DRAFT

Generalizable Neural Network for Learning Algebraic Operations Using Quantity Encoding and Abstract Representation of Numbers

Min Jean Cho
min_jean_cho@alumni.brown.edu

ABSTRACT

I propose a neural network, Q-Net, capable of generalizing basic math operations to unseen numbers. The key components of Q-Net are quantity encoding of numbers and abstract representation of quantities, which were inspired from children's algebra learning process and autoencoder's exploratory power of latent space. Q-Net is comprised of two modules: A-module and C-module. Numbers are first represented by quantity encodings and encoded numbers are fed into autoencoder-like A-module, where the encoder part of the A-module extracts the abstract representation of numbers. Algebraic operations are carried out by C-module using the learned abstract representation of numbers, and the abstract representation of resulting number is decoded to output number by the decoder part of A-module. Compare to baseline models, Q-Net showed significantly better performance for multiplication with unseen numbers.

INTRODUCTION

Humans have good generalization abilities. For example, children who have learned how to calculate '1+2' and '3+5' can later calculate '15 + 23' and '128 × 256.' They start out understanding the addition with small natural numbers and then later learn a general 'algorithm' which is quite explicitly a symbolic computation. Is it possible to develop an artificial intelligence (AI) that can have such generalization ability? If the AI initially trained with limited data for a specific task could easily expand its capability into more general tasks without additional training or modifying its initial architecture, it would be highly beneficial in many fields. This inquiry led me to formulate the present project as a starting point towards such a general AI. The specific goal of this work is to design a neural network (NN) capable of generalizing basic algebraic operations to unseen numbers.

I first hypothesized that there are two important components in human learning of algebraic operations: i) learning the meaning of numbers; ii) learning the meaning of algebraic operations. In human brain, numbers are abstract representations of quantities, and algebraic operations are processes in very abstract forms as well. The way of performing an operation such as '1+2' in human brain could be very different from a simple addition of two signals of

magnitude ‘1’ and ‘2’ in biological neural network. I speculate that human brain’s generalization ability could be based on a way human brain processes information in abstract forms and its ability to learn the general algorithm for complex symbolic computations could be founded on the understanding of the abstract meanings of quantities and algebraic operations. Based on this hypothesis, I developed a NN architecture, tentatively named Q-Net, capable of generalizing basic algebraic operations to unseen numbers. The key components of Q-Net are the *quantity encoding* ($\mathbf{x}_{quant}$) *of numbers* (x_{num}) and the *abstract representation* ($\mathbf{z}_x$) of the quantities (see Model Architecture for details). For the baseline model, I used NN that directly took numbers (x_{num}) as inputs.

Q-Net and baseline model were trained for addition and multiplication with numbers in training ranges (e.g., from zero to 24 for addition; from one to five for multiplication) and were evaluated on generalization performance with numbers outside of the training ranges (e.g., from 25 to 49 for for addition; from six to nine for multiplication). Hence, evaluations for both addition and multiplication were to test Q-Net on extrapolation, as opposed to interpolation, which is a challenging task because the model has not seen data in the test range before at all. For addition, both Q-Net and the baseline model showed perfect performances with unseen numbers. But for multiplication, only Q-Net showed very good performance with unseen numbers (and almost perfect performance with training numbers); the baseline model showed very poor performance even with training numbers. Details are described in Results.

Q-NET OVERVIEW

Q-Net has a modular architecture: A-module and C-module, which are trained separately. A-module, which is similar to autoencoder, is designed to learn the abstract meaning of numbers. Its encoder part converts numbers (x_{num}) into quantity encodings ($\mathbf{x}_{quant}$) and then extracts the abstract representations ($\mathbf{z}_x$) of quantity encodings. The decoder part of A-module decodes the abstract representation of quantities to numbers.

A-Module: Quantity encoding is a new approach that I propose to encode input numbers that are used to teach A-module on the meaning of numbers. It was inspired from the way that we may teach children about number. For example, the meaning of number ‘3’ may be taught by showing a card with three apples. I define quantity encoding as a multi-hot vector (or matrix) representation of numbers (Fig. 1).

$$x_{num} = 3$$

$$\mathbf{x}_{quant} = \square\blacksquare\square\square\cdots\blacksquare\square\square\blacksquare\square$$

Fig. 1. Quantity encoding of the number 3.

Since there are more than one possible configuration for the same number, the numbers (x_{num}) are converted into multi-hot vectors ($\mathbf{x}_{quant}$) using a stochastic function ϕ:

$$\mathbf{x}_{quant} = \phi(x_{num}).$$

To obtain the abstract representation of quantity encodings, I designed an NN architecture inspired from autoencoder. Autoencoder is a deep neural network in which the encoder and decoder are functions that map between a data point $\mathbf{x}$ and its latent representation $\mathbf{z}$ that describes abstract features of $\mathbf{x}$ (i.e., $\mathbf{z} = f(\mathbf{x})$ and $\mathbf{x} = g(\mathbf{z})$, where f and g denote encoder and decoder, respectively). The dimensionality of the latent vector of the autoencoder is much smaller than the dimensionality of data such that the encoder compresses the information contained in $\mathbf{x}$ rather than simply copying $\mathbf{x}$. In Q-Net, the abstract representations of the numbers ($\mathbf{z}_x$) are obtained by an encoder part of A-module (f):

$$\mathbf{z}_x = f(\mathbf{x}_{quant}),$$

which is similar to the encoder part of autoencoder in that it reduces the dimensionality of input vector. However, the decoder part of A-module is different from that of autoencoder in that it converts $\mathbf{z}_x$ directly into the number ($\hat{x}_{num}$):

$$\hat{x}_{num} = g(\mathbf{z}_x).$$

The decoder part of A-module is designed to avoid reconstructing the quantity vector ($\hat{\mathbf{x}}_{quant}$); because many different $\mathbf{x}_{quant}$ are possible for the same number, it is difficult to find an appropriate loss function to compare the input of encoder and the output of decoder for the same number. The NN architecture of A-module to learn the abstract meaning of numbers are shown in Fig. 2.

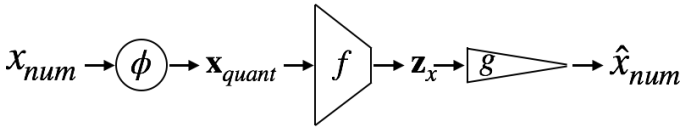

Fig. 2. NN architecture of A-module that extracts abstract representation of numbers.

C-Module: C-module, denoted by h, is a NN that performs algebraic operations in a latent space. It takes two abstract representations of numbers ($\mathbf{z}_{x=a}$ and $\mathbf{z}_{x=b}$) and a one-hot vector that instructs the type of algebraic operation ($\mathbf{o}$) and outputs the operation result also in the form of the latent vector ($\mathbf{z}_y$):

$$\mathbf{z}_y = h(\mathbf{z}_{x=a},\ \mathbf{z}_{x=b},\ \mathbf{o}).$$

$\mathbf{z}_{x=a}$ and $\mathbf{z}_{x=b}$ are provided by the encoder part of A-module, and $\mathbf{z}_y$ is converted to output number ($\hat{y}$) by the decoder of A-module (g):

$$\hat{y} = g(\mathbf{z}_y).$$

This is why A-module and C-module are to be trained separately. The overall architecture of Q-Net is shown in Fig. 3.

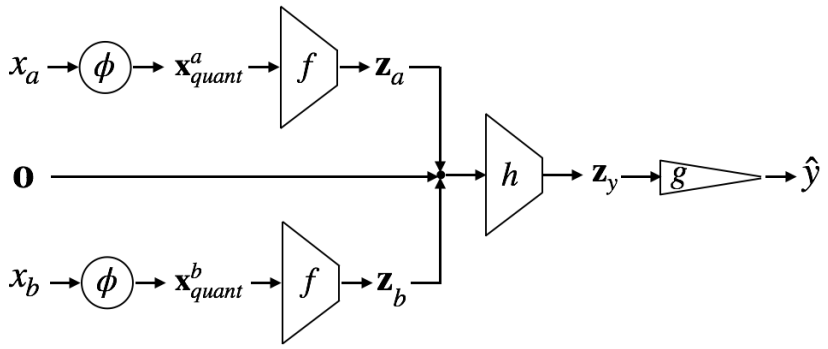

Fig. 3. Overall architecture of Q-Net

EXPERIMENTS

Setup, Training, and Evaluation of A-Module: A-module was designed to learn the abstract representations of integers ranging from zero to 99 ($x_{num} \in \{0,1,\ldots,99\}$). The encoder part ($f$) of the A-module was comprised of an input layer of size 100 for fixed size 100-dimensional quantity encoding vector ($\mathbf{x}_{quant} \in \{0, 1\}^{100}$), followed by two fully connected layers (64, 32) with ReLU activation function, and an output layer of size 16. The decoder part (g) of A-module was comprised of an input layer of size 16 corresponding to 16-dimensional abstract latent vector ($\mathbf{z}_x \in \mathbb{R}^{16}$), two fully connected layers (8, 4) with ReLU activation function, and an output layer with a single node that outputs the resulting number ($\hat{x}_{num}$) of algebraic operation. Since it was impractical to manage all possible configurations of $\mathbf{x}_{quant}$ (the total number of possible configurations for all numbers ranging from zero to 99 is $\Sigma_{x=0}^{99} 100!/[x!(100-x)!] \approx 10^{30}$), dataset for training and evaluating A-module was constructed with 1,000 randomly generated quantity encoding vectors per each number (total 10,000 data points); note that this results in redundant data points for numbers 0, 1, and 99. Train and test data were split by 7:3 ratio of data. Mean squared error (MSE) loss function and adam optimizer were used. A-module was trained for maximum 200 epochs with early stopping with patience 3, and the performance of the A-module was evaluated with test dataset using MSE as a test metric. In addition, the 16-dimensional latent space of A-module was visualized using principal component analysis (PCA).

Setup of C-Module and Q-Net: C-module (h) was comprised of an input layer of size 48 for taking two abstract latent vector ($\mathbf{z}_{x=a}$ and $\mathbf{z}_{x=b}$) and an operation instruction vector ($\mathbf{o} \in \mathbb{R}^{16}$), a single hidden layer of size 32, and an output layer of size 16 that outputs the operation result ($\mathbf{z}_y$). ReLU activations were used for hidden layers. Instruction vector was one-hot encoded for the two algebraic operations - addition and multiplication. After training the A-module, C-module was combined with the learned parts of A-module to construct Q-Net as shown in Fig. 3.

Training and Evaluation of Q-Net: Q-Nets for addition tasks and multiplication tasks were similarly trained and evaluate as described in the previous section (maximum 100 epochs) with training and test datasets prepared as follows. For *addition* tasks, training dataset was constructed using two input numbers (denoted as x_a and x_b) each ranging from zero to 24 and ground truth numbers (denoted as y) ranging from zero to 48:

$$x_a \in \{0,1,...,24\},$$
$$x_b \in \{0,1,...,24\},$$
$$y \in \{0,1,...,48\}.$$

There were 625 (25 × 25) combinations of the two numbers, and each combination was repeated 10 times, resulting total of 6,250 data points in the training dataset. After training, the performance of Q-Net for addition task was evaluated with test dataset constructed using two input numbers each ranging from 25 to 49 and ground truth numbers ranging from 50 to 98:

$$x_a \in \{25,26,...,49\},$$
$$x_b \in \{25,26,...,49\},$$
$$y \in \{50,51,...,98\}.$$

Test dataset was also prepared as described for the training dataset, resulting in a total of 6,250 data points. Note that there are no overlapping numbers in the train and test data.

For *multiplication* tasks, training dataset was constructed using two input numbers (denoted as x_a and x_b) each ranging from one to five and ground truth numbers ranging from one to 25:

$$x_a \in \{1,2,...,5\},$$
$$x_b \in \{1,2,...,5\},$$
$$y \in \{1,2,...,25\}.$$

And test dataset was constructed using two input numbers (denoted as x_a and x_b) each ranging from six to nine and ground truth numbers ranging from 36 to 81:

$$x_a \in \{6,7,...,9\},$$
$$x_b \in \{6,7,...,9\},$$
$$y \in \{36,42,...,81\}.$$

For datasets for multiplication tasks, each combination was repeated 200 times, resulting in a total of 5,000 data points in training dataset and total of 3,200 data points in test dataset. Note that there are no overlapping numbers in the train and test data.

RESULTS AND DISCUSSION

Abstract representation of numbers: The purpose of autoencoder-like A-module is to learn the abstract representations of natural numbers ranging from zero to 99. Although A-module was trained with a limited number of training samples (only 700 randomly generated quantity encoding vectors per each number), it reconstructed input numbers in test dataset with extremely low error rate (MSE: 3.4×10^{-4}) (Fig. 4)

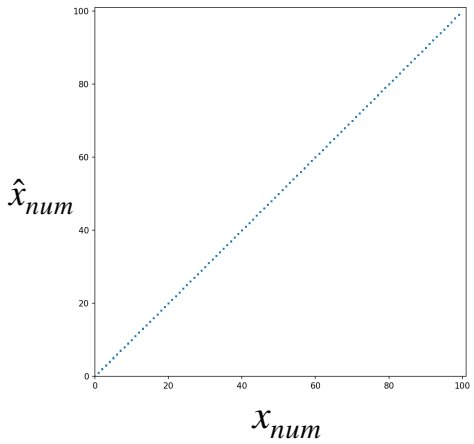

Fig. 4. Reconstruction of input numbers by A-module.

Interestingly, 16-dimensional latent vectors (abstract representation of numbers, $\mathbf{z}_x$) projected on two-dimensional and three-dimensional PCA spaces were well clustered according to numbers represented by them (Fig. 5). It appeared that PC1 could correspond to the numbers represented by the latent vectors and PC2 in 2D PCA (PC2 and PC3 in 3D PCA) could correspond to the diversity (the number of different configurations) in quantity encodings ($\mathbf{x}_{quant}$).

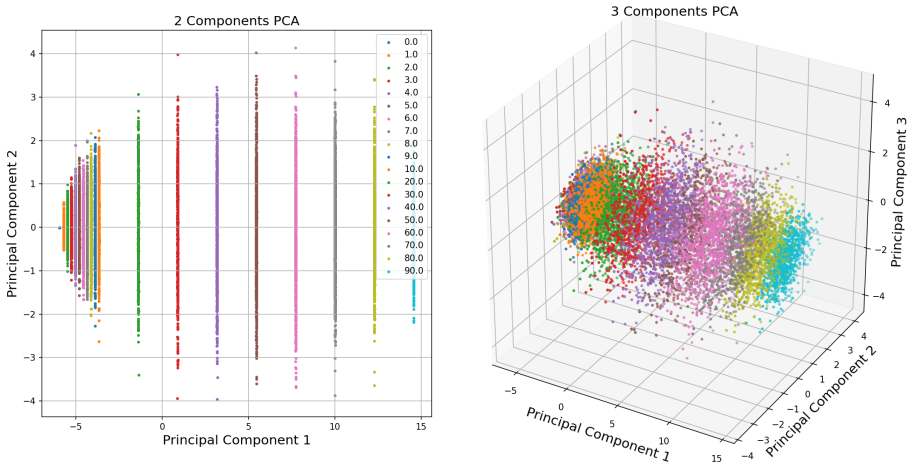

Fig. 5. Latent space projected on 2-dimensional space (left) and 3-dimensional space using PCA. Each point corresponds to latent vector (abstract representation of number, $\mathbf{z}_x$).

Addition Performance of Q-Net for Unseen Numbers: The generalization capability of Q-Net for addition was evaluated and compared with that of baseline model. The baseline model was a simple one layer perceptron without bias and activation function as shown in Fig. 6. Note that baseline model does not use quantity encoding and abstract representation of the quantity.

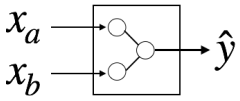

Fig. 6. Baseline model for addition.

Because the baseline model is '$\hat{y} = \mathbf{w}^{\mathrm{T}}\mathbf{x} = w_a x_a + w_b x_b$,' it should show good performance for addition with unseen numbers if it learns $w_a = 1$ and $w_b = 1$. When evaluated with the same test dataset, both baseline model and Q-Net showed almost perfect performance for the unseen numbers (Fig. 7).

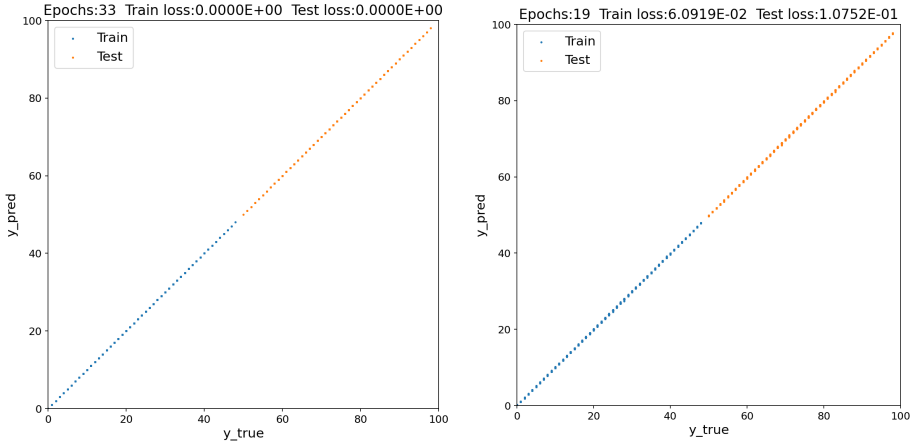

Fig. 7. Performances of baseline model (left) and Q-Net (right) for addition. Blue: numbers used in training, red: unseen numbers, diagonal line: perfect performance.

Multiplication Performance of Q-Net for Unseen Numbers: Theoretically, baseline model shown in Fig. 6 cannot perform multiplication. I tried many possible baseline architectures (without quantity encoding and abstract representation the quantity), but all failed to perform multiplication. Considering that the multiplication is a repeated addition, it was expected that MLP could show some performance for multiplication. But even with the training samples, the baseline models did not showed satisfactory performance. The baseline model that showed best performance among all models tried is shown in Fig. 8.

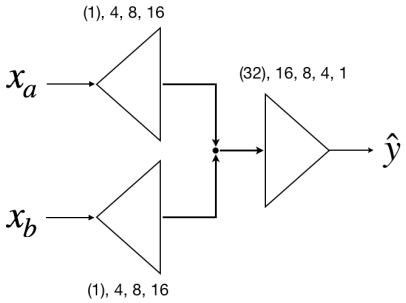

Fig. 8. Baseline model for multiplication.

When compared with the baseline model, Q-Net showed significantly better for multiplication task (Fig. 9). Baseline models failed even with the training samples. Furthermore, it seemed that outputs of baseline model were limited to the maximum output value (25) used in training. The failure of baseline model for multiplication task could be attributed to that it did not learn the abstract meaning of the numbers.

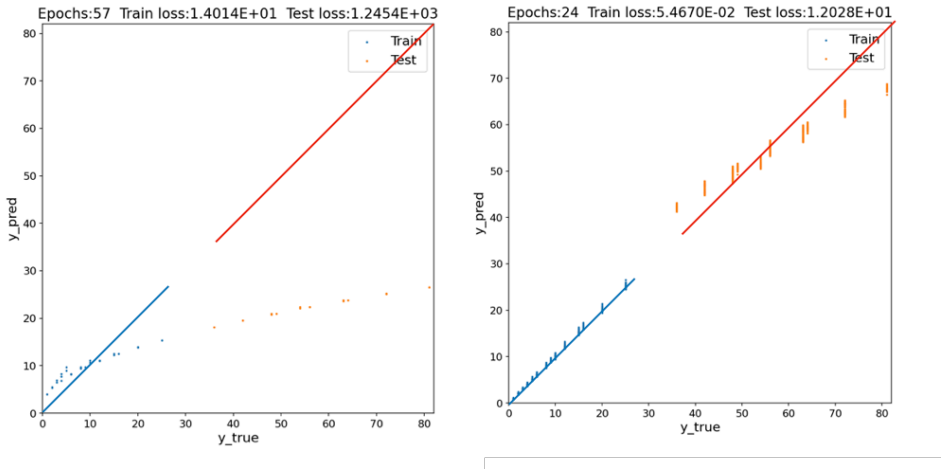

Fig. 9. Performances of baseline model (left) and Q-Net (right) for multiplication. Blue: numbers used in training, red: unseen numbers, diagonal line: perfect performance.

Compositional Algebra: To further challenge Q-Net on its generalization capability, I tested Q-Net with the following compositional algebraic equation with both seen and unseen natural numbers during training.

$$y_{ab+c} = x_a \times x_b + x_c$$

$$x_a = [1,2,3,4,5,6,7,8,9,10]$$
$$x_b = [5,6,7,8,9,5,6,7,8,9]$$
$$x_c = [45,43,39,33,25,45,38,29,18,5]$$

$$` \quad y_{ab+c} = [50,55,60,65,70,75,80,85,90,95]$$

In order to make Q-Net solve the compositional algebraic equation, I modified the original Q-Net by combining pre-trained C-modules for multiplication and addition as shown in Fig. 10.

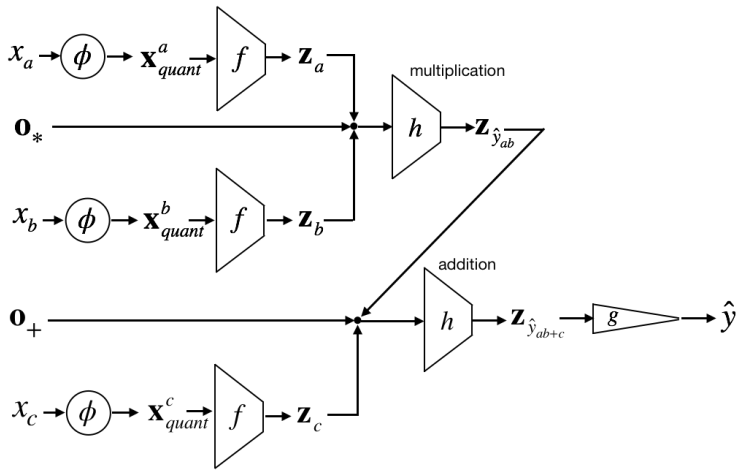

Fig. 10. Modified Q-Net for compositional algebra.

Note that 6, 7, 8, 9, and 10 are unseen numbers during pre-training C-module for multiplication, and 45, 43, 39, 33, 45, 38, and 29 are unseen numbers during pre-training C-module for addition. The evaluation result showed that Q-Net performs compositional algebraic operation quite well for small output numbers but not well for large output numbers (Fig. 11).

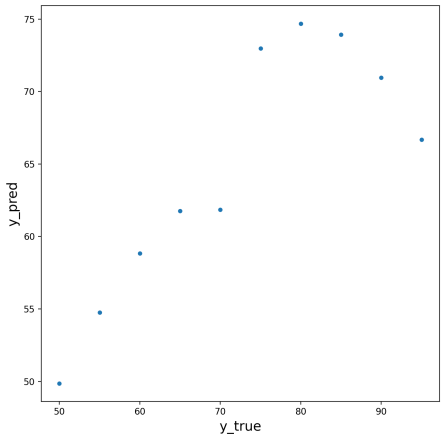

Fig. 11. Performance of Q-Net on compositional algebra

Concluding Remarks: Q-Net showed generalization potential of NN for basic algebraic operations on unseen numbers. The key components of Q-Net were the quantity encoding of numbers and the abstract representation of the quantities. Compare to baseline models, Q-Net showed significantly better performance for multiplication on unseen numbers. In this study, I have limited the input to a small range of natural numbers and also limited algebraic operations to addition and multiplication only. In the future, it would be interesting to test on real numbers. Although I have only used learned model to test compositional algebra, it would also be of interest to design a general algorithm for performing compositional algebra from parse tree.

While the specific goal of the present project was to design an NN capable of generalizing basic algebraic operations to unseen numbers, this project was launched in the effort to make AI one step forward to human brain that has a great generalization ability. Q-Net is yet in its infancy state, and there remains many steps for Q-Net to solve various math equations including compositional algebra. I hope this work serves as a starting point for improving NN's generalization capability with a very simple yet novel approach.

ACKNOWLEDGMENT

I would like to thank Professor George Konidaris at Brown University for exposure to this research area, general artificial intelligence, and for helpful discussion.