A NAQANet Details and Numeracy Requirements

Aside from typical paragraph spans, NAQANet can also output spans in the question, counts from 0–9, and arithmetic expressions. The arithmetic expressions are generated by first extracting all the numbers from the passage and then assigning a coefficient in {-1,0,1} to each number. The final answer is calculated as the sum of each number using its associated coefficient.

Addition and subtraction questions are present in DROP but do not require numeracy for NAQANet: the model can output the correct coefficients based on the surrounding context words without understanding number magnitude. For example, consider a paragraph containing, "Superbowl XXXI occurred in 1997 and Superbowl XXXVII occurred in 2003", and a question "How many years after Superbowl XXXI did Superbowl XXXVII occur?". The model can output coefficient +1 on "2003" and -1 on "1997" and answer correctly without understanding the magnitude of the two years. Counting questions also do not require reading or manipulating numbers in text: the model only needs to output the correct 10-way classification value.

B Left-padding for Enhancing Char-CNN Numeracy

In modern deep learning and NLP frameworks, pad tokens are often appended to the right of inputs to ensure equal lengths inside each batch (for efficiency). When using *character*-level convolutions, padding must be added to the characters of a word when its length is smaller than the minimum kernel width (for correctness). Naturally, the right padding implementation is re-used, e.g., the two numbers "11" and "110" are represented as:

$$\begin{aligned} 11 &= 11\emptyset\emptyset\emptyset \\ 110 &= 110\emptyset\emptyset, \end{aligned}$$

where $\emptyset$ is the pad token. However, right padding is detrimental to learning numeracy: the "hundreds" and "tens" digits are now in the same position in the two tokens. We found that left padding numbers substantially decreases the generalization error and convergence rate for Char-CNN models trained on the synthetic tasks.[6]

[6]Padding on the left and right via SAME convolutions also mitigates this issue.

C Training Details for Probing

We create training/test splits for the addition task in the following manner. We first shuffle and split an integer range, putting 80% into train and 20% into test. Next, we enumerate all possible pairs of numbers in the two respective splits. When using large ranges such as [0,999], we sub-sample a random 10% of the training and test pairs.

For the list maximum task, we first shuffle and split the data, putting 80% into a training pool of numbers and 20% into a test pool. In initial experiments, we created the lists of five numbers by sampling uniformly over the training/test pool. However, as the random samples will likely be spread out over the range, the numbers are easy to distinguish. We instead create 100,000 training examples and 10,000 examples in the following manner. We first sample a random integer from the training or test pool. Next, we sample from a Gaussian with mean zero and variance equal to 0.01 times the total size of the range. Finally, we add the random Gaussian sample to the random integer, and round to the nearest value in the pool. This forces the numbers to be nearby.

D Additional Synthetic Results

D.1 Linear Regression Accuracies

Table 8 provides accuracies for number decoding using linear regression (interpolation setting).

Interpolation	**Decoding (RMSE)**		
Integer Range	[0,50]	[-50,50]	[0,999]
Random	13.86	29.46	275.41
Word2Vec	4.15	8.93	29.04
GloVe	3.21	5.76	23.27
ELMo	1.20	2.89	21.53
BERT	3.23	7.86	64.42

Table 8: *Number Decoding interpolation accuracy with linear regression.* Linear regression is competitive to the fully connected probe for smaller numbers.

D.2 Variances on Digit Form Results

Table 9 shows the mean and standard deviation for the synthetic tasks using five random shuffles.

D.3 Float Values

We test floats with one decimal point. We follow the setup for the list maximum task (Appendix C) with a minor modification. For 50% of the training/test lists, we reuse the same integer five times but sample a different random value to put after the decimal point. For example, 50% of the lists are of the form: [15.3, 15.6, 15.1, 15.8, 15.2] (the same base integer is repeated with a different decimal), and 50% are random integers with a random digit after the decimal: [11.7, 16.4, 9.3, 7.9, 13.3]. This forces the model to consider the numbers on both the left and the right of the decimal.

D.4 Word Form Results

Table 10 presents the results using word-forms. We do not use numbers larger than 100 as they consist of multiple words.

E Automatically Modifying DROP Paragraphs

Modifying a DROP paragraph automatically is challenging as it may change the answer to the question if done incorrectly. We also cannot modify the answer because many DROP questions have count answers. To guarantee that the original annotated answer can still be used, we perform the number transformation in the following manner. We keep the original passage text the same, but, we modify the model's internal embeddings for the numbers directly. In other words, the model uses exactly the same embeddings for the original text except for the modified numbers. The model then needs to find the correct index (e.g. the index of the correct span, or the index of the correct number) given these modified embeddings.

F Extrapolation with Data Augmentation

For each superlative and comparative example, we modify the numbers in its paragraph using the *Add* and *Multiply* techniques mentioned in Section 2.5. We first multiply the paragraph's numbers by a random integer from [1, 10], and then add another random integer from [0, 20]. We train NAQANet on the original paragraph and an additional modified version for all training examples. We use a single additional paragraph for computational efficiency; augmenting the data with more modified paragraphs may further improve results.

We test NAQANet on the original validation set, as well as a *Bigger* validation set. We created the Bigger validation set by multiplying each paragraph's numbers by a random integer from [11,20] and then adding a random value from [21,40]. Note that this range is larger than the one used for data augmentation.

Table 11 shows the results of NAQANet trained with data augmentation. Data augmentation provide small gains on the original superlative and comparative question subset, and significant improvements on the *Bigger* version (it doubles the model's F1 score for superlative questions).

Interpolation	**List Maximum (5-classes)**			**Decoding (RMSE)**			**Addition (RMSE)**		
Integer Range	[0,99]	[0,999]	[0,9999]	[0,99]	[0,999]	[0,9999]	[0,99]	[0,999]	[0,9999]
Random Vectors	0.16 ± 0.03	0.23 ± 0.12	0.21 ± 0.02	29.86 ± 4.44	292.88 ± 13.48	2882.62 ± 71.68	42.03 ± 7.79	410.33 ± 16.05	4389.39 ± 310.91
Untrained CNN	0.97 ± 0.01	0.87 ± 0.02	0.84 ± 0.03	2.64 ± 0.68	9.67 ± 1.17	44.40 ± 4.98	1.41 ± 0.05	14.43 ± 1.90	69.14 ± 21.54
Untrained LSTM	0.70 ± 0.05	0.66 ± 0.03	0.55 ± 0.02	7.61 ± 1.33	46.5 ± 5.65	210.34 ± 9.91	5.11 ± 2.1	45.69 ± 3.78	510.19 ± 31.45
Value Embedding	0.99 ± 0.01	0.88 ± 0.04	0.68 ± 0.06	1.20 ± 0.76	11.23 ± 1.35	275.5 ± 135.59	0.30 ± 0.01	15.98 ± 3.62	654.33 ± 69.91
Pre-trained									
Word2Vec	0.90 ± 0.03	0.78 ± 0.05	0.71 ± 0.03	2.34 ± 1.44	18.77 ± 4.40	333.47 ± 14.83	0.75 ± 0.41	21.23 ± 3.53	210.07 ± 30.56
GloVe	0.90 ± 0.02	0.78 ± 0.04	0.72 ± 0.02	2.23 ± 1.26	13.77 ± 3.23	174.21 ± 31.91	0.80 ± 0.30	16.51 ± 1.17	180.31 ± 18.97
ELMo	0.98 ± 0.04	0.88 ± 0.02	0.76 ± 0.04	2.35 ± 0.65	13.48 ± 2.19	62.2 ± 5.04	0.94 ± 0.35	15.50 ± 2.49	45.71 ± 18.69
BERT	0.95 ± 0.02	0.62 ± 0.01	0.52 ± 0.07	3.21 ± 0.31	29.00 ± 7.93	431.78 ± 30.27	4.56 ± 1.76	67.81 ± 30.34	454.78 ± 91.07
Learned									
Char-CNN	0.97 ± 0.02	0.93 ± 0.01	0.88 ± 0.02	2.50 ± 0.49	4.92 ± 0.97	11.57 ± 1.34	1.19 ± 0.18	7.75 ± 1.85	15.09 ± 0.90
Char-LSTM	0.98 ± 0.02	0.92 ± 0.02	0.76 ± 0.03	2.55 ± 1.81	8.65 ± 1.11	18.33 ± 3.21	1.21 ± 0.10	15.11 ± 1.87	25.37 ± 3.40
DROP-trained									
NAQANet	0.91 ± 0.01	0.81 ± 0.02	0.72 ± 0.03	2.99 ± 1.11	14.19 ± 3.75	62.17 ± 4.31	1.11 ± 0.41	11.33 ± 1.67	90.01 ± 17.88
- GloVe	0.88 ± 0.02	0.90 ± 0.03	0.82 ± 0.02	2.87 ± 0.83	5.34 ± 0.77	35.39 ± 4.32	1.45 ± 0.95	9.91 ± 1.45	60.70 ± 13.04

Table 9: Mean and standard deviation for Table 4 (interpolation tasks with integers).

Interpolation	**List Maximum (5-classes)**		**Decoding (MSE)**		**Addition (MSE)**	
Integer Range	[zero, fifty]	[zero,ninety-nine]	[zero, fifty]	[zero,ninety-nine]	[zero, fifty]	[zero,ninety-nine]
Random Vectors	0.16 ± 0.03	0.23 ± 0.12	15.43 ± 1.13	29.86 ± 4.44	23.14 ± 4.46	43.86 ± 1.41
Pre-trained						
Word2Vec	0.96 ± 0.05	0.91 ± 0.05	2.40 ± 0.64	3.94 ± 1.88	2.10 ± 0.31	4.10 ± 0.98
GloVe	0.93 ± 0.07	0.90 ± 0.03	3.02 ± 0.98	4.75 ± 1.59	3.33 ± 0.51	4.81 ± 1.00
ELMo	0.75 ± 0.04	0.82 ± 0.10	3.86 ± 0.92	7.31 ± 2.06	3.91 ± 0.55	6.97 ± 1.19
BERT	0.61 ± 0.11	0.66 ± 0.05	7.59 ± 1.37	13.55 ± 3.15	7.33 ± 1.05	12.91 ± 2.57
Learned						
Char-CNN	0.98 ± 0.01	0.99 ± 0.01	5.73 ± 0.93	7.85 ± 2.36	6.11 ± 0.55	7.11 ± 2.34
Char-LSTM	0.89 ± 0.06	0.86 ± 0.08	8.45 ± 4.97	7.31 ± 2.06	8.91 ± 3.11	9.51 ± 1.19
DROP-trained						
NAQANet	0.98 ± 0.01	0.97 ± 0.02	3.17 ± 1.05	4.31 ± 0.68	3.05 ± 0.87	3.95 ± 0.33
- GloVe	0.96 ± 0.03	0.97 ± 0.02	7.81 ± 1.34	9.43 ± 2.31	8.55 ± 0.88	10.01 ± 2.07

Table 10: *Interpolation task accuracies with word form (e.g., "twenty-five").* The model is trained on a randomly shuffled 80% of the *Integer Range* and tested on the remaining 20%. We show the mean and standard deviation for five random shuffles.

	Superlative		**Comparative**		**All Validation**	
	Original	Bigger	Original	Bigger	Original	Bigger
NAQANet	64.5 / 67.7	30.0 / 32.2	73.6 / 76.4	70.3 / 73.0	**46.2** / 49.2	38.7 / 41.4
+ Data Augmentation	**67.6 / 70.9**	**59.2 / 62.4**	**76.0 / 77.7**	**75.0 / 76.8**	46.1 / **49.3**	**42.8 / 45.8**

Table 11: Data augmentation improves NAQANet's interpolation and extrapolation results. We created the *Bigger* version of DROP by multiplying numbers in the passage by a random integer from [11, 20] and then adding a random integer from [21, 40]. Scores are shown in EM / F1 format.