AFRL-RI-RS-TR-2019-172

MACHINE INTELLIGENCE (MI) RIDER TASK 4: DYNAMIC ROUTING BETWEEN CAPSULES IMPLEMENTATION

AUGUST 2019

INTERIM TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2019-172 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
MICHAEL P. HARTNETT
Branch Chief, Advanced Planning &
Autonomous C2 Systems
Information Directorate

/ S /
JULIE BRICHACEK
Chief, Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
AUGUST 2019	INTERIM TECHNICAL REPORT	DEC 2018 – MAY 2019

4. TITLE AND SUBTITLE

MACHINE INTELLIGENCE (MI) RIDER TASK 4: DYNAMIC ROUTING BETWEEN CAPSULES IMPLEMENTATION

5a. CONTRACT NUMBER
IN-HOUSE/R2TX

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Milvio Franco
Simon Khan

5d. PROJECT NUMBER
S2IH

5e. TASK NUMBER
MI

5f. WORK UNIT NUMBER
00

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site/RISC
525 Brooks Road
Rome NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site/RISC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2019-172

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW2019-4095
Date Cleared: 22 AUG 2019

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper implemented Geoffrey E Hinton proposed implementation of Dynamic Routing between Capsules. The intent was to confirm the claims asserted by the author about the ability of Capsnet being rotational invariant to changes in the position of an image. Some claims validated to be true; however, other claims did not live to all the hype created. Results were collected by running extensive trials of Capsnet with different datasets such as the MNIST and Cifar10 datasets. Suggestion and directions are provided within the paper on future actions on how to move forward with this technology potential.

15. SUBJECT TERMS

Convolutional Neural Network, Dynamic capsules, non-linearity functions

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**MILVIO FRANCO**
U	U	U	UU	21	19b. TELEPHONE NUMBER (*Include area code*) **N/A**

TABLE OF CONTENTS

LIST OF FIGURES & TABLES

1.0 Summary

In our in-house research effort entitled “Dynamic Routing between Capsules”, we implemented and validated a Capsules Neural Network in python language that was described in (Geoffrey E Hinton, 2017) paper. The paper claims that the Capsnet model 1) Produces a discriminatively trained multi-layer capsule system that achieves state-of-the-art performance on MNIST Dataset, and is considerably better than CNN (Convolutional Neural Network) models at recognizing highly overlapping digits. 2) Performs image classification that is not affected by changes in the posture or rotation of an input image. 3) Uses a vector activity or vector length that varies as viewpoint varies eliminating the viewpoint variation from the vectors instantiation parameters; thus, preserving information loss unlike CNN during max pooling or sub-sampling. 4) Generalizes about an image with much less training data than CNN. Our approach implemented and validated these claims by implementing our own version of the Capsnet model. The outcome of this research was to gain inside knowledge on neural network rotational invariance and to lay a foundation for potential future research to improve the current state of Capsules Network architecture.

2.0 Introduction

The novelty behind the Dynamic Routing between Capsules paper by Geoffrey E. Hinton, is that the human vision ignores irrelevant details by using a sequence of fixation points to ensure that only a tiny fraction of the optic array is ever processed at the highest resolution. For example, suppose we have a tree whose hierarchical parts are labeled root, trunk, and crown (leaves, branches); our eyes do not have to process every detail of the tree to determine that it is indeed a tree; just by looking at the leaves and branches, the brain is able to infer that this is a crown. Subsequently, it is able to infer that the crown stands on the trunk leading to the conclusion that this must be a tree. The paper refers to these parts of a tree as entities. Moreover, it explains that introspection alone is not enough to understand how much of our knowledge of a scene is gathered from a sequence of observations, and how much is gathered from a single observation. The paper assumes that a single observation is able to tell us much more than just a single identified object and its properties. In the multi-layer visual model proposed, a tree-like structure is created on each observation, and it ignores the question of how these single observations parsed trees are coordinated over multiple observations. Each layer of the network is composed of small groups of neurons called capsules; each node such as leaves or trunk in the parsed tree corresponds to an active capsule. Through the implementation of an iterative dynamic routing process, each capsule will choose a capsule in the layer above to be its parent tree. For example, leaves and branches are part of a crown on a tree. The brain is able to infer whole objects such as the crown on the tree by just looking at its parts such as leaves and branches, and treat them as objects or entities. Furthermore, the instantiation parameters of an active capsule, or better yet

that of a vector, represents the various properties of a particular entity that is present in the image. For example, one can capture a triangle entity or a rectangle entity in the set of parameters contained within the vector. The properties of the vector capture the characteristics that make up that specific entity, such as its pose parameters e.g. position, size, orientation and, deformation, velocity, albedo, hue, and texture parameters, etc. In addition, by computing the length of the vector using the L2 vector norm, we are able to express the probability that the entity exists and also enable the orientation of the vector to represent the properties of that entity. As images change orientation along some axis, we are still able to classify the image as the same image because we are capturing the information of the vector orientation and preserving it and not discarding it unlike in CNN (Convolutional Neural Network) during Max Pooling. Moreover, the output is a normalized vector to represent a probability using the squashing function provided in equation 2. The squashing function is a non-linearity that serves a similar purpose to that of a neural network sigmoid, rectified linear unit (Relu), or tangent activation functions leaving the orientation of the vector unchanged, but reduced in magnitude.2.1 Dynamic Routing Between Capsules Architecture.

2.1 Dynamic Routing between Capsules Architecture

The authors of (Geoffrey E Hinton, 2017) the paper suggested to start out by leveraging the power of CNN by implementing two convolutional neural network layers. The first convolution layer has 256 filters with a kernel size of 9, a stride of 1 and a rectified linear unit (Relu) activation function. The second convolutional layer has 256 filters, kernel size of 9, and a stride of 2, and a rectified linear unit activation function. Next, we implemented the primary capsules with 32 maps of 6 x 6 grid matrix of 8 dimension vectors.

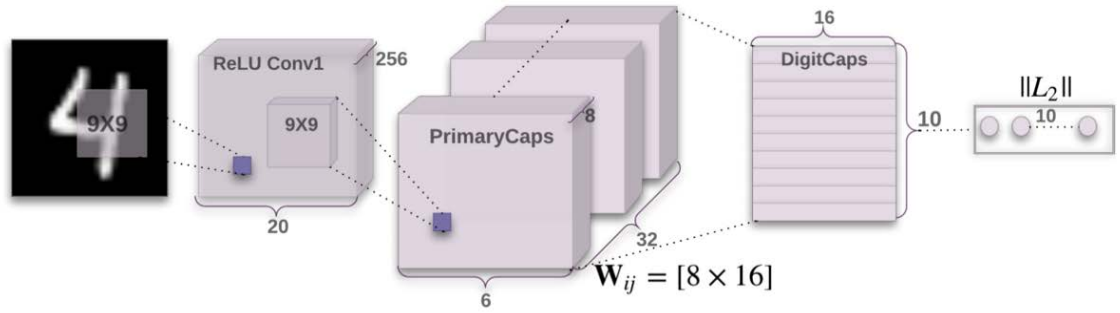


Figure 1: Capsnet Architecture

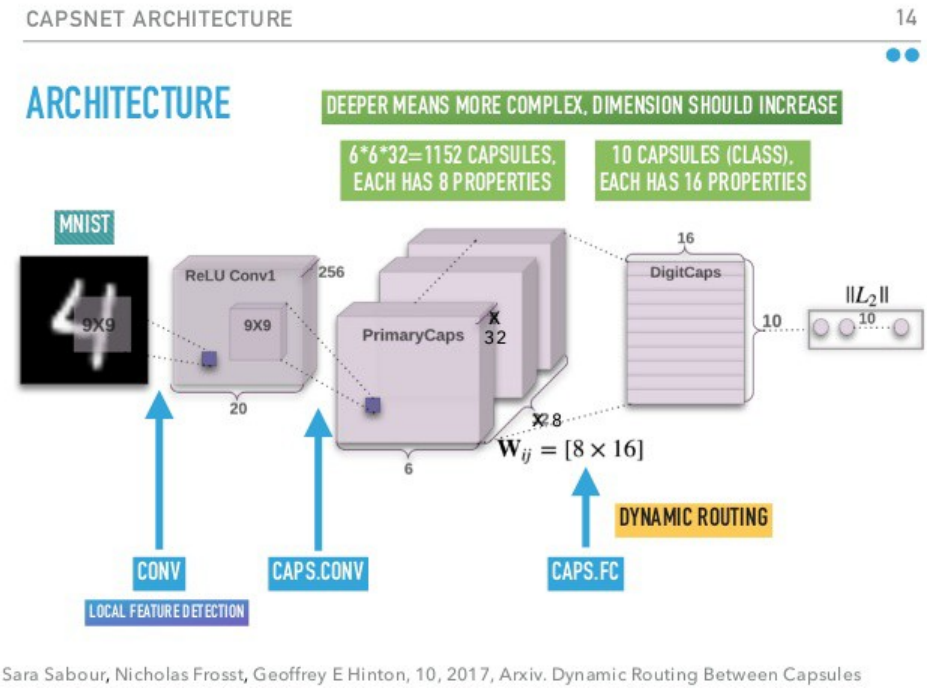
2.2 Primary Capsules

As noted in Figure 1, primary capsules are found in the layers after the first two convolutional layers. Primary capsules are the lowest level of multidimensional entities. Normally, one starts out with an image and tries to find what object it contains and what their instantiation parameters

are. In total, primary capsules have [32 x 6 x 6 x 8] capsule outputs and each capsule in the [6 x 6] grid is sharing their weights with one another other. The main function of primary capsules is to predict the output of the parent capsules; that is, they will predict the output of the digit capsules, which later will be compared with the actual output of the digit capsules during a dot product multiplication of the primary and digital capsule vectors.

2.3 Digit Capsules Layer

Digit Capsules is the third layer and has an output vector of 16 dimensions per digit class. As stated above, each of these capsules receives input from all the capsules in the layer below, when each primary capsule sends their output to a parent capsule via the dynamic routing algorithm. The digit capsules together with the primary capsules are responsible for carrying out the dynamic routing algorithm.

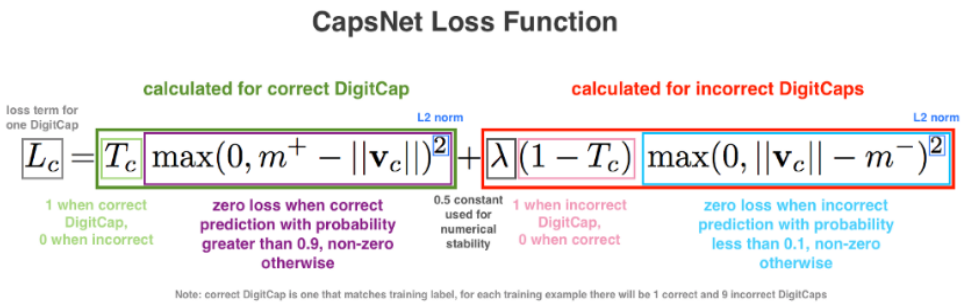


Sara Sabour, Nicholas Frosst, Geoffrey E Hinton, 10, 2017, Arxiv. Dynamic Routing Between Capsules

Figure 2 Capsnet Architecture

The output of the primary capsules gets sent to an appropriate parent in the layer above. Initially, the output is routed to all possible parents, but is scaled down by coupling coefficients produced through a softmax distribution function that sums to 1. For each possible parent, the primary capsules compute prediction vectors by multiplying their own outputs by a transformation weighted matrix. If this prediction has a large scalar product with the actual output of a possible parent, there is top-down feedback, which increases the coupling coefficient for that parent, and decreases it for other parents. This increases the likelihood contribution that the capsule makes to

that parents. Thus, further increasing the scalar product of the capsules prediction with the parents output. This is known as routing-by-agreement and is more effective than routing implemented by max-pooling, which allows neurons in one layer to ignore all, but the most active feature detector in a local pool in the layer below. No routing is necessary between convolutional layers and primary capsules layers; routing by agreement only occurs between primary capsules and digit capsules. The paper describes that all routing logits for the softmax function are initialized to zero. The authors also point out that they used the Adam optimizer with TensorFlow defaults parameters, including the exponentially decaying learning rate to minimize the sum of the margin losses Equation 1.



Equation 1

2.4 Margin loss for digit existence

A Margin loss for the digit existence is computed by using the length of the instantiation vector to represent the probability that a capsules entity exists. We would like the top-level capsule for digit class k to have a long instantiation vector if and only if that digit is present in the image. To allow for multiple digits, we use a separate margin loss L_k for each digit capsule, k:

$$L_k = T_k max(0, m^+ - ||v_k||)^2 + \lambda(1 - T_k) max(0, ||v_k|| - m^-)^2,$$

where $T_k = 1$ and if a digit of class k is present and $m^+ = 0.9$ and $m^- = 0.1$.

The lambda down-weighting of the loss for absent digit classes stops the initial learning from shrinking the lengths of the activity vectors of all the digit capsule. We make lambda = .5. The total loss is simply the sum of the losses of all digits capsules.

2.5 Reconstruction as a regularization method

The paper uses an additional reconstruction loss to encourage the digit capsules to encode the instantiation parameter of the input digit capsules. During training, one is told to mask out all but

the activity vector of the correct digit capsule. Then we use this activity vector to reconstruct the input image. The output of the digit capsule is fed into a decoder consisting of 3 fully connected layers that model the pixel intensities as described. One then minimizes the sum of the squared differences between the outputs of the logistic units and the pixel intensities. Then, one scales down this re-construction loss by 0.0005 so that it does not dominate the margin loss during training. The reconstruction from the 16 dimension vectors of the digit capsules are robust while keeping only important details.

3. 0 Methods, Assumptions, and Procedures

3.1 Methods

In this effort, we developed Dynamic Routing between Capsules method introduced by Hinton in his latest paper describing how neural network models can become rotational invariant robust to affine transformations. We looked to implement a direct approximation of the authors' description in the paper by following a methodical step-by-step approach using python

3.1.1 Dynamic routing algorithm

The dynamic routing algorithm is the heart of the (Geoffrey E Hinton, 2017) paper's implementation; its intent is to replace sub-sampling and route information from one layer to another. Depending on the number of iteration within this algorithm, the code we develop controls how fast a model start converging to a high accuracy rate during training. In this effort, we built a routing algorithm based on the pseudocode below. Our framework of choice for this implementation was TensorFlow along with Python3.6 programming language.

Procedure 1 Routing algorithm.

procedure ROUTING($\hat{\boldsymbol{u}}_{j|i}, r, l$)
  for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow 0$.
  for r iterations **do**
    for all capsule i in layer l: $\mathbf{c}_i \leftarrow \texttt{softmax}(\mathbf{b}_i)$
    for all capsule j in layer $(l+1)$: $\mathbf{s}_j \leftarrow \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}$
    for all capsule j in layer $(l+1)$: $\mathbf{v}_j \leftarrow \texttt{squash}(\mathbf{s}_j)$
    for all capsule i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$
  return $\mathbf{v}_j$

Figure 3: Algorithm

Steps:

2. For all capsuled in both primary and digital capsules initialize the routing weight b_{ij} to zero.

3. Enter the loop.

4. Compute the softmax probability distribution of the b_{ij} routing weights = the coupling coefficient c_i.

5. Multiply the coupling coefficient ci with the primary capsules prediction $u_{j|i}$ then perform a weighted sum and obtain the digital capsules output s_j.

6. Squash the result of step 5 between 0 and 1 = V_j.

7. Calculate the dot product between lower level capsules output and high level capsules output and add results to the initial routing weight.

3.1.2 Squash Function

Capsnet implementation uses a squash function for our activation function in place of a Relu, or typical sigmoid or tanh non-linear activation as mentioned in the paper. We want the length of the output vector to a capsule to represent the probability that the entity represented by the capsule is present in the current input. The non-linear squashing function ensures that vectors get shrunk to almost zero length and long vectors gets shrunk to a length slightly below 1.

Equation 2

$$\mathbf{v}_j = \underbrace{\frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}}_{\text{additional "squashing"}} \underbrace{\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}}_{\text{unit scaling}}$$

3.1.3 Convolutional Neural Network

Even though we are trying to overcome some inherent deficiencies of CNN sub- sampling, introducing the concept of capsules to capture e.g. pose information leverages the power of CNN to divide our image input Dataset into a number of desired feature maps. It is done to the point, where we would need sub-sampling to make up for the information loss induced from its use. The first two layers of the Capsnet model consist of two convolutional layers without sub-sampling.

3.1.4 Capsule Network

The capsule network consists of the primary and digital capsules. In our implementation, we defined both primary and digital capsules layers. The idea behind capsule relies on using

instantiation parameters such as pose (position, size, and orientation), velocity, albedo, hue, texture parameters to preserve information about the model. Through this approach we are able to make our model not only translational invariant, but also robust to affine transformations such as rotation and stretching or repositioning of the image. In order to accomplish this objective, we reshaped the output from the CNN layer into vectors without applying sub-sampling so we can infer the instantiation parameters aforementioned. The primary capsule layer predicts the outputs of their parent capsules in the digital capsules upper layer by multiplying their own reshaped output obtained from the CNN layer by a transformation matrix that contains all the pose, deformation, velocity, albedo, hue, texture information.

3.1.5 Label Masking

In this paper we used label masking introduced by Geoffrey Hinton (2017) to simplify training and testing with correct digits. Mask has the shape as the digit capsule output vector and is zero everywhere in the vector except on the target digit; by multiplying digit capsule output with the mask, we get the input to the decoder; however, we are not able to use label during test time. We use the predicted value to be masked instead. Therefore, we use a Boolean placeholder such as lambda to mask the label during training, and predict value during test time. If the condition is true, we use the label to be masked and if it is false, then we use the predicted value. When the output from the digit capsule is masked with respect to the current label of the image, the activated label vector passed through fully connected layers of 512, 1024, 784(28X28), which results in the original image.

3.2 Assumptions

During this project, there were many assumptions that we set out to prove according to the authors' claims in the paper. We conducted a series of sensitivity analysis in parallel with the most prominent hyper-parameters of Capsule Network (Capsnet) such as the number of iterations on the dynamic routing algorithm that controls how fast a model converges, different type of optimizers with different learning rates, as well as altering the number of convolutional layers and the number of capsules. Irrespective of which hyper-parameters or optimizers we chose, we expected better results for MNIST Dataset than CIFAR-10 Dataset as expressed in the authors' paper. The assumption that did not validate against the authors' claims was the ability of Capsnet to be rotational invariant after we applied some affine transformation such as performing horizontal flip, stretching the image slightly to the right and left as stated in the Capsnet paper. We expected the Capsnet model to be able to handle more aggressive affine transformations such as 180 degree rotation, but it failed being transformation invariant to the pronounced

perturbations. There may be issues with the ensemble techniques, which is very difficult to attain. It may be the reason of not getting exact result for CIFAR-10 Dataset.

3.3 Procedures

3.3.1 Capsnet with MNIST

In this effort, we set up Capsnet with the MNIST Dataset for training and testing. The approach taken consisted of training the Capsnet's model with both perturbed affined and non-perturbed regular MNIST Datasets. Moreover, for the testing, we utilized both perturbed and non-perturbed Datasets to prove the robustness of the Capsnet model underlining architecture. The model was trained using up to 10 epochs as 10 epochs proved to be more than enough to achieve a training accuracy of 99.9 %. Selective affine transformations were performed ensuring not to perform rotational transformation more than 180 degrees. We managed not to compromise the integrity of the inferencing engine by ensuring not to confuse a 6 for 9 and vice versa. Our rotation parameters stayed within less than 180 degrees and were mostly performed as a horizontal flip.

3.3.2 Capsnet with CIFAR-10

The CIFAR-10 Dataset was also explored and deployed within the Capsnet model to measure the model efficiency against a more complex Dataset CIFAR-10. It contains a fairly generous amount of noise level in the background. The Capsnet model with CIFAR-10 Dataset required over 50 training epochs to converge before it would start over fitting. Tensorflow too was used to process CIFIAR10 Capsnet model. CIFAR-10Capsnet model required similar parameters as the parameters for the MNIST Capsnet model. We used 1152 primary capsules of 8 dimensions and 10 digital capsules of 16 dimensions. We adjusted the input parameters 32 x 32 x 3 channels for CIFAR 10 Dataset vs 28 x 28 x 1 channel for MNIST Dataset due to CIFAR 10 being a color image with respect to MNIST (black and white). We then fed this information input to the convolutional layers in the model.

3.3.3 Convolutional Neural Network (CNN) base Model

A base model implementation of CNN was essential to contrast the performance of our basic Capsnet model against a baseline. The baseline model took 3 layers of 256, 256, and 128 channels. Each layer had a kernel of 5 x5 and a stride of 1. The last two layers were followed by two fully connected layers of size 328, 192. The last fully connected layer was connected with a dropout regularization method to a 10 class softmax probability output function. We trained and

tested the baseline on a MNIST Dataset of two pixel shifted both to the right and left with Adam optimizer. The baseline model had 35.4 M parameters and Capsnet had 8.2 M parameters.

4.0 Results and Discussion

4.1 CNN Base Model Confusion Matrix Results Testing with and without data augmentation for the MNIST Dataset

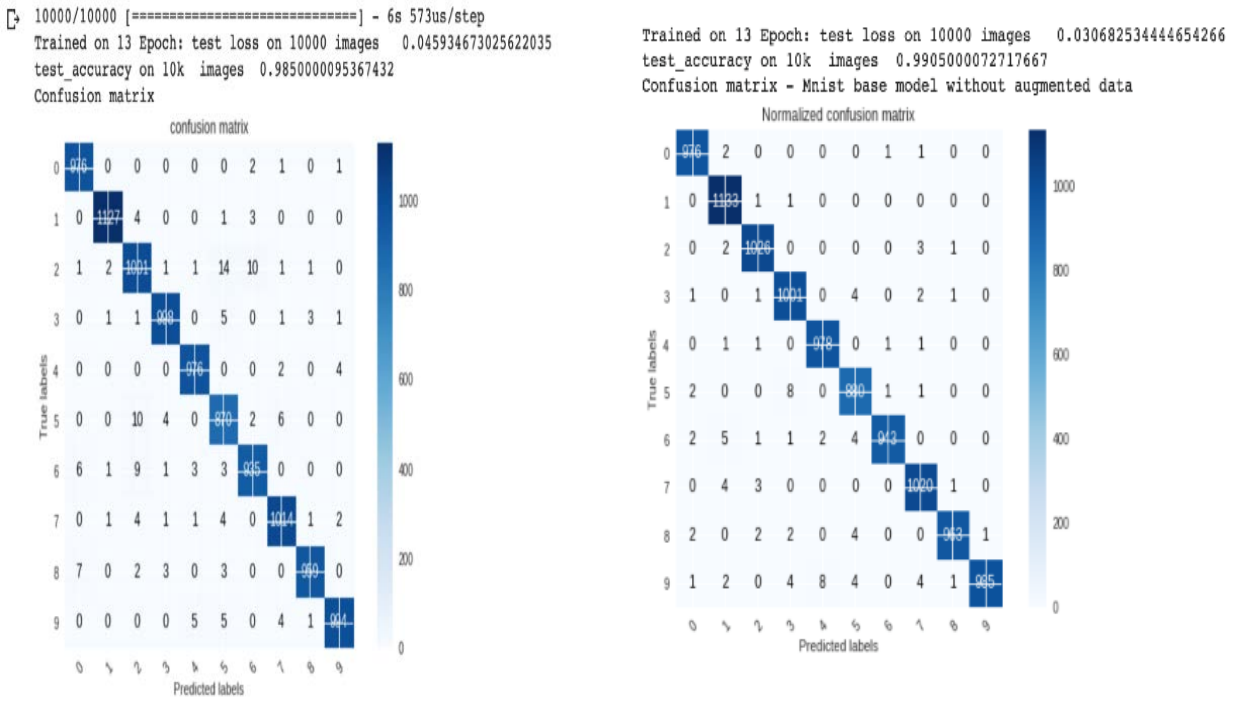


Figure 4: Augmented Data Accuracy 98% Non-Augmented data accuracy 99%

The Convolutional Neural Network (CNN) base model with the configuration proposed in the paper managed to score almost same as the base model with non-augmented data using Capsnet architecture for the MNIST Dataset. This shows that a simple model of CNN is robust enough to withstand a controlled data perturbation experiment given the perturbation parameters presented in the paper.

4.2 CNN Base Model Training and Validation Accuracy Graph with Augmented and Non-augmented MNIST Dataset

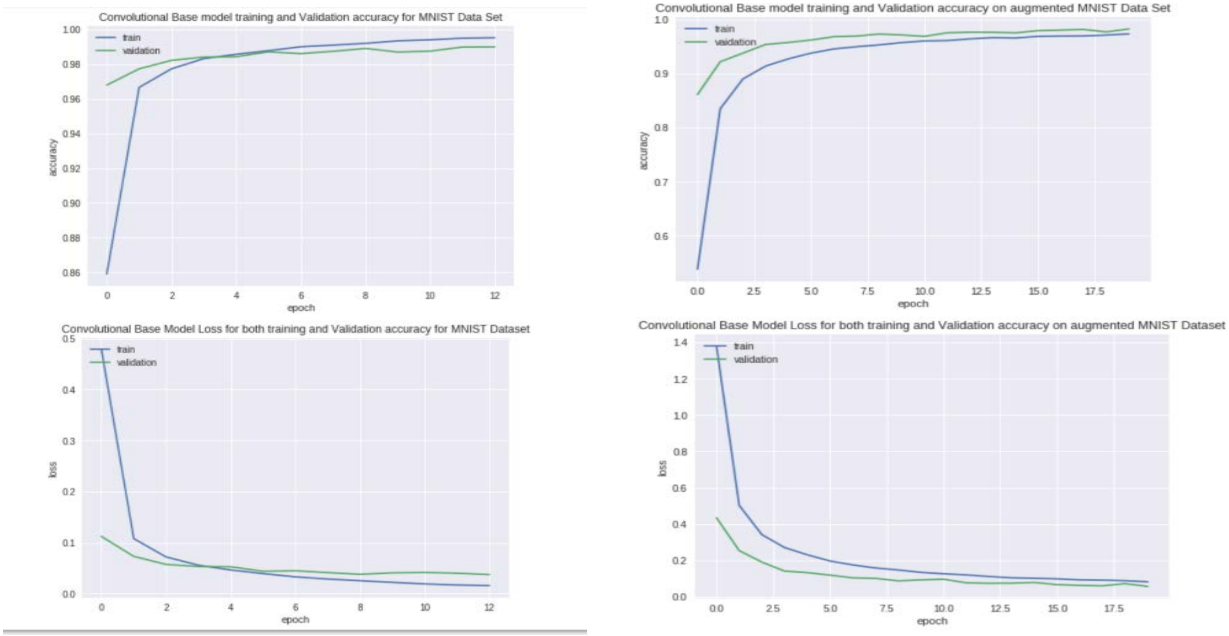


Figure 5: 98% Training Accuracy 99% Validation Accuracy

4.3 Capsnet with Augmented Data on Training and Non-augmented data on Testing

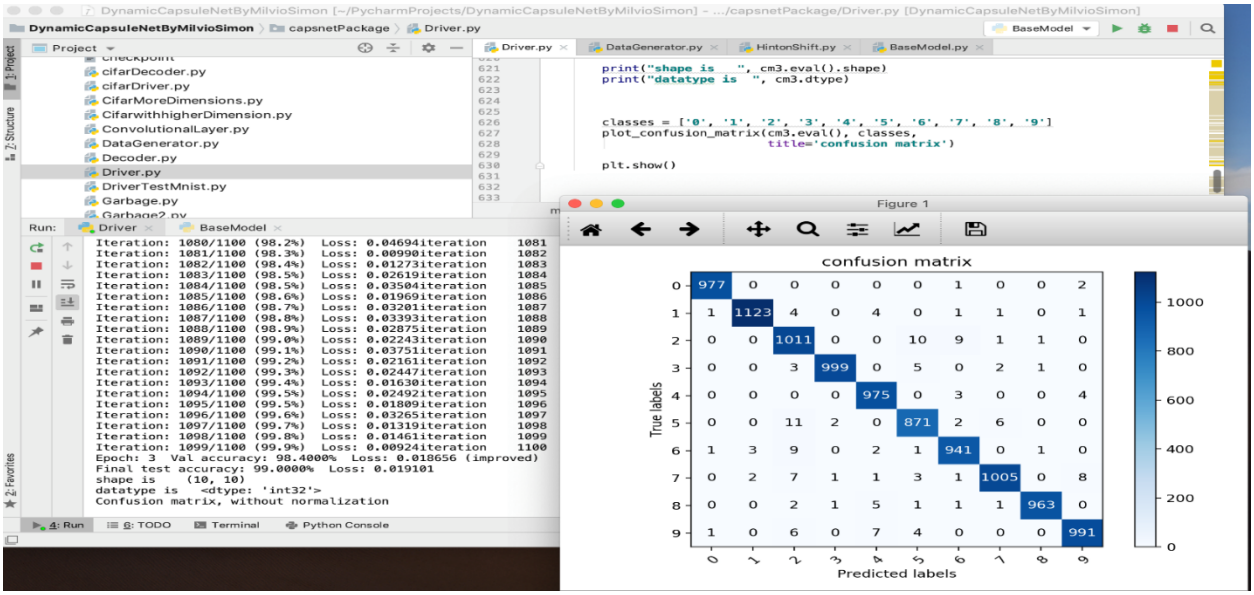


Figure 6: Training Accuracy 98.4 % Test Accuracy 99%

The results above show that the Capsnet model achieved an accuracy rate of 98.4% with MNIST perturbed Dataset and a test accuracy rate of 99.0%. The results are clear that the Capnet model does equally well compared to a simple convolutional neural base model. An observation was

that Capsnet is able to learn a lot faster than CNN; however, CNN will close the gap after 15 epochs.

4.4 Capsnet Trained on Augmented MNIST and Validated on Non-Augmented Data Using a Hyper-parameter of Routing as R = 2 Iterations within the Dynamic Routing Algorithm.

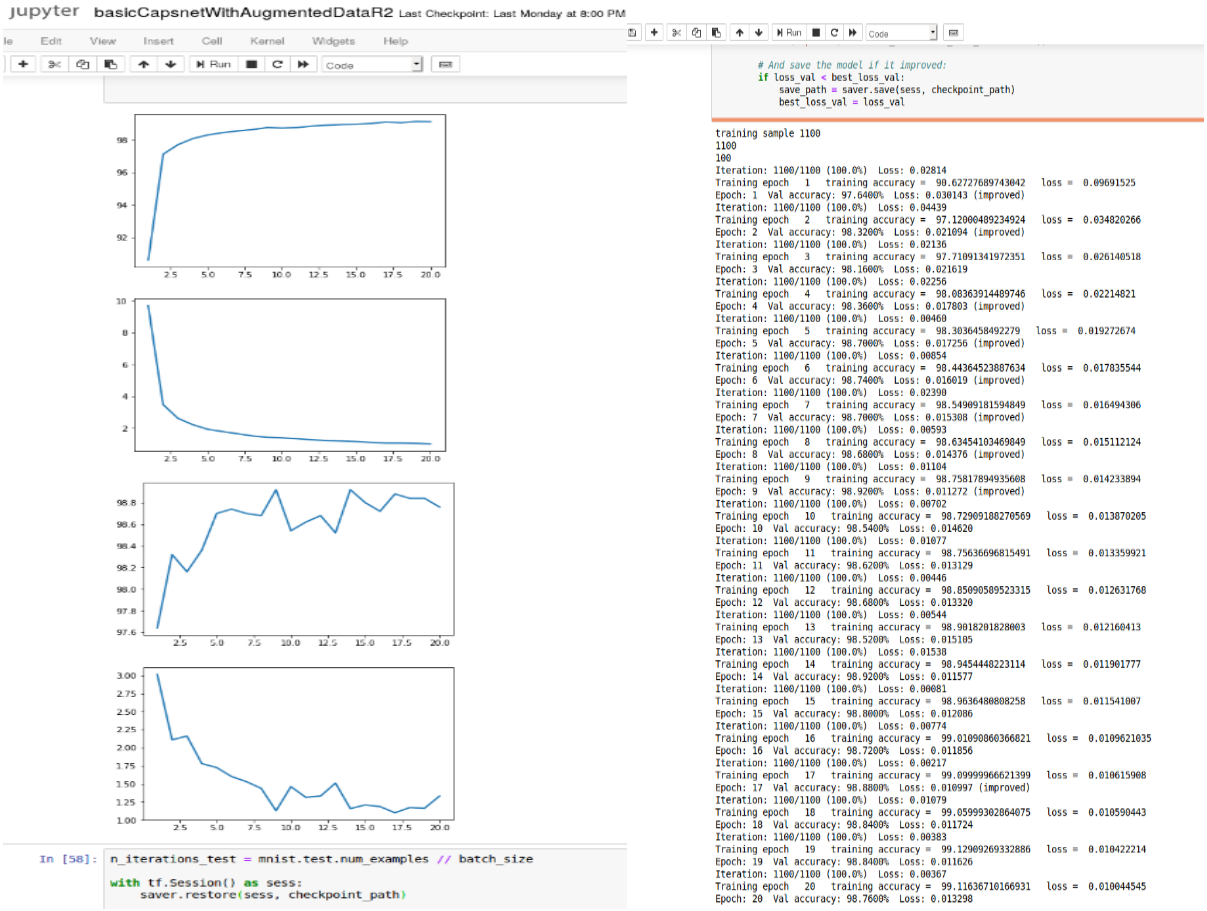


Figure 7: Training Accuracy 99.12 %, Validation Accuracy 98.76%

From the observation above, it is clear that the Capsnet model even with a low iteration hyper-parameter of 2 is able to converge to a 98% accuracy in the 2nd Epoch; this shows that the model is able to learn fairly fast. The Capsnet model proves that it is data invariant resistant as long as it trains on the perturbed data; the same applies to CNN base model.

4.5 Basic Capsnet Trained on Non-Augmented MNIST Data, Validated and Tested on Augmented MNIST Dataset R = 1, Epoch 20.

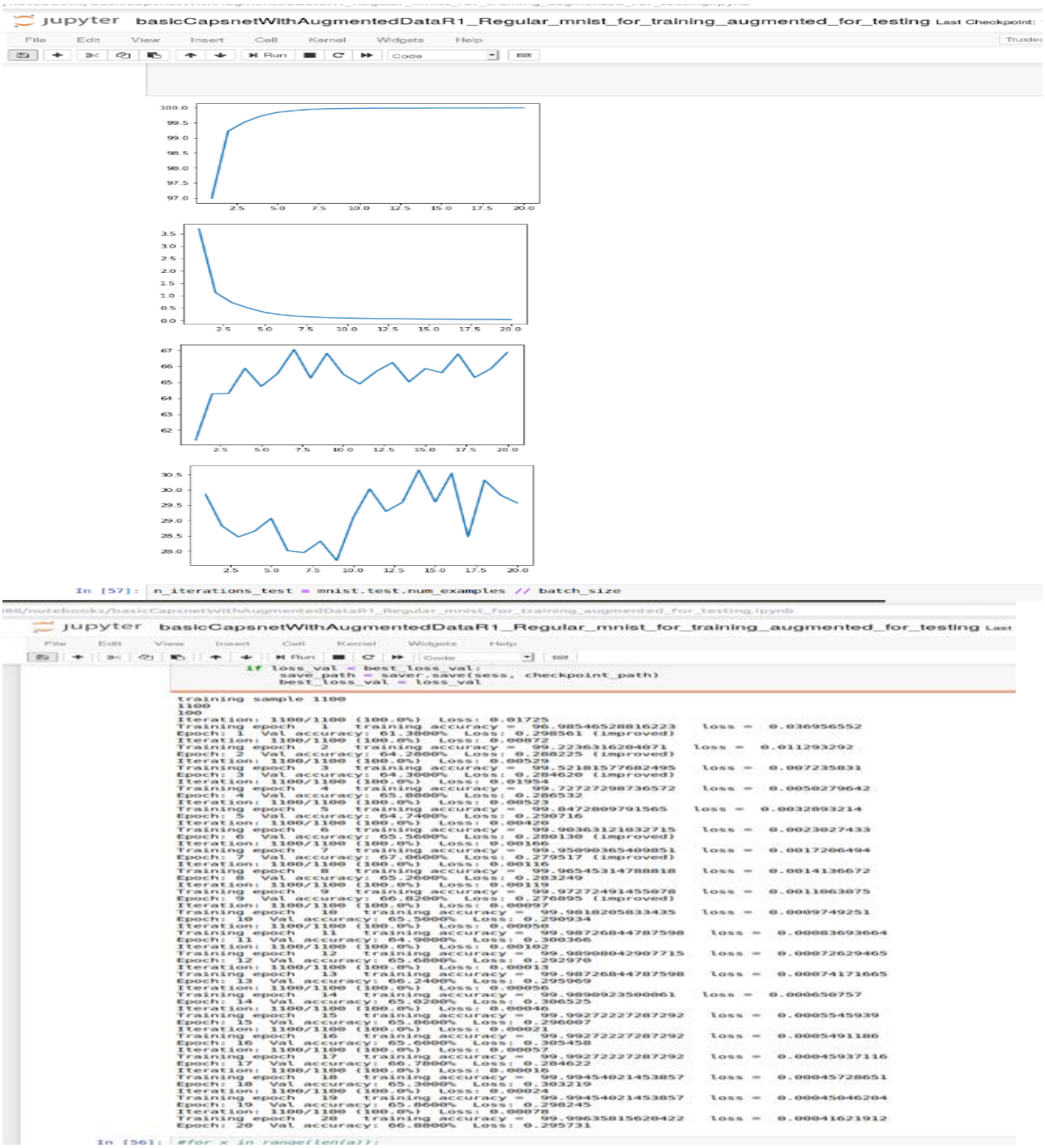


Figure 8: Training Accuracy = 99.99 % Validation Accuracy = 66.88%, Testing Accuracy = 10.81%

The results above show that the MNIST model does very well as long as we train the model on non-augmented data; however, when it is time for testing how robust it is to the perturbed data testing, it performs quite poorly.

4.6 Summary of Capsnet model performance for different epochs, number of routing iterations, and augmented vs non-augmented Dataset.

Table 1

	Aug-Train	Aug- Test	Training Accuracy	Testing Accuracy	Epoch	Routing
Capsnet	No	Yes	99.9	10.81	20	1
Capsnet	Yes	No	98.9	98.88	20	1
Capsnet	Yes	No	99.11	98.77	20	2
Capsnet	Yes	No	98.98	98.81	20	3
Capsnet	Yes	No	98.58	98.65	20	5
Capsnet	Yes	No	98.45	98.14	20	6

The table above represents Capsnet architecture with different augmentations, training and testing accuracies, epochs and routings. One observation we can take away is that the number of routing iteration stops making a significant difference after 3 epochs. In addition, training on non-augmented Dataset produces slightly better results than augmented. The testing accuracy is quite poor for Capsnet model under augmented testing and non-augmented training.

4.7 Augmented Training Data and Non-Augmented Testing Data for CIFAR-10

Table 2

Routing	Train Accuracy %	Testing Accuracy %	Training Loss	Testing Loss	Epoch
2	66.46	63.03	0.236	0.257	80
4	66.98	62.88	0.233	0.258	80
5	67.7	62.34	0.231	0.234	80

It’s not a surprise that the Capsnet Model performs poorly for the CIFAR-10 Dataset. The CIFAR-10 Dataset is a realistic and far more complicated Dataset than the MNIST Dataset. CIFAR-10 Dataset has a lot of background noise that the model needs to account for; our model had a lower score than the Hinton’s model; though, our model do not use ensemble technique used by Hinton’s model.

5.0 Conclusion

During this research and implementation endeavors, we were able to build a framework to validate Hinton claims successfully. We were able to generate almost the same results as the Hinton claims on the testing accuracy of the augmented MNIST Dataset with the code that we developed. For the CIFAR-10 Dataset, we were not able to duplicate the authors’ results. Our accuracy rate was 63% before it would start overfitting. In addition, no meaningful results were observed for the CIFAR-10model during training until we would reach 50 epochs. In addition, some take away knowledge obtained from this experiment conducted was that Capsnet and CNN are perturbation invariant to small affine transformation when trained on perturbed data but not to large data perturbation; thus, Capsnet is able to learn a lot faster than CNN, but CNN catches up after some number of epochs. We also learned that the routing algorithm controls the convergence rate and time it takes to run one epoch. An increase in the number of routing iterations of more than 3 has an exponential effect on the time as it takes to complete one epoch. Finally, Capsnet performs slightly better than CNN with Augmented MNIST Dataset, but this small difference could make all the difference when dealing with a much larger or complex Dataset as the underlining technology for the model becomes more mature in the future. A recommendation would be usage of dynamic routing between capsules with other deep neural networks to see if that improves the accuracy rate of CIFAR-10 Dataset higher than what it is shown here.

List of Symbols, Abbreviations, and Acronyms

CAPSNET	Capsule Network
CNN	Convolutional Neural Network

Bibliography

Geoffrey E Hinton, S. S. (2017, 11 7). Dynamic Routing Between Capsules. *7 Nov 2017*. Toronto, Canada.