lasso regularization, using increasingly large regularization strengths, resulting in increasingly compact models. As the goal of our experiments is to demonstrate the ability of our approach to reduce the number of parameters, we only consider rational baselines: the same rational RNNs trained without group lasso.[6] We manually tune the number and sizes of the baselines WFSAs, and then compare the tradeoff curve between model size and accuracy. We describe our experiments below. For more details, see Appendix A.

Data We experiment with the Amazon reviews binary sentiment classification dataset (Blitzer et al., 2007), composed of 22 product categories. We examine the standard dataset (**original_mix**) comprised of a mixture of data from the different categories (Johnson and Zhang, 2015).[7] We also examine three of the largest individual categories as separate datasets (**kitchen**, **dvd**, and **books**), following Johnson and Zhang (2015). The three category datasets do *not* overlap with each other (though they do with **original_mix**), and are significantly different in size (see Appendix A), so we can see how our approach behaves with different amounts of training data.

Implementation details To classify text, we concatenate the scores computed by each WFSA, then feed this d-dimensional vector of scores into a linear binary classifier. We use log loss. We experiment with both type-level word embeddings (GloVe.6B.300d; Pennington et al., 2014) and contextual embeddings (BERT large; Devlin et al., 2019).[8] In both cases, we keep the embeddings fixed, so the vast majority of the learnable parameters are in the WFSAs. We train models using GloVe embeddings on all datasets. Due to memory constraints we evaluate BERT embeddings (frozen, not fine-tuned) only on the smallest

[6]Rational RNNs have shown strong performance on the dataset we experiment with: a 2-layer rational model with between 100–300 hidden units obtained 92.7% classification accuracy, substantially outperforming an LSTM baseline (Peng et al., 2018). The results of our models, which are single-layered and capped at 24 hidden units, are not directly comparable to these baselines, but are still within two points of the best result from that paper.

[7]`http://riejohnson.com/cnn_data.html`

[8]`https://github.com/huggingface/pytorch-pretrained-BERT`

dataset (**kitchen**).

Baselines As baselines, we train five versions of each rational architecture without group lasso, using the same number of WFSAs as our regularized models (24 for GloVe, 12 for BERT). Four of the baselines each use the same number of transitions for all WFSAs (1, 2, 3, and 4, corresponding to 2–5 states, and to 24, 48, 72, and 96 total transitions). The fifth baseline has an equal mix of all lengths (6 WFSAs of each size for GloVe, leading to 60 total transitions, and 3 WFSAs of each size for BERT, leading to 30 total transitions).

Each transition in our model is independently parameterized, so the total number of transitions linearly controls the number of learnable parameters (in addition to the parameters in the embedding layer).

Results Fig. 3 shows our classification test accuracy as a function of the total number of WFSA transitions in the model. We first notice that, as expected, the performance of our unregularized baselines improves as models are trained with more transitions (i.e., more parameters).

Compared to the baselines, training with group lasso provides a better tradeoff between performance and number of transitions. In particular, our heavily regularized models perform substantially better than the unigram baselines, gaining between 1–2% absolute improvements in four out of five cases. As our regularization strength decreases, we naturally gain less compared to our baselines, although still similar or better than the best baselines in four out of five cases.

4 Visualization

Using our method with a high regularization strength, the resulting sparse structures often contain only a handful of WFSAs. In such cases, building on the interpretability of individual WFSAs, we are able visualize every hidden unit, i.e., the entire model. To visualize a single WFSA $\mathcal{B}$, we follow Schwartz et al. (2018) and compute the score of $\mathcal{B}$ on every phrase in the training corpus. We then select the top and bottom scoring phrases for $\mathcal{B}$,[9] and get a prototype-like description of the pattern representing $\mathcal{B}$.[10]

[9]As each WFSA score is used as a feature that is fed to a linear classifier, negative scores are also meaningful.

[10] While the WFSA scores are the sum of all paths deriving a document (plus-times semiring), here we search for the max (or min) scoring one. Despite the mismatch, a WFSA scores every possible path, and thus the max/min scoring path selection is still valid. As our examples show, many of these extracted paths are meaningful.

		transition$_1$	transition$_2$	transition$_3$
Patt. 1	Top	not	worth	the time$_{SL}$ </s>
		not	worth	the 30$_{SL}$ </s>
		not	worth	it$_{SL}$ </s>
		not	worth	it$_{SL}$ </s>
	Bottom	extremely	pleased	$\ldots_{SL}$ </s>
		highly	pleased	$\ldots_{SL}$ </s>
		extremely	pleased	$\ldots_{SL}$ </s>
		extremely	pleased	$\ldots_{SL}$ </s>
Patt. 2	Top	miserable	</s>	
		miserable	$\ldots_{SL}$ </s>	
		miserable	$\ldots_{SL}$ </s>	
		returned	</s>	
	Bottom	superb	</s>	
		superb	</s>	
		superb	</s>	
		superb	choice$_{SL}$ </s>	
Patt. 3	Top	bad	$\ldots_{SL}$ ltd	$\ldots_{SL}$ buyer
		bad	$\ldots_{SL}$ ltd	$\ldots_{SL}$ buyer
		horrible	$\ldots_{SL}$ hl4040cn	$\ldots_{SL}$ expensive
		left	$\ldots_{SL}$ ltd	$\ldots_{SL}$ lens
	Bottom	favorite	$\ldots_{SL}$ ltd	$\ldots_{SL}$ lens
		really	$\ldots_{SL}$ ltd	$\ldots_{SL}$ buyer
		really	$\ldots_{SL}$ ltd	$\ldots_{SL}$ buyer
		best	$\ldots_{SL}$ hl4040cn	$\ldots_{SL}$ expensive

Table 1: Visualization of a sparse rational RNN trained on **original_mix** containing only 3 WFSAs. For each WFSA (i.e., pattern), we show the 4 top and bottom scoring phrases in the training corpus with this WFSA. Each column represents one main-path transition, plus potential self-loops preceding it (marked like this$_{SL}$). "$\ldots_{SL}$" marks more than 2 self loops. "</s>" marks an end-of-document token.

Table 1 visualizes a sparse rational RNN trained on **original_mix** with only *three* WFSAs, (8 main-path transitions in total).[11] The table shows that looking at the top scores of each WFSA, two of the patterns respectively capture the phrases "*not worth X </s>*" and "*miserable/returned X </s>*". Pattern 3 is not as coherent, but most examples do contain sentiment-bearing words such as *bad, horrible,* or *best*. This might be the result of the tuning process of the sparse rational structure simply learning a collection of words, rather than coherant phrases. As a result, this WFSA is treated as a unigram pattern rather than a trigram. The lowest scoring phrases show a similar trend. Appendix B shows the same visualization for another sparse rational RNN containing only four WFSAs and 11 main-path transitions, trained with BERT embeddings.

We observe another interesting trend: two of the three patterns prefer expressions that appear near the end of the document. This could result from the nature of the datasets (e.g., many reviews end

[11]The test performance of this model is 88%, 0.6% absolute below the average of the five models reported in Fig. 3.

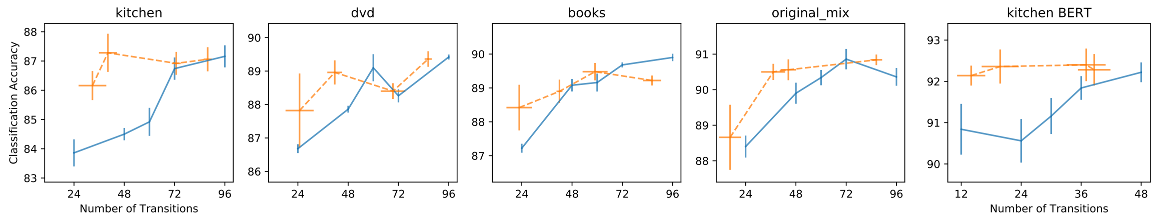

Figure 3: Text classification with GloVe embeddings (four leftmost graphs) and BERT (rightmost): accuracy (y-axis) vs. number of WFSA transitions (x-axis). **Higher** and to the **left** is better. Our method (dashed orange line, varying regularization strength) provides a better tradeoff than the baseline (solid blue line, directly varying the number of transitions). Vertical lines encode one standard deviation for accuracy, while horizontal lines encode one standard deviation in the number of transitions (applicable only to our method).

with a summary, containing important sentiment information), and/or our rational models' recency preference. More specifically, the first self loop has weight 1, and hence the model is not penalized for taking self loops before the match; in contrast, the weights of the last self loop take values in $(0, 1)$ due to the sigmoid, forcing a penalty for earlier phrase matches.[12]

5 Conclusion

We presented a method for learning parameter-efficient RNNs. Our method applies group lasso regularization on rational RNNs, which are strongly connected to weighted finite-state automata, and thus amenable to learning with structured sparsity. Our experiments on four text classification datasets, using both GloVe and BERT embeddings, show that our sparse models provide a better performance/model size tradeoff. We hope our method will facilitate the development of "thin" NLP models, that are faster, consume less memory, and are interpretable (Schwartz et al., 2019).

Acknowledgments

This work was completed while the first author was an intern at the Allen Institute for Artificial Intelligence. The authors thank Pradeep Dasigi, Gabriel Stanovsky, and Elad Eban for their discussion. In addition, the authors thank the members of the Noahs ARK group at the University of Washington, the researchers at the Allen Institute for Artificial Intelligence, and the anonymous reviewers for their valuable feedback. This work was supported in part by a hardware gift from NVIDIA Corporation.

[12]Changing this behavior could be easily done by fixing the final self-loop to 1 as well.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

Leonard E. Baum and Ted Petrie. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *The Annals of Mathematical Statistics*, 37(6).

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, Bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL*.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2017. Quasi-recurrent neural network. In *Proc. of ICLR*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of EMNLP*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL*.

Jakob N. Foerster, Justin Gilmer, Jan Chorowski, Jascha Sohl-Dickstein, and David Sussillo. 2017. Intelligible language modeling with input switched affine networks. In *Proc. of ICML*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *Proc. of ICLR*.

Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. 2018. MorphNet: Fast & simple resource-constrained structure learning of deep networks. In *Proc. of CVPR*.

Colin de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

Rie Johnson and Tong Zhang. 2015. Effective use of word order for text categorization with convolutional neural networks. In *Proc. of NAACL*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proc. of ICML*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. of ICLR*.

Yann LeCun, John S. Denker, and Sara A. Solla. 1990. Optimal brain damage. In *NeurIPS*.

Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. 2019. SNIP: Single-shot network pruning based on connection sensitivity. In *Proc. of ICLR*.

Tao Lei, Yu Zhang, and Yoav Artzi. 2017. Training RNNs as fast as CNNs. arXiv:1709.02755.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2019. DARTS: Differentiable architecture search. In *Proc. of ICLR*.

Roi Livni, Shai Shalev-Shwartz, and Ohad Shamir. 2014. On the computational efficiency of training neural networks. In *NeurIPS*.

André F. T. Martins, Noah A. Smith, Mario Figueiredo, and Pedro Aguiar. 2011. Structured sparsity in structured prediction. In *Proc. of EMNLP*.

José Oncina, Pedro García, and Enrique Vidal. 1993. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15:448–458.

Neal Parikh and Stephen P. Boyd. 2013. *Proximal Algorithms*. Foundations and Trends in Optimization.

Hao Peng, Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. Rational recurrences. In *Proc. of EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Sashank J. Reddi, Suvrit Sra, Barnabs Pczos, and Alexander J. Smola. 2016. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In *NeurIPS*.

Dana Ron, Yoram Singer, and Naftali Tishby. 1994. Learning probabilistic automata with variable memory length. In *Proc. of COLT*.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. 2017. Group sparse regularization for deep neural networks. *Neurocomputing*, 241.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. arXiv:1907.10597.

Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. SoPa: Bridging CNNs, RNNs, and weighted finite-state machines. In *Proc. of ACL*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proc. of ACL*.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *NeurIPS*.

Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1).

Barret Zoph and Quoc V Le. 2017. Neural architecture search with reinforcement learning. In *Proc of ICLR*.