Visual Learning of Arithmetic Operations

Yedid Hoshen and **Shmuel Peleg**

School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem, Israel

Abstract

A simple Neural Network model is presented for end-to-end visual learning of arithmetic operations from pictures of numbers. The input consists of two pictures, each showing a 7-digit number. The output, also a picture, displays the number showing the result of an arithmetic operation (e.g., addition or subtraction) on the two input numbers. The concepts of a number, or of an operator, are not explicitly introduced. This indicates that addition is a simple cognitive task, which can be learned visually using a very small number of neurons.

Other operations, e.g., multiplication, were not learnable using this architecture. Some tasks were not learnable end-to-end (e.g., addition with Roman numerals), but were easily learnable once broken into two separate sub-tasks: a perceptual *Character Recognition* and cognitive *Arithmetic* sub-tasks. This indicates that while some tasks may be easily learnable end-to-end, other may need to be broken into sub-tasks.

1 Introduction

Visual learning of arithmetic operations is naturally broken into two sub-tasks: A perceptual sub-task of optical character recognition (OCR) and a cognitive sub-task of learning arithmetic. A common approach in such cases is to learn each sub-task separately. Examples of popular perceptual sub-tasks in other domains include object recognition and segmentation. Cognitive sub-tasks include language modeling and translation.

With the progress of deep neural networks it has become possible to learn complete tasks end-to-end. Systems now exist for end-to-end training of image to sentence generation (Vinyals et al. 2014) and speech to sentence generation (Hannun et al. 2014). But end-to-end learning may introduce an extra difficulty: sub-tasks do not have unique training data, but depend on the results of other sub-tasks.

We examine end-to-end learning from a neural network perspective as a model for perception and cognition: performing arithmetic operations (e.g., addition) for visual input and visual output. Both input and output examples of the network are pictures (as in Fig. 1). For each training example we give the student (the network) two input pictures,

each showing a 7 digit integer number written in a standard font. The target output is also a picture, displaying the sum of the two input numbers.

In order to succeed at this task, the network is required to implicitly be able to learn the arithmetic operation without being taught the meaning of numbers. This can be seen as similar to teaching arithmetic to a person with whom we do not possess a common language.

We model the learning process as a feed-forward artificial neural network (Bishop 1995; Hinton and Salakhutdinov 2006). The input to the network are pictures of numbers, and the output is also a picture (of the sum of the input numbers). The network is trained on a sufficient number of examples, which are only a tiny fraction of all possible inputs. After training, given pictures of two previously unseen numbers, the network generates the picture displaying their sum. It has therefore learned the concept of numbers without direct supervision and also learned the addition operation.

Although initially a surprising result, we present an analysis of visual learning of addition and demonstrate that it is realizable using simple neural network architectures. Other arithmetic operations such as subtraction are also shown to be learnable with similar networks. Multiplication, however, was not learned successfully under the same setting. It is shown that the multiplication sub-task is more difficult to realize than addition under such architecture. Interestingly, for addition with Roman numerals both the OCR and the arithmetic sub-tasks are shown to be realizable, but the end-to-end training of the task fails. This demonstrates the extra difficultly of end-to-end training.

Our results suggest that some mathematical concepts are learnable purely from vision. An exciting possible implication is that some arithmetic concepts can be taught visually across different cultures. It has also been shown that end-to-end learning fails for some tasks, even though their sub-tasks can be learned easily. This work deals with arithmetic tasks, and future research is required to characterize what other non-visual sub-tasks can be learned visually e.g., by video frame prediction.

2 Arithmetic as Neural Frame Prediction

In this section we describe a visual protocol for learning arithmetic by image prediction. This is done by training an artificial neural network with input and output examples.

	Example A	Example B	Failure Example
Input Picture 1	981925	3570002	3750668
Input Picture 2	2010445	1216536	3643197
Network Output Picture	2992370	4786538	7393855
Ground Truth Picture	2992370	4786538	7393865

Figure 1: Input and output examples from our neural network trained for addition. The first two examples show a typical correct response. The last example shows a rare failure case.

2.1 Learning Arithmetic from Visual Examples

Our protocol for visual learning of arithmetic is based on image prediction. Given two input pictures F_1, F_2, target picture E is the correct prediction. The learner is required to predict the output picture, and the predicted picture is denoted P. The prediction loss is evaluated by the sum of square differences (SSD) between the pixel intensities of the predicted picture P and the target picture E.

The input integers are randomly selected in a pre-specified range (for addition we use the range of [0,4999999]), and are written on the input pictures. The result of the arithmetic operation on the input numbers (e.g., their sum) is written on the target output picture E. The numbers were written on the pictures using a standard font, and were always placed at the same image position. See Fig. 1 for examples.

Learning consists of training the network with N such input/output examples (we use $N = 150,000$).

2.2 Network Architecture

In this section we present a method to test the feasibility of learning arithmetic given the protocol presented in Sec. 2.1. Our simple but powerful learner is a feed-forward fully-connected artificial neural network as shown in Fig. 2.

The network consists of an input layer of dimensions $F_x \times F_y \times 2$ where F_x and F_y are the dimensions of the 2 input pictures. We used $F_x \times F_y = 15 \times 60$ unless specified otherwise. The network has three hidden layers each with 256 nodes with ReLU activation functions ($max(0, x)$) and an output layer (of the same height and width as the input pictures) with sigmoid activation. All nodes between adjacent layers were fully connected. An L_2 loss function is used to score the difference between the predicted picture and the expected picture. The network is trained via mini-batch stochastic gradient descent using the backpropagation algorithm.

3 Experiments

The objective of this paper is to examine if arithmetic operations can be learned end-to-end using purely visual information. To this end several experiments were carried out:

3.1 Experimental Procedure

Using the protocol from Sec. 2.1 we generated 2 input pictures per example, each showing a single integer number. The numbers were randomly generated from a pre-specified range as detailed below. The output pictures were created similarly, displaying the result of the arithmetic operation on the input.

The following arithmetic operations were examined:

- Addition: Sum of two 7 digit numbers, each in the range [0,4999999].
- Subtraction: Difference between two 7 digit numbers in the range [0,9999999]. The first number was chosen to be larger or equal to the second number to ensure a positive result.
- Multiplication: Product of two numbers, each in the range [0,3160].
- Addition of Roman Numerals: Sum of two numbers in the range [0,4999999]. Both input and output were written in Roman numerals (IVXLCDM and another 7 numerals we "invented" from 5000 to 5000000). The longest number 9,999,999 was 35 numerals long. The medieval notation (IV instead of IIII) was not used.

For each experiment, 150,000 input/output pairs were randomly generated for training and 30,000 pairs were randomly created for testing. The proportion of numbers used for training is a very small fraction of all possible combinations.

We have also examined robustness to image noise of the addition experiment. Both input and output pictures were corrupted with a strong additive Gaussian noise.

A feed-forward artificial network was trained with the architecture described in Fig. 2. The network was trained using mini-batch stochastic gradient descent with learning rate

Input Layer
Nodes: 60X15
Depth: 2

HL1
256 nodes
Fully Con.
ReLU

HL2
256 nodes
Fully Con.
ReLU

HL3
256 nodes
Fully Con.
ReLU

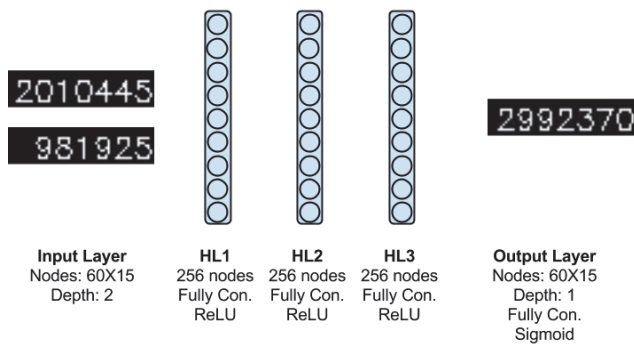**Output Layer**
Nodes: 60X15
Depth: 1
Fully Con.
Sigmoid

Figure 2: A diagram showing the construction of a neural network with 3 hidden layers able to preform addition using visual data. Two pictures are used as input and one picture as output. The network is fully connected and uses ReLU units in the hidden layers and sigmoid in the output layer. The hidden layers have 256 units each.

0.1, momentum 0.9 and mini-batch size was 256. 50 epochs of training were carried out. The network was implemented using the Caffe package (Jia et al. 2014).

3.2 Results

The correctness of the test set was measured using an OCR software (Tesseract (Smith 2007)) which was applied to the output pictures. The OCR results were compared to the desired output, and the percentage of incorrect digits was computed. The effectiveness of the neural network approach has been tested on the following operations.

Addition: Three results from the test set are shown in Fig. 1. The input and the output numbers were not included in the training set. The examples qualitatively demonstrate the effectiveness of the network at learning addition from purely visual information. Quantitatively, the network has been able to learn addition with great accuracy, with incorrect digit prediction rate being only 1.9%.

Subtraction: We trained a neural network having identical architecture to the network used for addition. Subtraction of a small number from a larger one was found to be of comparable difficulty to addition. The predicted digit error rate was around 3.2% which is comparable to addition.

Multiplication: This task was found to be a much more challenging operation for a feed-forward Neural Network. The data for this experiment consisted of two input pictures with 4-digit integers, resulting in an output picture with 7 digit number, and the network used was similar to the one used for addition. As theoretical work (e.g., (Franco and Cannas 1998)) has shown that multiplication of binary numbers may require two more layers than their addition, we experimented with adding more hidden layers. The network, even with 5 hidden layers, did not perform well on this task, giving very large train and test errors. An example input/output pair can be seen in Fig.3. It can be seen that the least significant digit and two most significant digits were predicted correctly, as enumeration of the different possibilities is feasible, but the network was uncertain about the

Operation	Pictures		1-hot Vectors	
	No. Layers	% Error	No. Layers	% Error
Add	3	1.9%	1	1.7%
Subtract	3	3.2%	1	2.1%
Multiply	5	71.5%	3	37.6%
Roman Addition	5	74.3 %	3	0.7 %

Table 1: The digit prediction error rates for end-to-end training on pictures, and for the stripped 1-hot representation described in Sec. 5. For the purpose of error computation, the digits in the output predicted images were found using OCR. Addition and subtraction are always accurate. The network was not able to learn multiplication. Although Roman numeral addition failed using the picture prediction network, it was learned successfully for 1-hot vectors.

central 4 digits. The predicted digit error rate was as high as 71%, and the OCR engine was often unable to read numbers that had several blurry (uncertain) digits.

Addition of Roman numerals: It has been hypothesized by Marr (Marr 1982) and others (see (Schlimm and Neth 2008)) that arithmetic using Roman numerals can be more challenging than using Arabic numerals. We have repeated the addition experiment with all numbers written as Roman numerals, which can be up to 35 digits long. As is demonstrated quantitatively in Tab. 1 the network was not able to predict the output frame in Roman numeral basis. This suggests that end-to-end visual learning of addition in Roman numeral basis is more challenging, in agreement with Marr's hypothesis. We further analyze this result in Sec. 5.

Addition with Noisy Pictures: In one experiment we added a strong Gaussian noise ($\sigma=0.3$) to all input and output pictures, as can be seen in Fig.3. The network achieved very good performance on this task, giving output pictures that display the correct result, which are also clean from noise. Failures can occur when the input digits are almost illegible. In such cases the network generated a "probabilistic" output digit displaying a mixture of two digits. Mixture of digits caused problems to our verification using an OCR, reporting 9.8% digit error rate whereas human inspection obtained only 3.2% error rate. See Fig.5 for further details.

4 Previous Work

Theoretical characterization of the operations learnable by neural networks is an established area of research. A pioneering paper presented by (Hajnal et al. 1987) used threshold circuits as a model for neural network capacity. A line of papers (e.g., (Hofmeister, Hohberg, and Köhling 1991; Siu et al. 1993; Franco and Cannas 1998)) established the feasibility of the implementation of several arithmetic operations on binary numbers. Recently (Graves, Wayne, and Danihelka 2014) has addressed implementing Universal Turing Machines using neural networks. Most theoretical work in the field used binary representation of numbers, and did not address arithmetic operations in decimal form. Notably, a general result (see (Shalev-Shwartz and Ben-David

	Subtraction	Multiplication	Noisy Addition
Input Picture 1	8184756	2531	4752072
Input Picture 2	5363281	1312	230865
Network Output Picture	2821475	3338852	4982677
Ground Truth Picture	2821475	3320672	4352677

Figure 3: Examples of the performance of our network on subtraction, multiplication, and addition with noisy pictures. The network performs well on subtraction and is insensitive to additive noise. It performs poorly on multiplication. Note that the bottom right image is not the ground truth image, but an example of the type of training output images used in the Noisy Addition scenario.

2014)), shows that operations implementable by Turing machine in time $T(n)$ can be implemented by a neural network of $O(T(n))$ layers and with $O(T(n)^2)$ nodes. It has sample complexity $O(T(n)^2)$ but has no guarantees on training time. Research has also not dealt with visual learning.

Hypotheses about the difference in difficulty of learning arithmetic using decimal vs. Roman representations was made by Marr (Marr 1982) and others. see (Schlimm and Neth 2008) for a review and algorithms for Roman numeral addition and multiplication.

Optical Character Recognition (OCR) (LeCun et al. 1990; Jaderberg, Vedaldi, and Zisserman 2014) is a well studied field. In this work we only deal with a very simple OCR scenario, dealing with more complex characters and backgrounds is out of scope of this work.

Learning to execute Python code (including arithmetic) from textual data has been shown to be possible using LSTMs by Zaremba and Sutskever (Zaremba and Sutskever 2014). Adding two MNIST digits randomly located in a blank picture has been performed by Ba et al. (Ba, Mnih, and Kavukcuoglu 2014). In (Zaremba, Kurach, and Fergus 2014), Recurrent Neural Networks (RNNs) were used for algebraic expression simplification. These works, however, required a non-visual representation of a number either in the input or in the output. In this paper we show for the first time that end-to-end visual learning of arithmetic is possible.

End-to-end learning of Image-to-Sentence (Vinyals et al. 2014) and of Speech-to-Sentence (Hannun et al. 2014) has been described by multiple researchers. A recent related work by Vondrick et al. (Vondrick, Pirsiavash, and Torralba 2015) successfully learned to predict the objects to appear in a future video frame from several previous frames. Our work can also be seen as frame prediction, requiring the network to implicitly understand the concepts driving the change between input and output frames. But our visual arithmetic is an easier task: easier to interpret and to analyze. The greater simplicity of our task allows us to use raw frames rather than

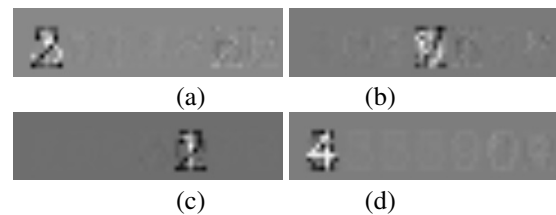|	
(a)	(b)
(c)	(d)

Figure 4: (a-b) Examples of bottom layer weights for the first input picture. (a) recognizes '2' at the leftmost position, while (b) recognized '7' at the center position. (c-d) Examples of top layer weights. (c) outputs '1' at the second position, while (d) outputs '4' at the leftmost position.

an intermediate representation as used in (Vondrick, Pirsiavash, and Torralba 2015).

5 Discussion

In this paper we have shown that feed-forward deep neural networks are able to learn certain arithmetic operations end-to-end by purely visual cues. Several other operations were not learned by the same architecture. In this section we give some intuition for the method the network employs to learn addition and subtraction, and the reasons why multiplication and Roman numerals were more challenging. A proof by construction of the capability of a shallow DNN (Deep Neural Network) to perform visual addition is presented in Sec. 6.

When looking at the network weights for both addition and subtraction, we can see that each bottom hidden layer node is sensitive to a particular digit at a given position. Example bottom layer weights can be observed in Fig. 4.a-b. The bottom hidden layer nodes therefore represent each of the two M-digit numbers as a vector of length $10 \times M$, each element representing the presence of digit $0 - 9$ in position

$m \in [1, M]$. This representation of converting a variable with D possible values (here 10) as D binary variables all being 0 apart from a single 1 at the dth position is known as "1-hot". The top hidden layer contains a similar representation of the output number representing the presence of digit $0-9$ in position $m \in [1, M]$ with total size $10 \times M$. The task of the central hidden layers is mapping between the 1-hot representations of the input numbers (size $10 \times M \times 2$) and the 1-hot representation of the output number (size $10 \times M$).

The task is therefore split into 2 sub-tasks:

- Perception: learn to represent numbers as a set of 1-hot vectors.

- Cognition: map between the binary vectors as performed by the arithmetic operation.

Note that the second sub-task is different from arithmetic operations on binary numbers (and is often harder).

In order to evaluate the above sub-tasks separately, we repeated the experiments with the (input and output) data transformed to 1-hot representation, thereby bypassing the visual sub-task. We used the same architecture as in the end-to-end case, except that we removed the first and last hidden layers (that are used for detecting or drawing images of numbers at each location).

The results on the test sets measured as the percentage of wrong digits in the output number is presented in Tab. 1. Addition and subtraction are both performed very accurately as in the visual case. The network was not able to learn multiplication due to the difficulty of the arithmetic sub-task, in line with the results of the visual case. This is also justified theoretically as (i) Binary multiplication was shown by previous papers (Siu et al. 1993; Franco and Cannas 1998) to require deeper networks than binary addition. (ii) The Turing Machine complexity of the basic multiplication algorithm (effective for short numbers) is $O(n^2)$ as opposed to $O(n)$ for decimal addition (n is the number of digits). This means (Shalev-Shwartz and Ben-David 2014) that the operation is realizable only by a deeper ($O(n^2)$ vs. $O(n)$ layers) and larger network ($O(n^4)$ vs. $O(n^2)$ nodes).

More interesting is the relative accuracy at which Roman numeral addition was performed, as opposed to the failure in the visual case. We believe this is due to the high number of digits for large numbers in Roman numerals (35 digits), which causes both input and output images to be very high dimensional. We hypothesize that convergence may be improved with preliminary unsupervised learning of the OCR tasks (i.e. teaching the network what numbers are by clustering). We conclude that Roman arithmetic can be learned by DNNs, but visual end-to-end learning is more challenging due to the difficulty of joint optimization with the OCR sub-task.

Visual learning when data were corrupted by strong noise was quite successful. In fact the concepts were learned well enough that the output pictures were denoised by the network. The performance on illegible digits is particularly interesting. We found that on corrupted digits that could possibly be read as multiple possibilities (In Fig. 5, digits 8 or 5), the output digit also reflected this uncertainty, resulting in a
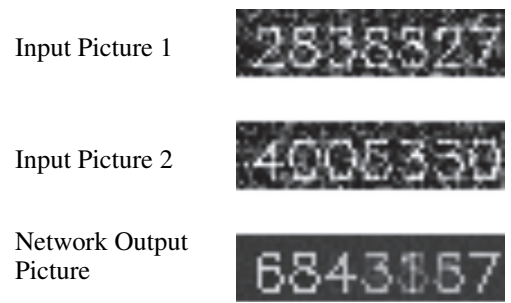
Input Picture 1

Input Picture 2

Network Output Picture

Figure 5: Probabilistic arithmetic for noisy pictures: The third digit from right in "Input Picture 1" can be either 5 or 8. The corresponding output digit is a mixture of 1 and 8.

mixture of the two possible outputs (In Fig. 5, digits 1 or 8) with their respective probabilities. In other experiments (not shown) we have found that visual learning works for unary operations too (e.g., division by 2).

A significant difference between our model and the cognitive system is its invariance to a fixed permutation of the pixels. A human would struggle to learn from such images, but the artificial neural networks manages very well. This invariance can be broken by slight random displacement of the training data or by the introduction of a convolutional architecture.

Although Recurrent Neural Networks are generally better for learning algorithms (such as multiplication), we have chosen to use a fully connected architecture for ease of analysis. We hypothesize that better performance on multiplication can be obtained using an LSTM-RNN (Long Short Term Memory - Recurrent Neural Network) but we leave this investigation for future work.

6 Feasibility of a Visual Addition Network

In this section we provide a feasibility proof by construction of a neural network architecture that can learn addition from visual data end-to-end. The construction of the network is illustrated in Fig. 6.

We rely on logic gates for simplicity. A logic gate can be implemented to an arbitrary accuracy by a single sigmoid or by a linear combination of 2 ReLU units $\Theta(x > 0) = (ReLU(x + \delta) - ReLU(x))/\delta$. Although our reported results were obtained using a network utilizing ReLU units, we have also tested our network with ReLU units replaced by sigmoid units obtaining similar results but much slower convergence. Logic gates are therefore a sufficiently good model of our network.

An input example is shown in Fig. 1. The first layer of the network is a dimensionality reduction layer. We choose weights that correspond to the set of filters containing each digit n ($n \in 0..9$) at each position m. Our experimental network in fact chooses more complex filters usually concentrated between similar digits to increase accuracy of digit detection (see Fig. 6 for examples). We construct $10 \times M \times 2$ nodes in the HL1 layer indicating if each of the templates is triggered. Each first hidden layer node responds to a spe-
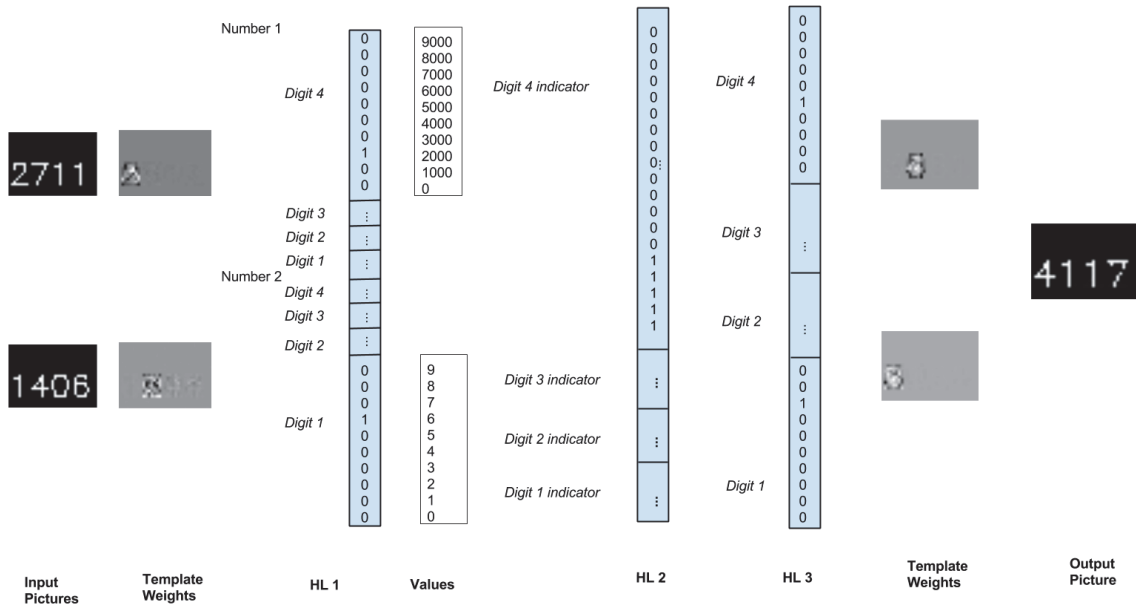
Figure 6: An illustration of the operation of a 3 hidden layer neural network able to perform addition using visual training. In this example the network handles only 4 digit numbers, but larger numbers are handled similarly with a linear increase in the number of nodes. i) The pictures are first projected onto a binary vector $HL1$ indicating if digit n is present at position m in each of the numbers. ii) In HL2 we compute indicator variables v_i^m for each digit $1..M$ and threshold $i = 0..19$. The variable is on if the summation result $\sum_{j=1}^{m}(d1^m + d2^m)\times 10^j$ exceeds threshold $i\times 10^m$. iii) In the final hidden layer we calculate if a template is displayed by observing if the indicator variable corresponding to its digit and position is on but the following indicator variable is off. The templates are then projected to the output layer.

cific template, for example $T2_m^n$ corresponds to the template detecting if the digit n is present at the mth position in picture 2. It has value 1 if a template appears and 0 if it does not. Similarly the output layer is represented as a set of templates each corresponding to a digit (0..9) at a given position (1..M).

It is worth noticing that given two digits $d1^m$ and $d2^m$ at the mth position in numbers 1 and 2 respectively, the mth digit in the output d_o^m can be either $(d_1^m + d_2^m)mod10$ or $(d_1^m + d_2^m + 1)mod10$. For each pair of digits, the arithmetic problem is to choose the correct result from the possible two.

In $HL2$ we compute an indicator function for each digit m, where node v_i^m is on when the sum of digits d_1 and d_2 and the possible increment from previous digits is larger than its threshold i ($i \in 0..19$). This is formulated as

$$v_i^m = \mathbb{1}_{\sum_{j=1}^{m}(d1^m + d2^m)*10^j >= i\times 10^m} \quad (1)$$

It is easily implemented for each node v_i^m with weights from $HL1$ nodes $T1_n^m$ and $T2_n^m$ with values $n*10^j$ for $j \in 1..m$, $n \in 0..9$ and threshold $i\times 10^m$. For later convenience we denote $v_{20}^m = 0$.

In $HL3$, output template o_n^m corresponding to the digit n at position m is turned on if in $HL2$ indicator $v_n^m = 1$ or $v_{n+10}^m = 1$ while $v_{n+1}^m = 0$ or $v_{n+11}^m = 0$ respectively. This corresponds to the cases where the summation result of the numbers up to digit m is $n\times 10^m \leq result < (n+1)\times 10^m$ or $(n+10)\times 10^m \leq result < (n+11)\times 10^m$. The equation

is therefore:

$$o_n^m = \mathbb{1}_{v_n^m - v_{n+1}^m + v_{n+10}^m - v_{n+11}^m > 0} \quad (2)$$

Finally the values are projected onto the output picture using the corresponding digit templates.

By end-to-end training of the network with a sufficient number of examples the network can arrive at the above weights (although it is by no means guaranteed to), and in practice good performance is achieved. End-to-end training from visual data is therefore theoretically shown and experimentally demonstrated to be able to learn addition with little guidance. This is a powerful paradigm that can generalize to visual learning of non-visual concepts that are not easily directly communicated to the learner.

7 Conclusions

We have examined the capacity of neural networks for learning arithmetic operations from pictures, using a visual end-to-end learning protocol. Our neural network was able to learn addition and subtraction, and was robust to strong image noise. The concept of numbers was not explicitly used. We have shown that the network was not able to learn some other operations such as multiplication, and visual addition using Roman numerals. For the latter we have shown that although all sub-tasks are easily learned, the end-to-end task is not.

In order to better understand the capabilities of the network, a theoretical analysis was presented showing how a

network capable of performing visual addition may be constructed. This theoretical framework can help determine if a new arithmetic operation is learnable using a feed-forward DNN architecture. We note that such analysis is quite restrictive, and hypothesize that experimental confirmation of the end-to-end learnability of complex tasks will often result in surprising findings.

Although this work dealt primarily with arithmetic operations, the same approach can be used for general cognitive sub-task learning using frame prediction. The sub-tasks need not be restricted to the field of arithmetic, and can include more general concepts such as association. Generating data for the cognitive sub-task in not trivial, but generating visual examples is easy, e.g., by predicting future frames in video.

While our experiments use two input pictures and one output picture, the protocol can be generalized for more complex operations involving more input and output pictures. For learning non-arithmetic concepts, the pictures may contain other objects beside numbers.

References

Ba, J.; Mnih, V.; and Kavukcuoglu, K. 2014. Multiple object recognition with visual attention. *arXiv:1412.7755*.

Bishop, C. M. 1995. *Neural networks for Pattern Recognition*. Oxford Univ. Press.

Franco, L., and Cannas, S. A. 1998. Solving arithmetic problems using feed-forward neural networks. *Neurocomputing* 18(1):61–79.

Graves, A.; Wayne, G.; and Danihelka, I. 2014. Neural turing machines. *arXiv:1410.5401*.

Hajnal, A.; Maass, W.; Pudlák, P.; Szegedy, M.; and Turan, G. 1987. Threshold circuits of bounded depth. In *Annual Symposium on Foundations of Computer Science*, 99–110. IEEE.

Hannun, A.; Case, C.; Casper, J.; Catanzaro, B.; Diamos, G.; Elsen, E.; Prenger, R.; Satheesh, S.; Sengupta, S.; Coates, A.; et al. 2014. Deepspeech: Scaling up end-to-end speech recognition. *arXiv:1412.5567*.

Hinton, G. E., and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Hofmeister, T.; Hohberg, W.; and Köhling, S. 1991. Some notes on threshold circuits, and multiplication in depth 4. In *Int. Conf. Fundamentals of Computation Theory*, 230–239. Springer.

Jaderberg, M.; Vedaldi, A.; and Zisserman, A. 2014. Deep features for text spotting. In *ECCV'14*, 512–528. Springer.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*.

LeCun, Y.; Boser, B. E.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. E.; and Jackel, L. D. 1990. Handwritten digit recognition with a back-propagation network. In *NIPS'89*.

Marr, D. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman & Co.

Schlimm, D., and Neth, H. 2008. Modeling ancient and modern arithmetic practices: Addition and multiplication with arabic and roman numerals. In *30th Annual Conference of the Cognitive Science Society*, 2097–2102.

Shalev-Shwartz, S., and Ben-David, S. 2014. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

Siu, K.-Y.; Bruck, J.; Kailath, T.; and Hofmeister, T. 1993. Depth efficient neural networks for division and related problems. *IEEE Trans. Information Theory* 39(3):946–956.

Smith, R. 2007. An overview of the tesseract ocr engine. In *Int. Conf. on Document Analysis and Recognition (ICDAR)*, 629–633.

Vinyals, O.; Toshev, A.; Bengio, S.; and Erhan, D. 2014. Show and tell: A neural image caption generator. *arXiv:1411.4555*.

Vondrick, C.; Pirsiavash, H.; and Torralba, A. 2015. Anticipating the future by watching unlabeled video. *arXiv:1504.08023*.

Zaremba, W., and Sutskever, I. 2014. Learning to execute. *arXiv:1410.4615*.

Zaremba, W.; Kurach, K.; and Fergus, R. 2014. Learning to discover efficient mathematical identities. In *NIPS'14*, 1278–1286.