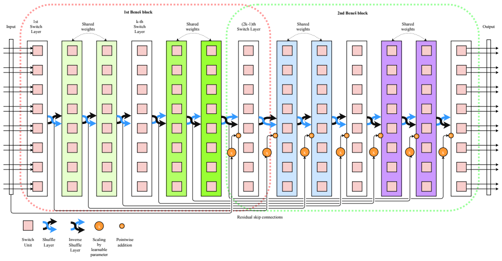
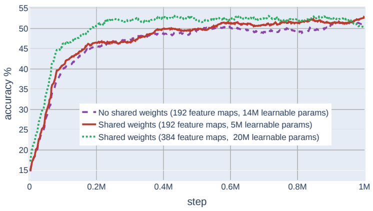
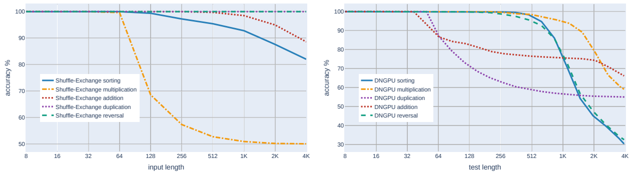
Thank you for your valuable comments and suggestions. Please see the following drawing of the complete model(input length 16, $k = 4$). Weight sharing is shown with colored layers. White layers have unique weights. You are right that for a sequence of length $2^k$, [x1, x2, x3, x4, . . . .] adjacent pairs [(x1, x2), (x3, x4), . . . ] are given to each Switch Unit. There are $2^{k-1}$ pairs with shared weights per layer. We designed the sharing to be minimal such that the model generalizes to longer inputs. We found that this schema helps to reduce the parameter count without accuracy loss also on tasks not requiring generalization, see the Lambada figure. It could be possible to reduce more parameters by sharing more weights. We will analyze this in the final paper.



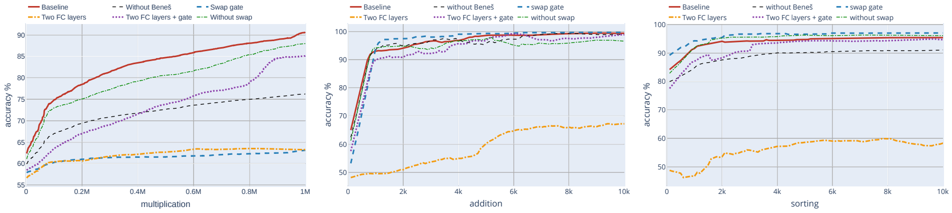


Effect of weight sharing on Lambada

Our architecture compared to Diagonal Neural GPU has competitive generalization on longer test sequences. Both models were trained for 40k steps on length up to 64 symbols.



The main purpose of the update gate is to stabilize the gradient flow through the layers(similar to LSTM and GRU). It does not increase the expressive power of the unit much since its logic can be simulated by the other parts of the unit. See the following ablation experiments. They display multiplication accuracy trained and tested on length 128, addition and sorting accuracy trained on 64, tested on 512. Three additional ablations are provided:(swap gate) an additional swap gate is introduced; (Two FC layers) the entire Switch Unit is replaced with two fully connected layers with ReLU and twice the number of feature maps in the middle;(Two FC layers+gate) the part involving reset gates is replaced with two FC layers. ReLU on both FC layers does not work well. To maximize the variety of solvable tasks, the hardest ones should be given more impact in ablation study. We have used about 15 tasks in total for tuning our model.



This architecture can learn positional information by itself. Assume that the input has a marked position(end-of-line marker, for example) and consider the binary tree of paths from it to nodes at depth $\log(n)$. Each leaf of this tree can be uniquely labeled according to left-or-right choices on the path connecting it to the root.

We would like to note that learning fast algorithms is a considerably harder task than learning slow algorithms(you may try challenging your students to come up from scratch with a log-depth circuit to add long binary numbers). Fast algorithms are often considerably more complex and, remarkably, that our architecture can learn them. Also, it is natural that the more complex algorithms learned by our model generalize worse than possibly simpler ones learned by the Neural GPU. Therefore we consider our work a significant contribution in the field of algorithm learning, regardless that we do not obtain better accuracies.

We agree that showing the benefits of our architecture on a real-wold task with long sequences is an important future work.

We will include the analysis given here and take care of the other review suggestions in the final paper.