Dataset	Initial Embedding Size ($d_{initial\_embed}$)	Hidden Size (d_h or d_{embed})	Input Dropout	Output Dropout	Hidden Dropout
Logical Infer.	200	200	0.1	0.3	0.1
ListOps	128	128	0.3	0.2	0.1
SST2	300	300	0.3	0.2	0.4
SST5	300	300	0.3	0.2	0.4
SNLI	300	300	0.4	0.1	0.1
MNLI	300	300	0.4	0.1	0.1

Table 9. Hyperparameter details for CRvNN.

A. Architecture Details

For every task used in our experiments we use an initial affine transformation where the initial embeddings of size $d_{initial\_embed}$ are transformed into the size d_{embed}. Typically, we set d_{embed} as d_h. See Table 9 for their values.

We treat the last representation in the sequence after being processed by CRvNN as the sentence encoding constructed by CRvNN.

For classification tasks, we classify the sentence encoding by transforming it into logits for the classes after passing it through a series of affine layers (typically, 1 or 2). Intermediate layers have d_h neurons where d_h is also the dimension of the sentence encoding.

For inference tasks (requiring sequence-pair comparison), like logical inference or SNLI and MNLI, we use a Siamese framework. Concretely, we first encode (using the same encoder with same parameters) both the premise and hypothesis (separately) into sentence vectors, say, s_1 and s_2 respectively (both with d_h dimensions). Then, we construct a classification feature vector o as:

$$o = [s_1; s_2; |s_1 - s_2|; s_1 \odot s_2] \tag{19}$$

Here, $[;]$ indicates concatenation. We send o to a Multi Layer Perceptron (MLP) to classify the sequence relationship. The final layer activation is $Softmax$, but if there are intermediate MLP layers, we use $GeLU$ for them. We use a dropout (hidden dropout) in the gated recursive cell (after its first affine transformation). We use a dropout (input dropout) on the input just before sending it to CRvNN. We use another dropout (output dropout) in between the final MLP layers. All our models were trained on AWS p3.2× instance (Nvidia v100).

B. Implementation Details

For all experiments, as an optimizer, we use Ranger (Wright, 2019) or Rectified (Liu et al., 2020) Adam (Kingma & Ba, 2015) with lookahead ($k = 5, \alpha = 0.8$) (Zhang et al., 2019) and decorrelated weight decay (Loshchilov & Hutter, 2019) ($1e - 2$) with a learning rate of $1e - 3$. We used GloVe (300 dimensions, 840B) (Pennington et al., 2014) as untrainable embeddings for natural language data. We set the cell size (d_{cell} as referred in §3.3.3) as $4 \cdot d_h$ (d_h is the hidden size). We set the size of transition features (d_s as referred in §3.4.2) as 64. For convolution in the decision function, we always use a window size of 5. For halt penalty in §3.4.3, we set γ as 0.01. Generally, we use a two-layered MLP on the sentence encoding from CRvNN. However, on ListOps, we used a single-layer. We use a batch size of 128 for all tasks. We describe other hyperparameters of CRvNN in Table 9. We cut the learning rate by half if the validation loss does not decrease for 3 contiguous epochs.

C. Hyperparameter Search

On ListOps, we tune different dropouts among $\{0.1, 0.2, 0.3, 0.4\}$ separately using grid search (we ran for 10 epochs and $50,000$ subsamples). For the Logical Inference task (length generalization task), we tune the different dropouts among $\{0.1, 0.2, 0.3\}$ for 7 epochs per trial using grid search. We use the same hyperparamters for systematicity splits. For SST5, we tune the dropouts in $\{0.2, 0.3, 0.4\}$ for 3 epochs. For SNLI, we tune the different dropouts among $\{0.1, 0.2, 0.3, 0.4\}$ for 5 epochs, using a sub-sample of $100K$ examples, and for a maximum of 20 trials using Tree of Parzen Estimators (TPE) (Bergstra et al., 2011). We use Hyperopt (Bergstra et al., 2013) for hyperparameter tuning. For other components we mostly use similar hyperparameters as Shen et al. (2019a) or default settings. We share the hyperparameters found for SST5 with SST2 and we also share the hyperparameters found for SNLI with MNLI.

D. Datasets

For all datasets, we use the standard splits as used by prior work. For training efficiency, we filter out training samples of sequence size > 150 from MNLI. We filter out training samples with sequence length > 100 from ListOps. We use the $90K$ sample version of ListOps similar to prior work.