Training Neural Networks for Object Classification and Number Abstraction

Shu-man Lin, *Student at Udacity*

***Abstract*—Two exercises are implemented on Nvidia's DIGITS deep learning GPU training system. First, object classification is performed on a neural network trained on images of candy boxes and bottles photographed on a conveyor belt. Second, a comparison is made between a neural network trained on images of numerically distinguished groups of three dimensional objects and a neural network trained on images of numerically distinguished groups of two dimensional shapes. This pair of networks are tested on images containing groups of three dimensional objects or groups of two dimensional irregular shapes to investigate each network's accuracy in classifying the number of objects or shapes in an image. Realization of accurate classification for numerical abstraction is achieved on the model trained on photographed three dimensional objects which successfully recognized number in both computer drawn, two dimensional figures and groups of three dimensional objects. The model trained on recognizing number for two dimensional figures failed to classify three dimensional photographed objects but succeeded to classify shapes in its respective dimension.**

***Index Terms*—Neural Networks, Deep Learning, Object Recognition, Number Abstraction, Synthetic Data**

I. Introduction

KEY to finding a deep learning model to solve the problem of object classification include a deep convolutional neural network and an annotated, many exampled dataset. Deep learning has made great advances in object recognition by improved network architecture construction and massive data preparation. Examples of such are LeNet-5 [1] which is a network trained on the MNIST dataset of handwritten digits to classify numerals and AlexNet [2] and GoogleNet [3] which are networks originally trained on the ImageNet dataset to classify images of objects partitioned into 1000 classes. Since the GoogleNet architecture was built to fit data sets to classification tasks involving varied types of objects, it would likely be suitable to train the network on a supplied dataset of images of objects on a conveyor belt for an object classification task of determining whether an image contained a candy box, a bottle, or nothing at all. This real time sorting task is the first exercise completed in this paper.

Beyond concrete object classifications, inquiry proceeds on whether a trained network can learn to identify number classes. Through visual perception and mathematical abstraction, processing of three dimensional segmentation of real world objects allows humans to learn to count. But can a machine learn to acquire the same ability? Is there a deep learning methodology with a network and prepared data representations accurate enough to train a model to learn number abstraction? Applications of item specific number counting in images has been asked and answered in many cases and multiple methods have been supplied for tasks such as estimating the number of cells in a microscopic image or estimating the number of humans in surveillance video frames [4], [5] and references within. Although real world classification tasks typically require a painstakingly large amount of similar situation, real world data samples for robust classification, a method in [6] generates synthetic images simulating tomato plants as training images for an Inception-ResNet based network to estimate the number of real world tomatoes in a photograph without detecting them.

On another note, a related curiosity is the nature of how a machine learns number differentiation as compared to humans. In [7], AlexNet is trained on a large number of two dimensional clusters of circles of varied sizes representing small natural numbers to evaluate the extent of visual numerosity detection on testing sets of polygon clusters of varied sizes. Findings support a failure for deep neural networks to learn only the property of topological invariance of number when circles are slightly perturbed to polygons from the original circle clusters they were trained on. Proper number estimation cannot be abstracted on testing samples with objects geometrically varying from circles if training examples only include number of circles and not number of the wider class of polygons. The specific property of identifying circles is mixed with the property of number estimation so the isolation of the emergent property of number by clustered classes of any shape is impossible to realize when a more general property of polygon is introduced. To train by specifying classes that answer the particular question of how many circles there are does not generalize to answering how many shapes there are when the model was not trained in recognizing what the general notion of a shape is. The import in defining clear object classes and providing many examples of such for machine learning differs from the relatively few examples a human being needs to learn a particular concept and the flexibility a human has to partition concepts in order to discover properties of new object classes.

In the second exercise, the question of how well two dimensional learning of number abstraction can generalize to three dimensional testing and how well three dimensional learning of number abstraction can restrict to two dimensional testing is explored. Object number is simulated and trained on two data sets. One data set consists of synthetically generated one, two, or three randomly dispersed irregular two dimensional shapes filled in black with white background. The second data set consists of black and white photographs of three dimensional

Shu-man Lin, e-mail: impetusengine@gmail.com

objects pictured together in classes of one, two, or three. The data set trained on two dimensional irregular shapes is tested on photographs of three dimensional objects and the data set trained on three dimensional objects is tested on images of two dimensional irregular shapes for the task of identifying number within images. Further, baseline classifications are made to check if training in three dimensions on similarly prepared testing sets tests well in three dimensions, likewise a test is made for two dimensions.

The remainder of the paper is structured as follows. Each section discusses a topic with two subsections, one subsection for each exercise discussing the topic on that particular exercise. Section 2 describes the experimental setup and network used. Section 3 describes how the data was collected. Section 4 discusses results. Section 5 provides a reflection on the results. Section 6 concludes with future work.

II. Background

A. Conveyor Belt Object Classification

The network architecture that was chosen for the project was GoogleNet. This architecture was originally tested on an augmented version of the Imagenet database which consists of 1.2 million images for training, 50,000 images for validation, and 100,000 images for testing to accomplish the task of classifying images into 1000 leaf-node categories. Classification performance was top performing and ended with a 6.67% top-5 error rate. For the task of conveyor belt object classification we consider a smaller sample of classes as compared to the ImageNet dataset and a smaller set of examples per class, however classification tasks are comparable and would be a feasible choice as an architecture to detect features of objects for object classification.

Other parameters set for the model include the following. Each image fit an input size of 256x256 pixels as specified by the GoogleNet architecture. The dataset contained 7570 images for training and 2524 images for validation. The solver type of choice was the standard stochastic gradient descent with an initial learning rate of 0.01 and reduction of the learning rate by a factor of 10 was taken for every 3-4 epochs within 10 epochs. The training time for the network was about 19 minutes on Nvidia's DIGITS deep learning GPU training system.

B. Classify Number Abstraction

GoogleNet was also the architecture of choice to test classification results for number abstraction. Two networks were trained, one on images of two dimensional representations of number and one on images of three dimensional representations of number to obtain two models: FlatNet and RealNet for 2-d and 3-d respectively. The particular feature to extract in two dimensions is the idea of two dimensional connected component while the feature to capture in three dimensions is three dimensional connected component projected onto two dimensional space. To do this, we use a proven high performing classification architecture that captures detailed structure of object classes of which includes the detection of shape and modify the data set with augmentation methods to prevent overfitting.

Other parameters set for the model include the following. Each image for both trained networks fit an input size of 256x256 pixels as specified by the GoogleNet architecture. When testing in two dimensions, FlatNet trained on 2550 images with a validation set of 450 images and a separate testing set of 300 images. When testing in three dimensions, FlatNet trained on the same 2550 images as in two dimensions with a testing set of 300 images from three dimensions. When testing in three dimensions, RealNet trained on 2400 images with 300 images set aside for validation and the same 300 images set aside for testing FlatNet in three dimensions. When testing in two dimensions, RealNet trained on the same 2400 images as in two dimensions and tested with the same 300 images set aside for testing FlatNet in two dimensions. For both FlatNet and RealNet stochastic gradient descent was used as solver type with an intitial learning rate of 0.01 and reduction of the learning rate by a factor of 10 was taken for every 5 epochs within 15 epochs. Each network took about 10 minutes to train on Nvidia's DIGITS deep learning GPU training system.

III. Data Acquisition

A. Conveyor Belt Object Classification

Data was acquired through photographs taken from a Jetson mounted over a conveyor belt as items such as bottles and candy boxes were placed on a running conveyor belt and passed by the camera for capture. Each image is resized to fit the GoogleNet architecture which takes in 256x256 pixel images. The following Fig. 1 shows examples of the dataset.

Fig. 1. Sample of Conveyor Belt Data

B. Classify Number Abtraction

The data set used to train FlatModel was drawn from a subset of 3000 images generated by augmenting 50 computer drawn images per class. First, three classes of 50 images each was computer drawn. The class representing the number one contained images where each image had a single irregular shape of varied size such that each shape was filled in

black and randomly placed on a white background. The class representing the number two contained images where each image had two irregular shapes of varied size such that each shape was filled in black and placed randomly on a white background. The class representing the number three contained images where each image had three irregular shapes of varied size such that each shape was filled in black and randomly placed on a white background. For each class, 1000 images were generated from the 50 hand crafted seed images by randomized rotation, reflection, and distortion performed on the images using the augmentor package in python [8]. The code used to generate these images is given by the following snippet.

```
import Augmentor

p = Augmentor.Pipeline("/path/to/image")

p.random_distortion(probability=0.8,
    grid_width=4, grid_height=4, magnitude=8)
p.rotate90(probability=0.5)
p.rotate270(probability=0.5)
p.flip_left_right(probability=0.5)
p.flip_top_bottom(probability=0.5)

p.sample(1000)
```

A sample of the training data for FlatModel is given by the following Fig. 2.

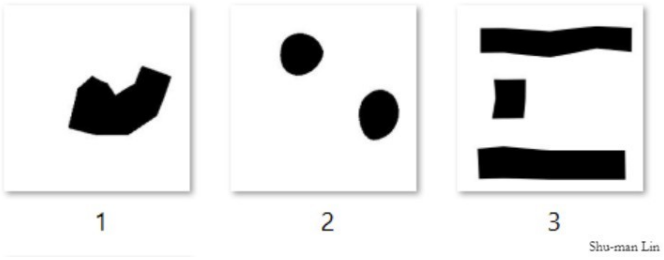

Fig. 2. Sample of training data for FlatModel

The data set used to train RealModel was a subset of an augmented data set of 1000 images per class where the seed classes consisted of 19 monochrome photographs of the single object class, 16 monochrome photographs of the double object class, and 15 monochrome photographs of the triple object class. For each class, 1000 images were generated by reusing the FlatModel augmentation code in which randomly selected seed photographs had randomized rotation, reflection, and distortion performed on them. A sample of the training data for RealModel is given by the following Fig. 3.

Fig. 3. Sample of training data for RealModel

For testing and training on two dimensional data, 15% of the total 2-d figure data generated by augmentation (3000 images) was set aside for validation while the rest was used to train FlatModel. A separate testing set generated by augmenting 15 computer drawn seed images of each class was made with a total of 100 images for each class. Call this 2-d, 300 image test set FlatTest. For testing and training on three dimensional data, 10% of the total 3-d object data generated by augmentation (3000 images) was set aside for validation and 10% of the total data was set aside for testing, the rest was used to train RealModel. Call this 3-d, 300 image test set RealTest. In addition to being tested on FlatTest, FlatModel was also tested on RealTest. Similarly, RealModel was tested on FlatTest. The following are examples of test set images from both FlatTest, Fig. 4, and RealTest, Fig. 5.

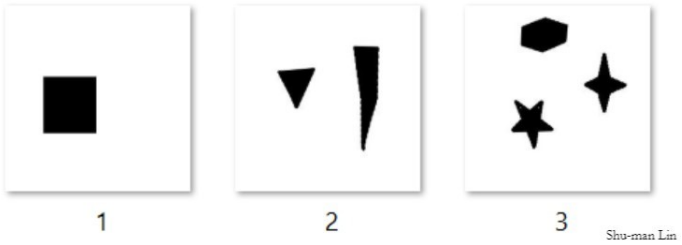

Fig. 4. Testing examples from FlatTest

Fig. 5. Testing examples from RealTest

IV. Results

A. Conveyor Belt Object Classification

A graphical representation of the loss on the training set and the loss and accuracy on the validation set as the network trains up to 10 epochs is given by the Fig. 6. Overall, loss trends towards zero and accuracy trends toward 100% as training progresses. However, accuracy falls to 75.4% upon testing.

The results of the conveyor belt object classification experiment include an inference time of approximately 5.5 ms over 10 runs and an overall accuracy of 75.4%. The output of the evaluate command is summarized in Fig. 7.

B. Classify Number Abstraction

A graphical representation of the loss on the training set and the loss and accuracy on the validation set as the FlatModel trains up to 15 epochs is given by the Fig. 8. Overall, loss trends towards zero and accuracy trends toward 100%.

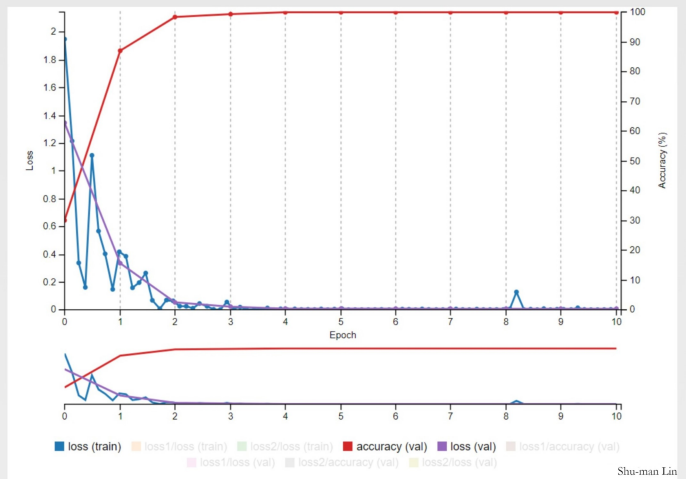

Fig. 6. Loss and accuracy graph for conveyor belt classification

```
Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20180221-134251-3cfc/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20180221-134251-3cfc/snapshot_iter_2370.caffemodel
output: softmax
iterations: 5
avgRuns: 10
Input "data": 3x224x224
Output "softmax": 3x1x1
name=data, bindingIndex=0, buffers.size()=2
name=softmax, bindingIndex=1, buffers.size()=2
Average over 10 runs is 5.48788 ms.
Average over 10 runs is 5.47563 ms.
Average over 10 runs is 5.48465 ms.
Average over 10 runs is 5.14734 ms.
Average over 10 runs is 4.93861 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14679  100 12363  100  2316    213     39  0:00:59  0:00:57  0:00:02  2586

Your model accuacy is 75.4098360656 %
Shu-man Lin
```

Fig. 7. Output of evaluate command

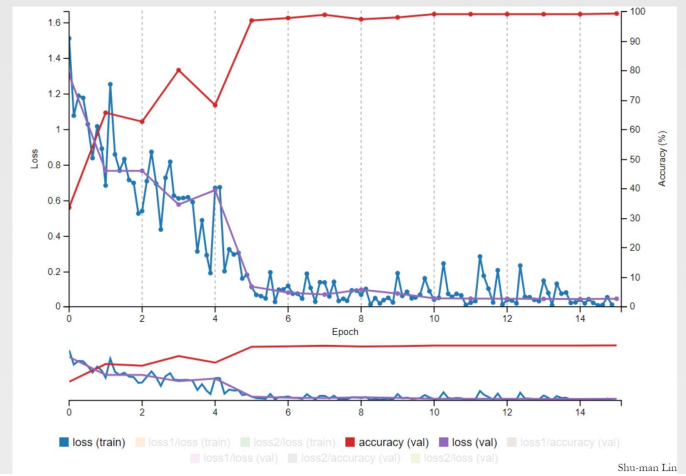

Fig. 8. Loss and accuracy graph for FlatModel

Tested on FlatTest dataset, FlatModel achieves a top-1 accuracy of 99.33% for classifying 300 images. Out of the 300 images set aside for testing, FlatModel only classifies two images of numerical representation incorrectly and obtains an inference time of about one minute. FlatModel produces the following results summarized in the confusion matrix in Fig. 9.

Confusion matrix

	1	2	3	Per-class accuracy
1	100	0	0	100.0%
2	1	98	1	98.0%
3	0	0	100	100.0%

Shu-man Lin

Fig. 9. Confusion matrix of FlatModel tested on FlatTest

Tested on RealTest dataset, FlatModel achieves a top-1 accuracy of 38.33% with a one minute inference time for classifying 300 images. Out of the 300 images set aside for testing, FlatModel only classifies 70/100 images of object class one correctly, 25/100 images of object class two correctly, and 20/100 images of object class three correctly. FlatModel produces the following results summarized in the confusion matrix in Fig. 10.

	1	2	3	Per-class accuracy
1	70	30	0	70.0%
2	69	25	6	25.0%
3	69	11	20	20.0%

Shu-man Lin

Fig. 10. Confusion matrix of FlatModel tested on RealTest

A graphical representation of the loss on the training set and the loss and accuracy on the validation set as the RealModel trains up to 15 epochs is given by the Fig. 11. Accuracy reaches 90% on the validation set at the 15th epoch and loss on both training and validation sets oscillates at a decreasing rate to about 20%.

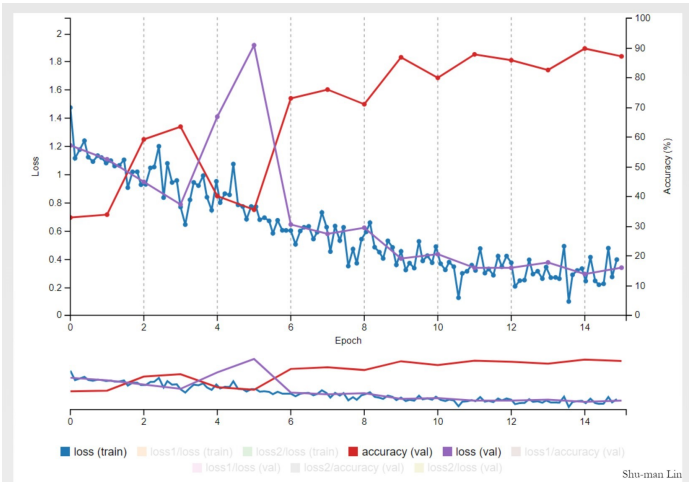

Fig. 11. Loss and accuracy graph for RealModel

Tested on FlatTest, RealModel reaches a top-1 accuracy rate of 95% with an inference time of one minute for 300 images. Out of the 300 images of the FlatTest set, RealModel misclassifies 2/100 images of the object class one, 12/100 images of the object class two, and 1/100 images of the object class three. RealModel produces the following results summarized in the confusion matrix in Fig. 12.

Confusion matrix

	1	2	3	Per-class accuracy
1	98	2	0	98.0%
2	2	88	10	88.0%
3	0	1	99	99.0%

Shu-man Lin

Fig. 12. Confusion matrix of RealModel tested on FlatTest

Tested on RealTest, RealModel reaches a top-1 accuracy rate of 87% with an inference time of about one minute for 300 images. Out of the 300 images of the RealTest set, RealModel misclassifies 5/100 images from object class one, 25/100 images from object class two, and 9/100 images from object class three. RealModel produces the following results summarized in the confusion matrix in Fig. 13.

	1	2	3	Per-class accuracy
1	70	30	0	70.0%
2	69	25	6	25.0%
3	69	11	20	20.0%

Shu-man Lin

Fig. 13. Confusion matrix of RealModel tested on RealTest

V. Discussion

A. Conveyor Belt Object Classification

For industry application of conveyor belt waste separation classifiers for recycling, high accuracy is more important than low inference time. A low accuracy rate of 75% is not exchangeable for a low inference time of less than 10 ms. As the rate of producing massive amounts of waste increases, faster methods of waste separation is required but impurities caused by low accuracy separation cannot be overlooked in favor of a low separation time.

B. Classify Number Abstraction

For applications in number estimation of highly dense populations, a low inference time is valued over high accuracy. For mathematical curiosities and human-machine comparisons, a high accuracy rate is valued over low inference time. Both FlatModel and RealModel tested with high accuracy in their respective dimensions. FlatModel tested well on FlatTest and RealModel tested well on RealTest. This suggests that number abstraction can be learned from training on a wide variety of examples within each class of number representation so that both what is a number and what is an object are learned simultaneously. Granted, generating the training and test sets of RealModel on such a small seed set should be regarded with caution since training and test sets may bear some redundancy. In this respect, training on a larger augmented real object dataset with a separate set of objects as testing set to see if accuracy still holds would provide more assurance that classifying on low number visual numerosity works.

On the other hand, FlatModel performed poorly on the RealTest dataset while RealModel performed better on FlatTest than its high accuracy performance on RealTest. This shows that simple two dimensional, planar perspective inference on complex two dimensional projections of three dimensional connected component does not always generalize while three dimensional connected component inference on number of planar, two dimensional shapes is a particular sub-case that is learned by modeling on three dimensional examples. Restricted to two translational degrees of freedom, the necessity to include depth as a feature to perceive connectedness of projected three dimensional objects cannot occur. Intrinsic in three dimensions is the hidden feature of number abstraction of two dimensional representations.

VI. Future Work

In the object classification exercise for conveyor belt items, an object classifier producing a 75% accuracy rate and a 5 ms inference time was built. It is unlikely that a classification system with 75% accuracy would be likely to be implemented in a production line regardless of the low inference time, unless a classifier for conveyor belt recycling is used as a quick first pass of separation before more refined methods are applied. In the number classification model that was built, visual numerosity trained on three dimensional objects performed well on both three dimensional objects and two dimensional irregular shapes, while visual numerosity trained on two dimensional irregular shapes performed poorly on three dimensional objects but performed well on irregular shapes. Utility was not in particular an outcome of the investigation, but perhaps the lesson that is learned is that collection of data from the more feature rich physical world that are properly augmented provides better results than poorly simulated data. In particular, flat simulated data cannot properly infer depth to recognize three dimensional components but three dimensional components have flat surfaces that project to irregular two dimensional shapes.

Future work includes testing the extent in which synthetic two dimensional data can mimic the three dimensional world well enough so that training on synthetic data will produce accurate results when tested on in three dimensional samples. One suggestion is to add noise to the FlatModel training set, add more two dimensional seed examples, and augment with more complexity and larger sample size to see if making such changes would produce a noticeable difference in classifying on photographed clusters of three dimensional objects. One can imagine whole world simulations that can take the place of real world interactions. We leave the future to ponder the question: to what extent can synthetic data represent the physical world?

References

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. *Gradient-Based Learning Applied to Document Recognition*. Proc. of the IEEE. November 1998.

[2] A. Krizhevsky, I. Sutskever, G. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. Advances in Neural Information Processing Systems 25. 2012.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. *Going Deeper with Convolutions*. arXiv preprint arXiv:1409.4842v1. September 2014.

[4] V. Lempitsky, A. Zisserman. *Learning to Count Objects in Images*. Advances in Neural Information Processing Systems 23. 2010.

[5] D. Onoro-Rubio, R.J. Lopez-Sastre. *Towards Perspective-Free Object Counting with Deep Learning*. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9911. Springer, Cham

[6] M. Rahnemoonfar, C. Sheppard. *Deep Count: Fruit Counting Based on Deep Simulated Learning*. Sensors (Basel) v. 17(4). URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5017387/. August 2016.

[7] X. Wu, X. Zhang, J. Du. *Two is Harder to Recognize than Tom: the Challenge of Visual Numerosity for Deep Learning*. arXiv preprint arXiv: 1802.05160v1. February 2018.

[8] M.D. Bloice, C. Stocker, A. Holzinger. *Augmentor: An Image Augmentation Library for Machine Learning*. arXiv preprint arXiv: 1708.04680. 2017.