Google

Brain Toronto

Capsules and stuff and vision

Making use of the work of Sara Sabour, Geoff Hinton, Yao Qin, and me Nick Frosst

Convolutions - x y translation invariance built in

Deep Image prior

(a) Ground truth (b) SRResNet [19], **Trained**

(c) Bicubic, **Not trained** (d) Deep prior, **Not trained**

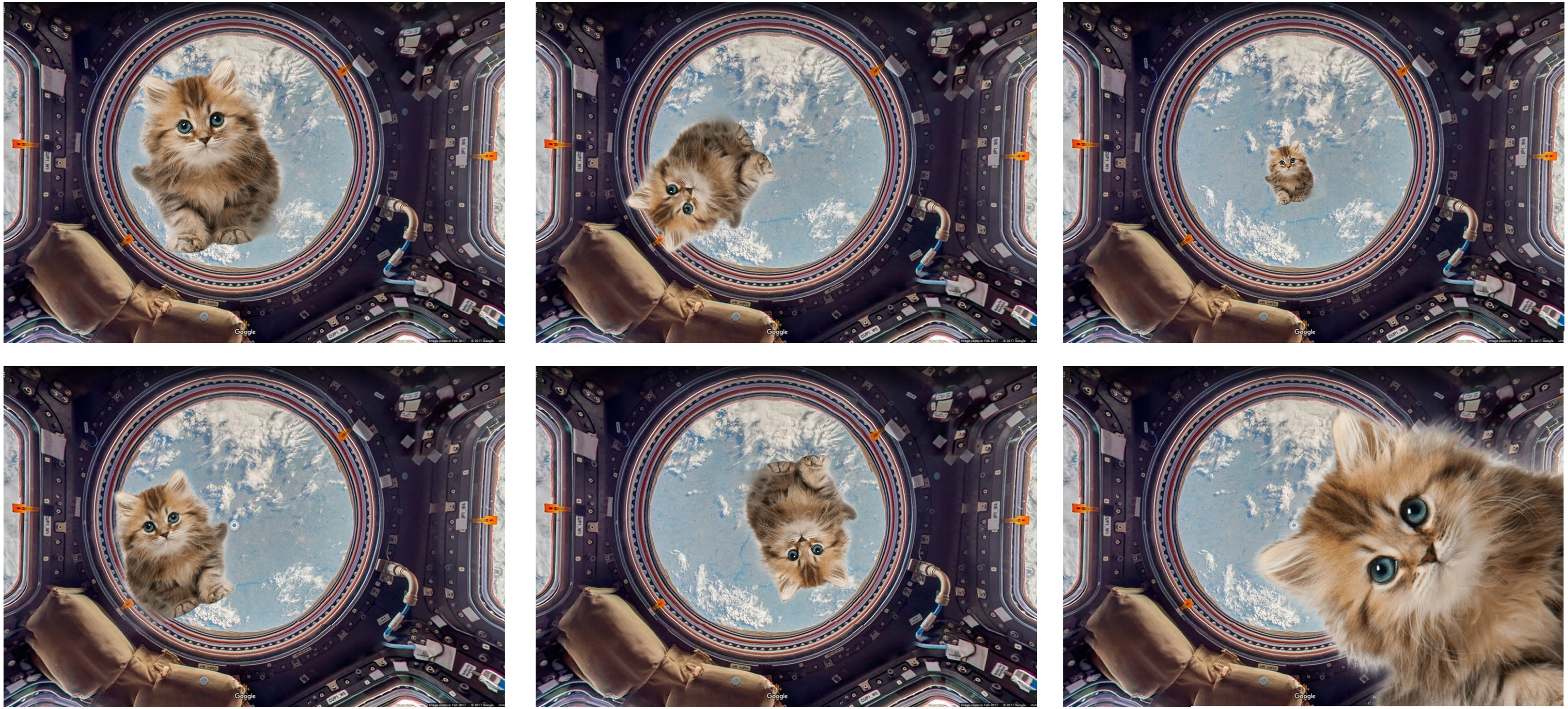
You don't really need to train convnets
to get good image results

This isn't really true for Resnets

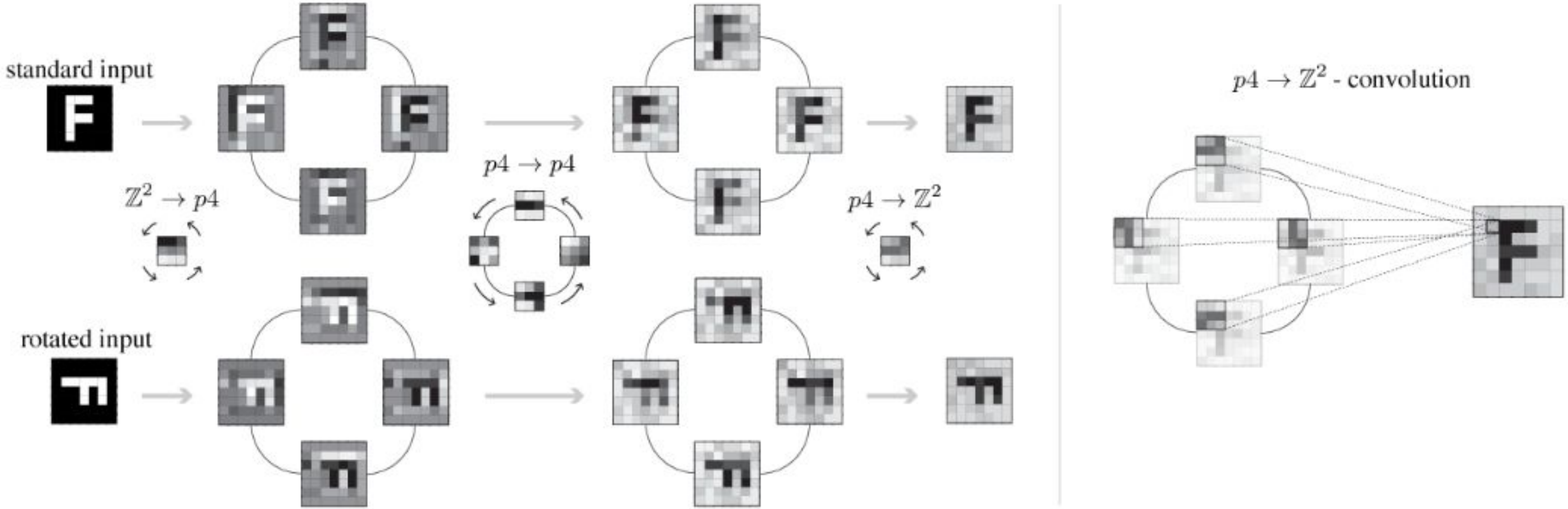
(a) Input (white=masked) (b) Encoder-decoder, depth=6

(d) Encoder-decoder, depth=2 (e) ResNet, depth=8

But what about other transformations?



Rotational Equivierient Conv Nets

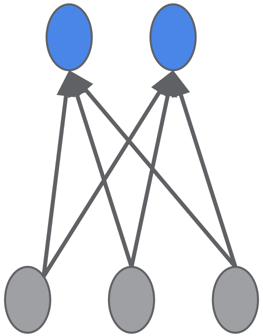


But there are many more transformations than rotation

Google

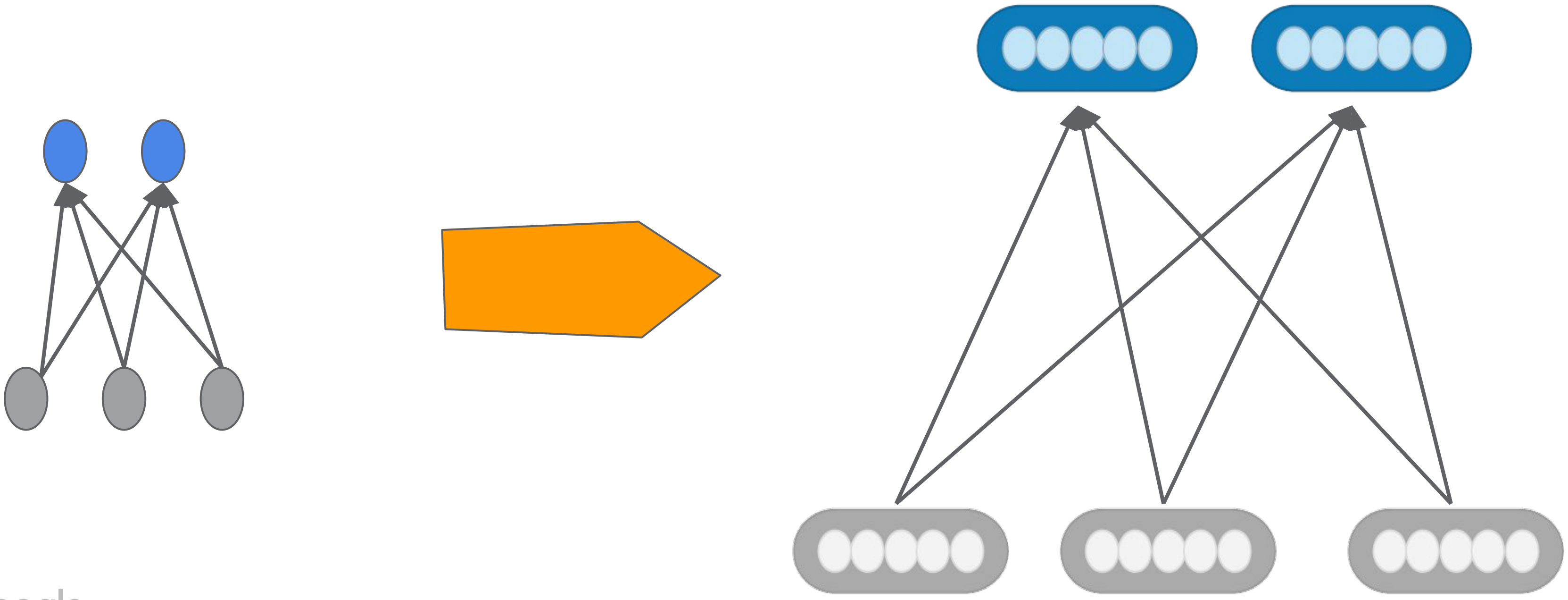
Capsule Network

- Build a network that has units which output more than just a single value:



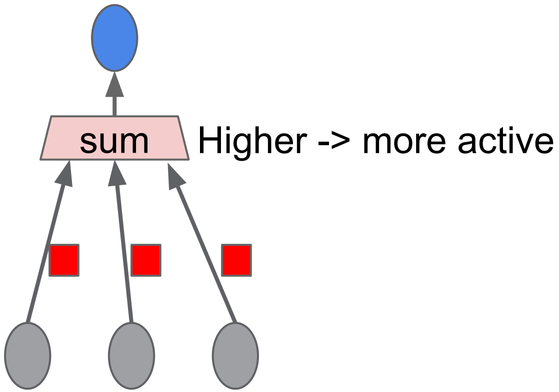
Capsule Network

- Build a network that has units which output more than just a single value:
  - Capsule: A group of units.
    - Is it present or not? (Activation)
    - How it is present? (Instantiation parameter)



Capsule Network

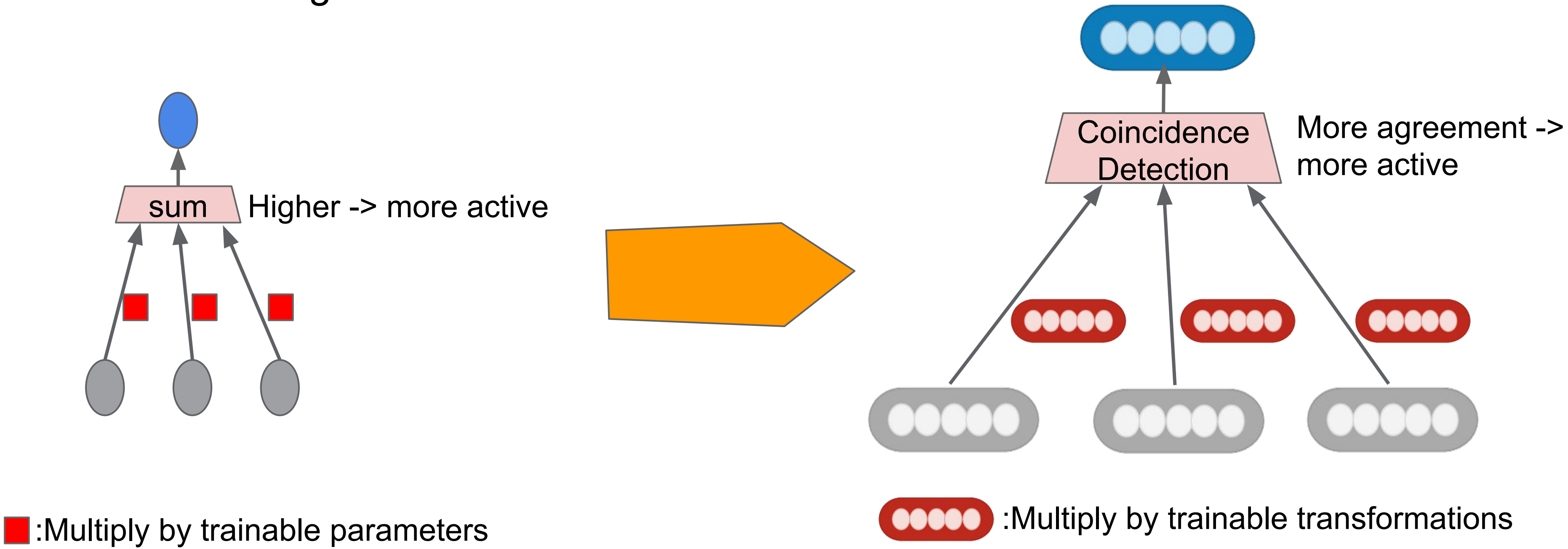
- Activate by agreement between incoming predictions, instead of pattern matching.



:Multiply by trainable parameters

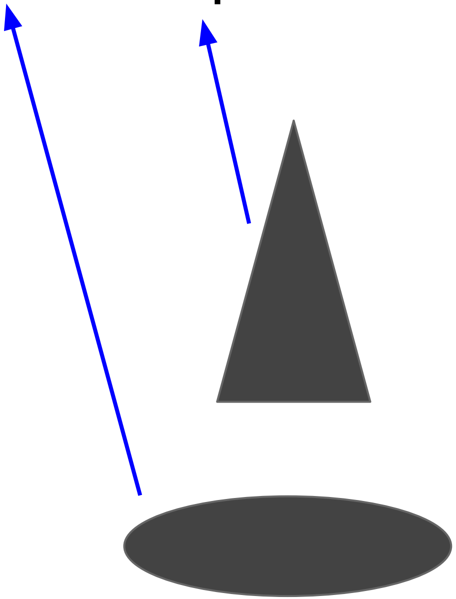
Capsule Network

- Activate by agreement between incoming activations, instead of pattern matching.

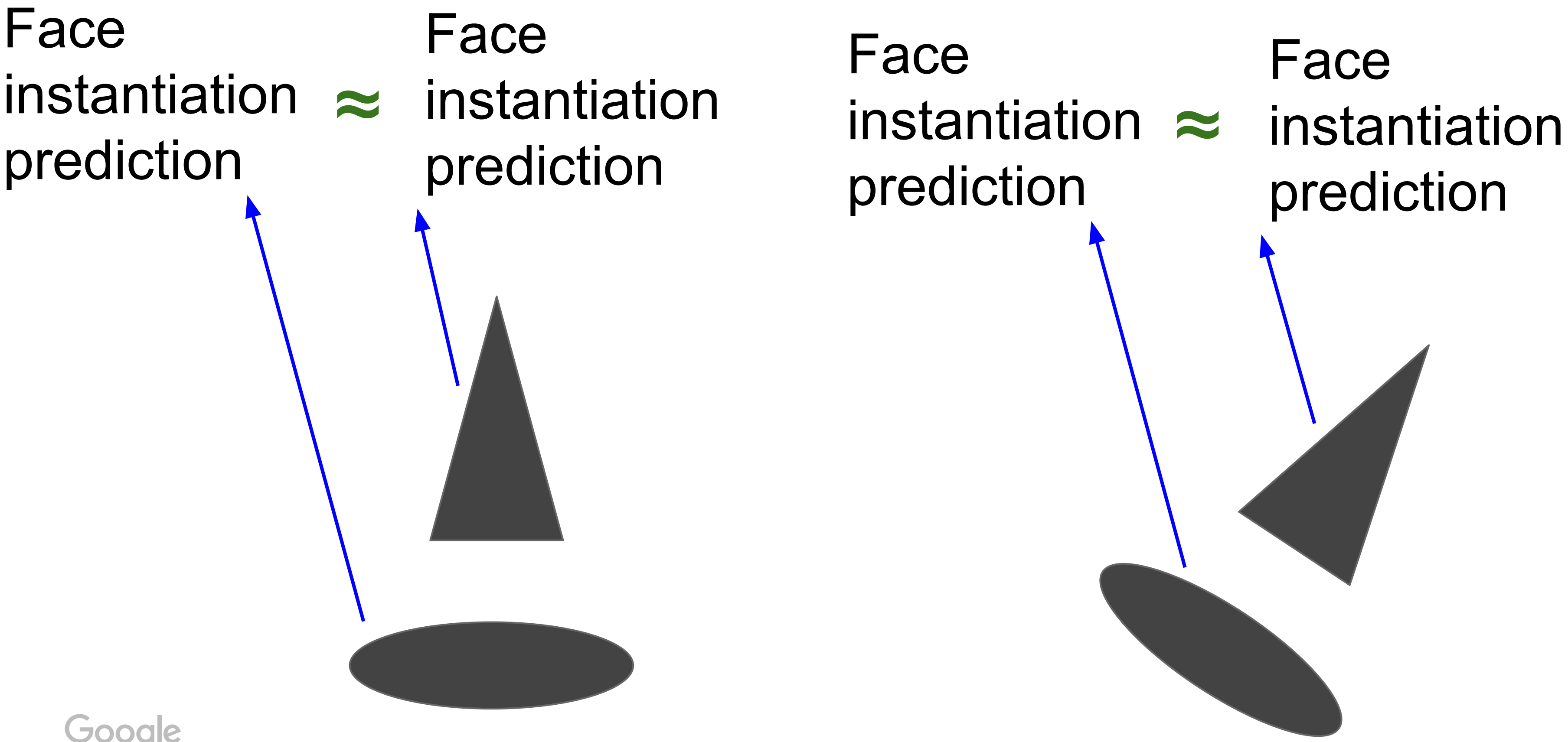


Coincidence Detection

Face instantiation prediction ≈ Face instantiation prediction

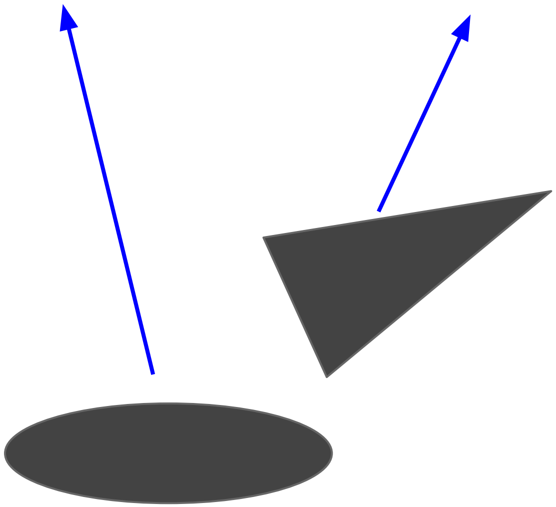


Coincidence Detection

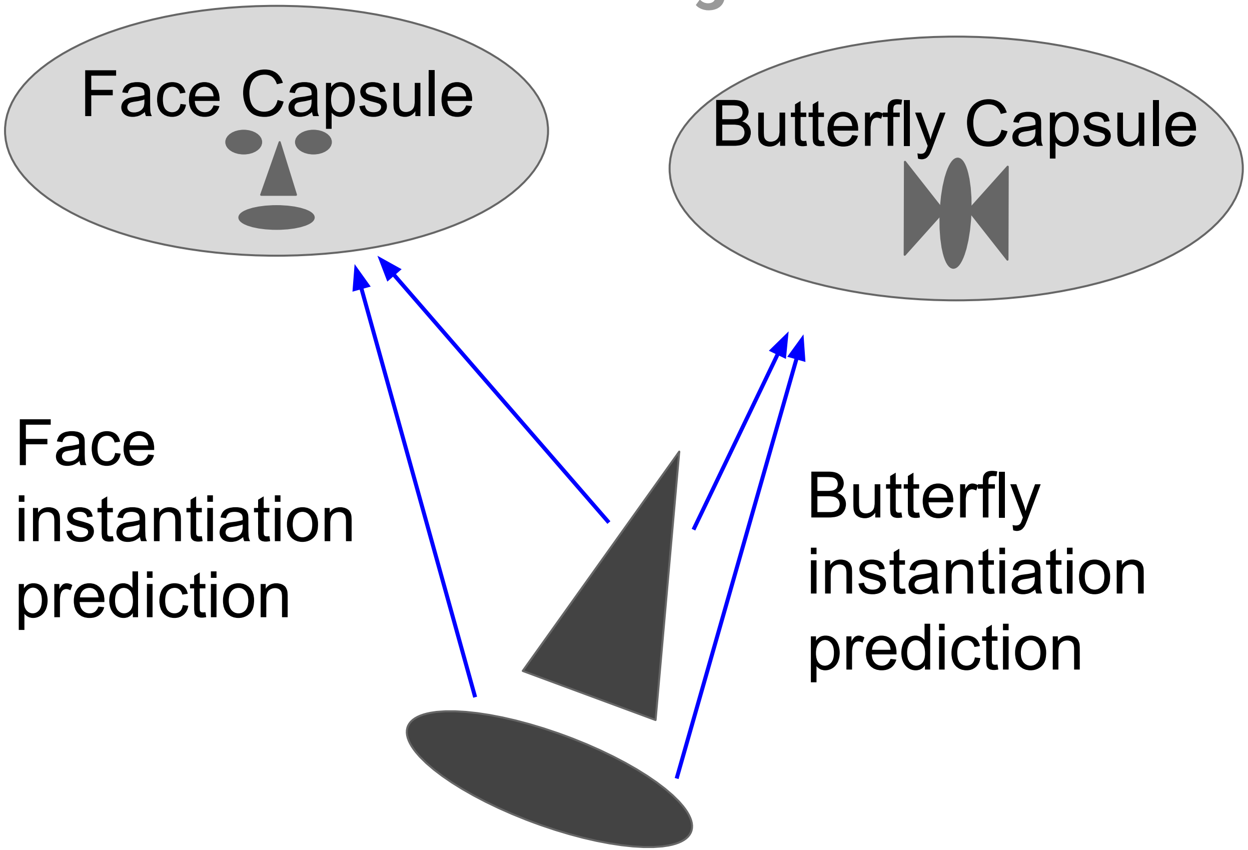


Coincidence Detection

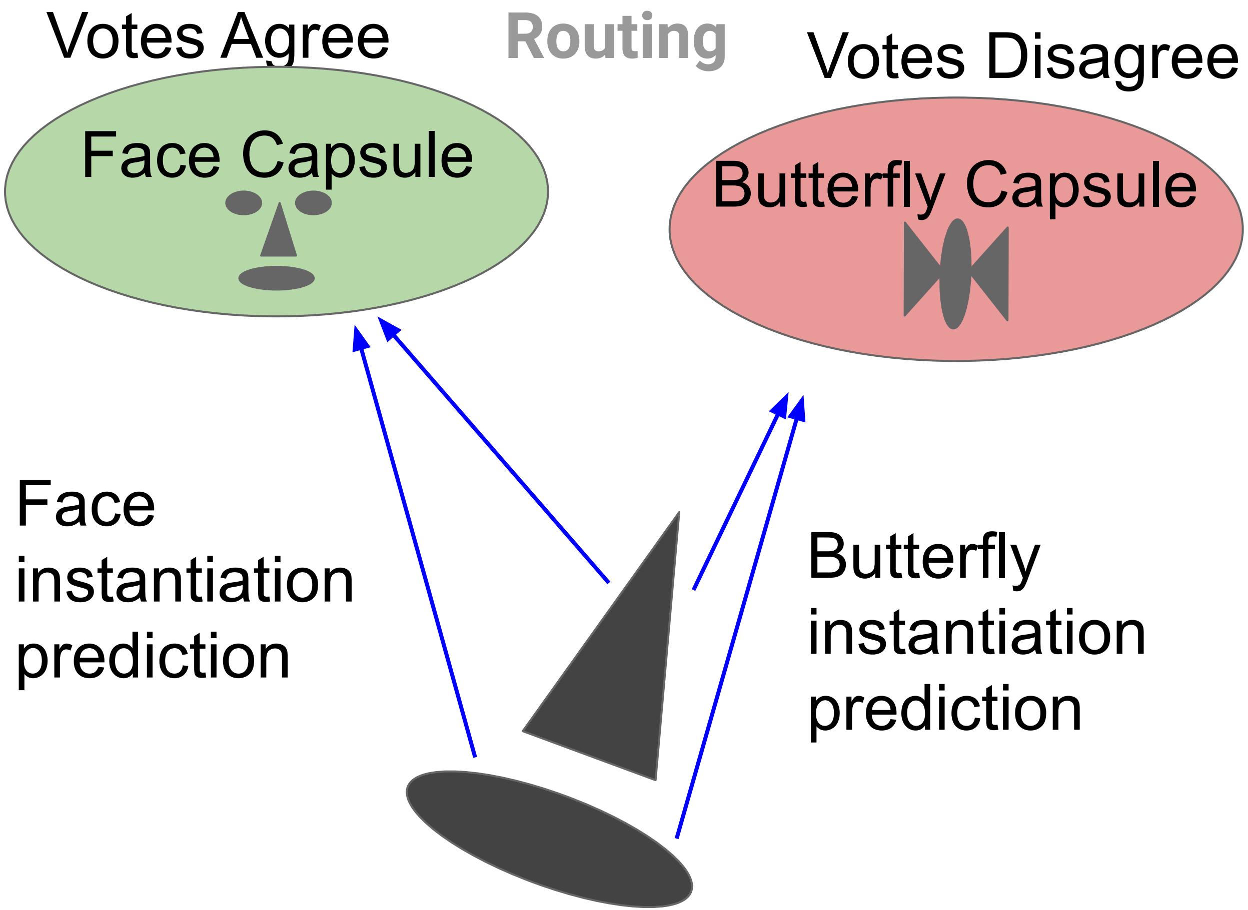
Face instantiation prediction $\neq$ Face instantiation prediction



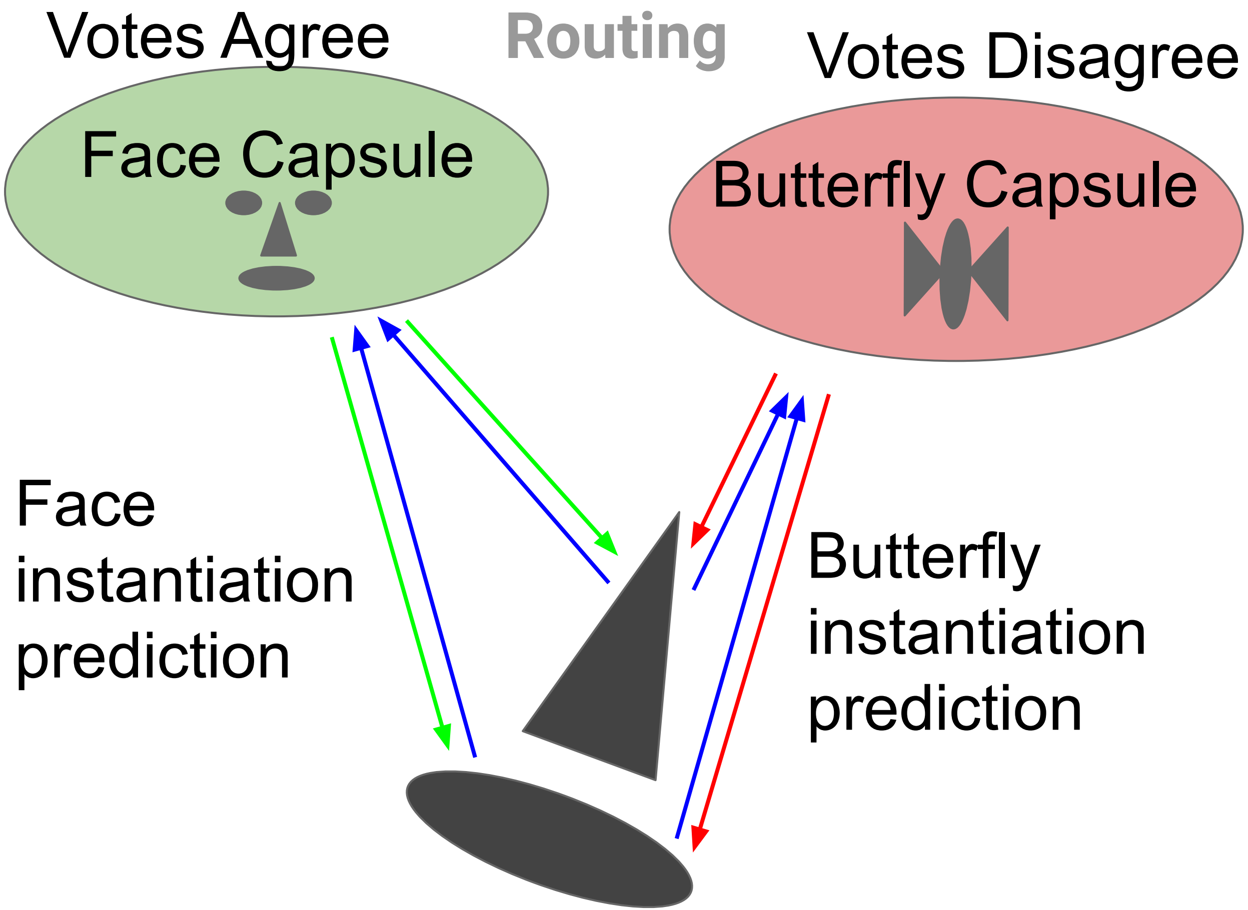
Routing



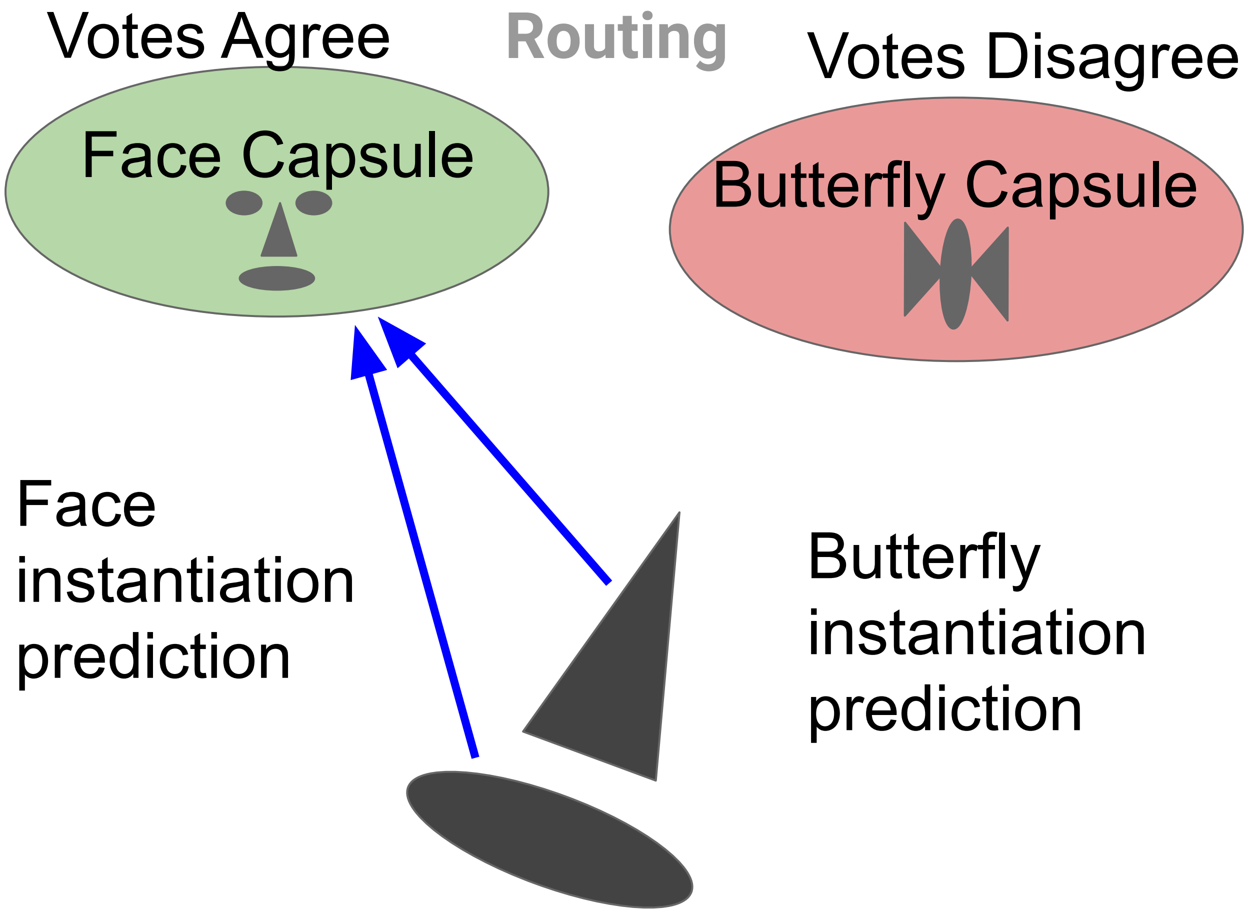
Face Capsule
Butterfly Capsule
Face instantiation prediction
Butterfly instantiation prediction



Votes Agree
Routing
Votes Disagree
Face Capsule
Butterfly Capsule
Face instantiation prediction
Butterfly instantiation prediction

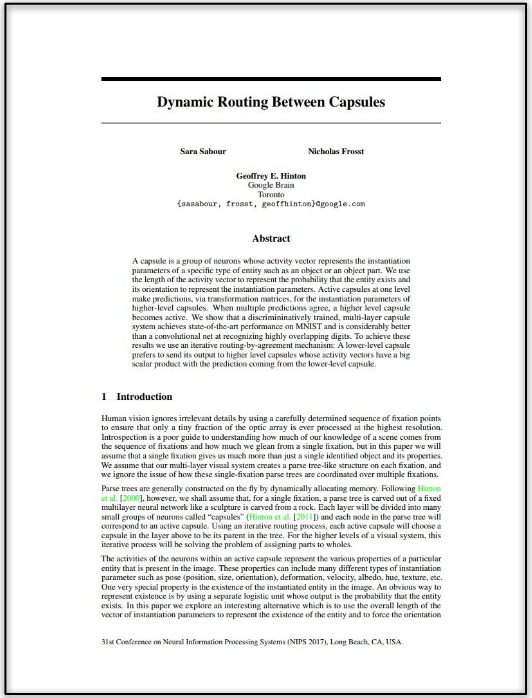


Votes Agree
Routing
Votes Disagree
Face Capsule
Butterfly Capsule
Face instantiation prediction
Butterfly instantiation prediction


Votes Agree
Routing
Votes Disagree
Face Capsule
Butterfly Capsule
Face instantiation prediction
Butterfly instantiation prediction

Capsule implementations - 2 takes

- **Commonalities between the two**
  - Capsules output more than a single value
  - Capsules are activated if there is agreement between incoming activity patterns
  - Capsules have dynamic routing algorithm between then to improve coincidence detection



Dynamic Routing Between Capsules

Sara Sabour **Nicholas Frosst**

Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com

Abstract

A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part. We use the length of the activity vector to represent the probability that the entity exists and its orientation to represent the instantiation parameters. Active capsules at one level make predictions, via transformation matrices, for the instantiation parameters of higher-level capsules. When multiple predictions agree, a higher level capsule becomes active. We show that a discrimininatively trained, multi-layer capsule system achieves state-of-the-art performance on MNIST and is considerably better than a convolutional net at recognizing highly overlapping digits. To achieve these results we use an iterative routing-by-agreement mechanism: A lower-level capsule prefers to send its output to higher level capsules whose activity vectors have a big scalar product with the prediction coming from the lower-level capsule.

1 Introduction

Human vision ignores irrelevant details by using a carefully determined sequence of fixation points to ensure that only a tiny fraction of the optic array is ever processed at the highest resolution. Introspection is a poor guide to understanding how much of our knowledge of a scene comes from the sequence of fixations and how much we glean from a single fixation, but in this paper we will assume that a single fixation gives us much more than just a single identified object and its properties. We assume that our multi-layer visual system creates a parse tree-like structure on each fixation, and we ignore the issue of how these single-fixation parse trees are coordinated over multiple fixations.

Parse trees are generally constructed on the fly by dynamically allocating memory. Following Hinton et al. [2000], however, we shall assume that, for a single fixation, a parse tree is carved out of a fixed multilayer neural network like a sculpture is carved from a rock. Each layer will be divided into many small groups of neurons called "capsules" (Hinton et al. [2011]) and each node in the parse tree will correspond to an active capsule. Using an iterative routing process, each active capsule will choose a capsule in the layer above to be its parent in the tree. For the higher levels of a visual system, this iterative process will be solving the problem of assigning parts to wholes.

The activities of the neurons within an active capsule represent the various properties of a particular entity that is present in the image. These properties can include many different types of instantiation parameter such as pose (position, size, orientation), deformation, velocity, albedo, hue, texture, etc. One very special property is the existence of the instantiated entity in the image. An obvious way to represent existence is by using a separate logistic unit whose output is the probability that the entity exists. In this paper we explore an interesting alternative which is to use the overall length of the vector of instantiation parameters to represent the existence of the entity and to force the orientation

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

MATRIX CAPSULES WITH EM ROUTING

Geoffrey Hinton, Sara Sabour, Nicholas Frosst
Google Brain
Toronto, Canada
{geoffhinton, sasabour, frosst}@google.com

ABSTRACT

A capsule is a group of neurons whose outputs represent different properties of the same entity. Each layer in a capsule network contains many capsules. We describe a version of capsules in which each capsule has a logistic unit to represent the presence of an entity and a 4x4 matrix which could learn to represent the relationship between that entity and the viewer (the pose). A capsule in one layer votes for the pose matrix of many different capsules in the layer above by multiplying its own pose matrix by trainable viewpoint-invariant transformation matrices that could learn to represent part-whole relationships. Each of these votes is weighted by an assignment coefficient. These coefficients are iteratively updated for each image using the Expectation-Maximization algorithm such that the output of each capsule is routed to a capsule in the layer above that receives a cluster of similar votes. The transformation matrices are trained discriminatively by backpropagating through the unrolled iterations of EM between each pair of adjacent capsule layers. On the smallNORB benchmark, capsules reduce the number of test errors by 45% compared to the state-of-the-art. Capsules also show far more resistance to white box adversarial attacks than our baseline convolutional neural network.

1 INTRODUCTION

Convolutional neural nets are based on the simple fact that a vision system needs to use the same knowledge at all locations in the image. This is achieved by tying the weights of feature detectors so that features learned at one location are available at other locations. Convolutional capsules extend the sharing of knowledge across locations to include knowledge about the part-whole relationships that characterize a familiar shape. Viewpoint changes have complicated effects on pixel intensities but simple, linear effects on the pose matrix that represents the relationship between an object or object-part and the viewer. The aim of capsules is to make good use of this underlying linearity, both for dealing with viewpoint variations and for improving segmentation decisions.

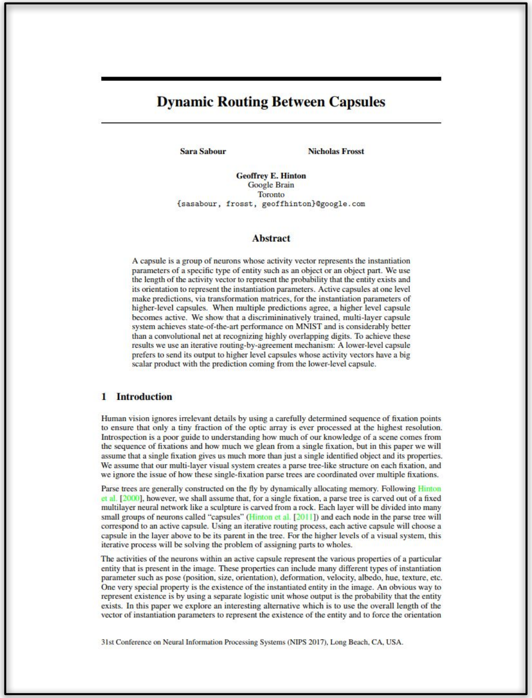
Capsules use high-dimensional coincidence filtering: a familiar object can be detected by looking for agreement between votes for its pose matrix. These votes come from parts that have already been detected. A part produces a vote by multiplying its own pose matrix by a learned transformation matrix that represents the viewpoint invariant relationship between the part and the whole. As the viewpoint changes, the pose matrices of the parts and the whole will change in a coordinated way so that any agreement between votes from different parts will persist.

Finding tight clusters of high-dimensional votes that agree in a mist of irrelevant votes is one way of solving the problem of assigning parts to wholes. This is non-trivial because we cannot grid the high-dimensional pose space in the way the low-dimensional translation space is gridded to facilitate convolutions. To solve this challenge, we use a fast iterative process called "routing-by-agreement" that updates the probability with which a part is assigned to a whole based on the proximity of the vote coming from that part to the votes coming from other parts that are assigned to that whole. This is a powerful segmentation principle that allows knowledge of familiar shapes to derive segmentation, rather than just using low-level cues such as proximity or agreement in color or velocity. An important difference between capsules and standard neural nets is that the activation of a capsule is based on a comparison between multiple incoming pose predictions whereas in a standard neural net it is based on a comparison between a single incoming activity vector and a learned weight vector.

Google

Capsule implementations - 2 takes

- **Commonalities between the two**
    - Capsules output more than a single value
    - Capsules are activated if there is agreement between incoming activity patterns
    - Capsules have dynamic routing algorithm between them to improve agreement detection



Dynamic Routing Between Capsules

Sara Sabour Nicholas Frosst

Geoffrey E. Hinton
Google Brain
Toronto
{sasabour, frosst, geoffhinton}@google.com

Abstract

A capsule is a group of neurons whose activity vector represents the instantiation parameters of a specific type of entity such as an object or an object part. We use the length of the activity vector to represent the probability that the entity exists and its orientation to represent the instantiation parameters. Active capsules at one level make predictions, via transformation matrices, for the instantiation parameters of higher-level capsules. When multiple predictions agree, a higher level capsule becomes active. We show that a discrimininatively trained, multi-layer capsule system achieves state-of-the-art performance on MNIST and is considerably better than a convolutional net at recognizing highly overlapping digits. To achieve these results we use an iterative routing-by-agreement mechanism: A lower-level capsule prefers to send its output to higher level capsules whose activity vectors have a big scalar product with the prediction coming from the lower-level capsule.

1 Introduction

Human vision ignores irrelevant details by using a carefully determined sequence of fixation points to ensure that only a tiny fraction of the optic array is ever processed at the highest resolution. Introspection is a poor guide to understanding how much of our knowledge of a scene comes from the sequence of fixations and how much we glean from a single fixation, but in this paper we will assume that a single fixation gives us much more than just a single identified object and its properties. We assume that our multi-layer visual system creates a parse tree-like structure on each fixation, and we ignore the issue of how these single-fixation parse trees are coordinated over multiple fixations.

Parse trees are generally constructed on the fly by dynamically allocating memory. Following Hinton et al. [2000], however, we shall assume that, for a single fixation, a parse tree is carved out of a fixed multilayer neural network like a sculpture is carved from a rock. Each layer will be divided into many small groups of neurons called "capsules" (Hinton et al. [2011]) and each node in the parse tree will correspond to an active capsule. Using an iterative routing process, each active capsule will choose a capsule in the layer above to be its parent in the tree. For the higher levels of a visual system, this iterative process will be solving the problem of assigning parts to wholes.

The activities of the neurons within an active capsule represent the various properties of a particular entity that is present in the image. These properties can include many different types of instantiation parameter such as pose (position, size, orientation), deformation, velocity, albedo, hue, texture, etc. One very special property is the existence of the instantiated entity in the image. An obvious way to represent existence is by using a separate logistic unit whose output is the probability that the entity exists. In this paper we explore an interesting alternative which is to use the overall length of the vector of instantiation parameters to represent the existence of the entity and to force the orientation

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Published as a conference paper at ICLR 2018

MATRIX CAPSULES WITH EM ROUTING

Geoffrey Hinton, Sara Sabour, Nicholas Frosst
Google Brain
Toronto, Canada
{geoffhinton, sasabour, frosst}@google.com

ABSTRACT

A capsule is a group of neurons whose outputs represent different properties of the same entity. Each layer in a capsule network contains many capsules. We describe a version of capsules in which each capsule has a logistic unit to represent the presence of an entity and a 4x4 matrix which could learn to represent the relationship between that entity and the viewer (the pose). A capsule in one layer votes for the pose matrix of many different capsules in the layer above by multiplying its own pose matrix by trainable viewpoint-invariant transformation matrices that could learn to represent part-whole relationships. Each of these votes is weighted by an assignment coefficient. These coefficients are iteratively updated for each image using the Expectation-Maximization algorithm such that the output of each capsule is routed to a capsule in the layer above that receives a cluster of similar votes. The transformation matrices are trained discriminatively by backpropagating through the unrolled iterations of EM between each pair of adjacent capsule layers. On the smallNORB benchmark, capsules reduce the number of test errors by 45% compared to the state-of-the-art. Capsules also show far more resistance to white box adversarial attacks than our baseline convolutional neural network.

1 INTRODUCTION

Convolutional neural nets are based on the simple fact that a vision system needs to use the same knowledge at all locations in the image. This is achieved by tying the weights of feature detectors so that features learned at one location are available at other locations. Convolutional capsules extend the sharing of knowledge across locations to include knowledge about the part-whole relationships that characterize a familiar shape. Viewpoint changes have complicated effects on pixel intensities but simple, linear effects on the pose matrix that represents the relationship between an object or object-part and the viewer. The aim of capsules is to make good use of this underlying linearity, both for dealing with viewpoint variations and for improving segmentation decisions.

Capsules use high-dimensional coincidence filtering: a familiar object can be detected by looking for agreement between votes for its pose matrix. These votes come from parts that have already been detected. A part produces a vote by multiplying its own pose matrix by a learned transformation matrix that represents the viewpoint invariant relationship between the part and the whole. As the viewpoint changes, the pose matrices of the parts and the whole will change in a coordinated way so that any agreement between votes from different parts will persist.

Finding tight clusters of high-dimensional votes that agree in a mist of irrelevant votes is one way of solving the problem of assigning parts to wholes. This is non-trivial because we cannot grid the high-dimensional pose space in the way the low-dimensional translation space is gridded to facilitate convolutions. To solve this challenge, we use a fast iterative process called "routing-by-agreement" that updates the probability with which a part is assigned to a whole based on the proximity of the vote coming from that part to the votes coming from other parts that are assigned to that whole. This is a powerful segmentation principle that allows knowledge of familiar shapes to derive segmentation, rather than just using low-level cues such as proximity or agreement in color or velocity. An important difference between capsules and standard neural nets is that the activation of a capsule is based on a comparison between multiple incoming pose predictions whereas in a standard neural net it is based on a comparison between a single incoming activity vector and a learned weight vector.

1

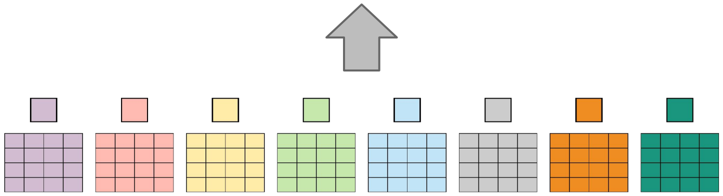
Google

Matrix capsules with Em routing

- Matrix Capsules

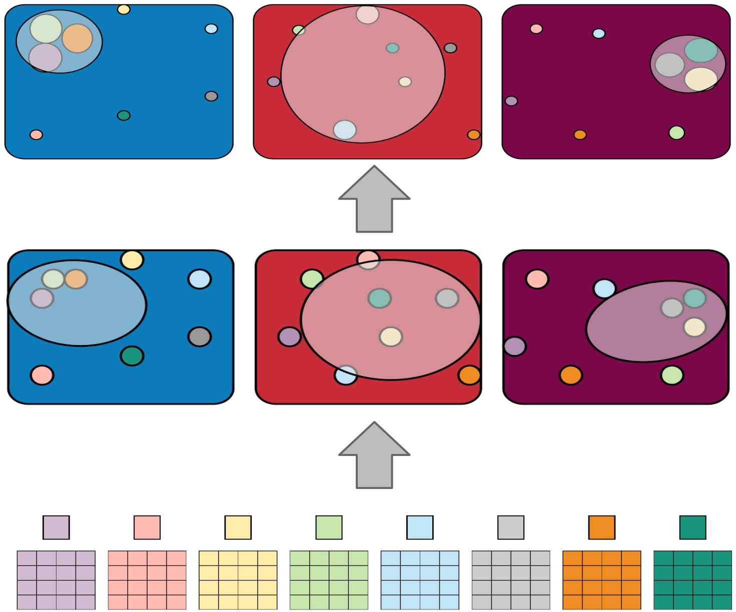
- Instantiation parameters is represented by pose matrix

- Separate activation variable



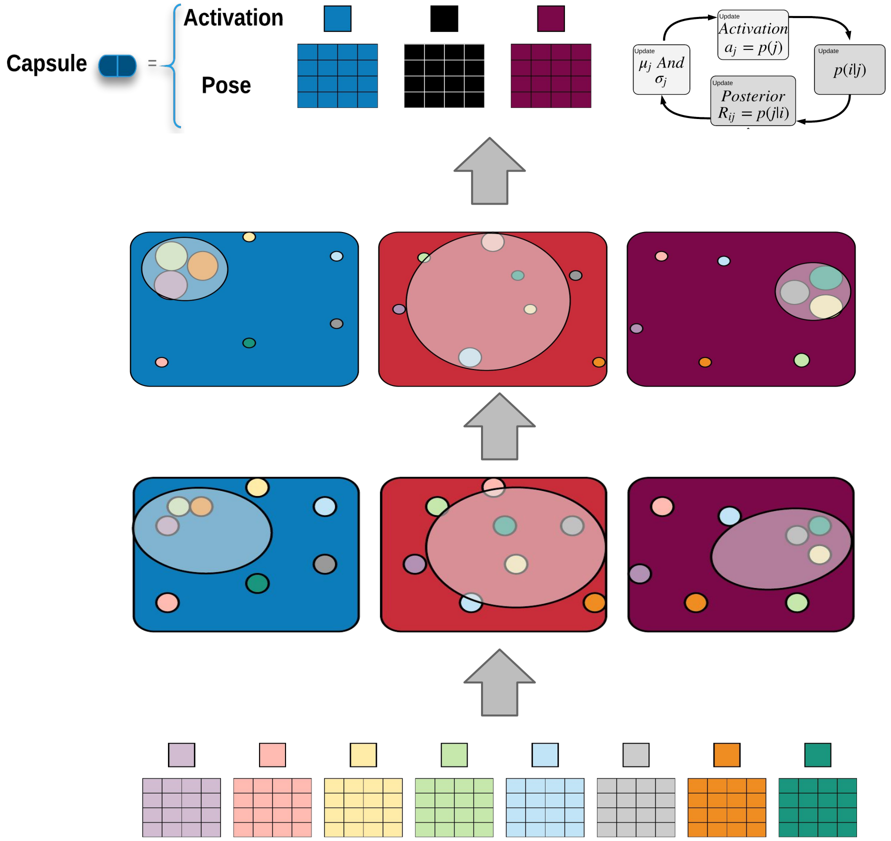
Em capsules with Matrix Transformations

- Matrix Capsules
- Instantiation parameters is represented by pose matrix
- Separate activation variable
- Matrix transformations between capsules
- Capsules output the center of the cluster as well as a confidence value

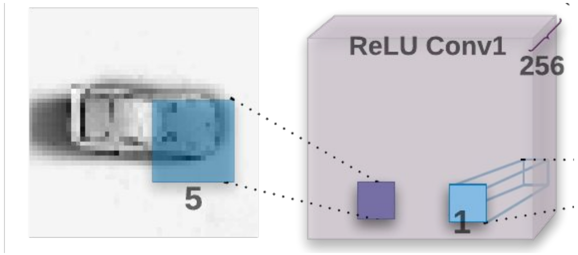


Em capsules with Matrix Transformations

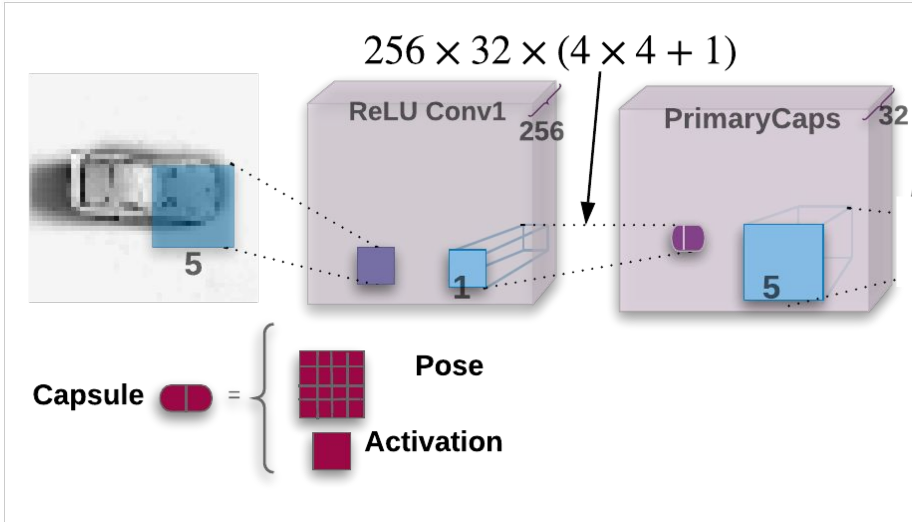
- Matrix Capsules
- Instantiation parameters is represented by pose matrix
- Separate activation variable
- Matrix transformations between capsules
- Capsules output the center of the cluster as well as a confidence value
- Clusters are found with iterative EM routing algorithm
- In this one the routing iterations provably converge



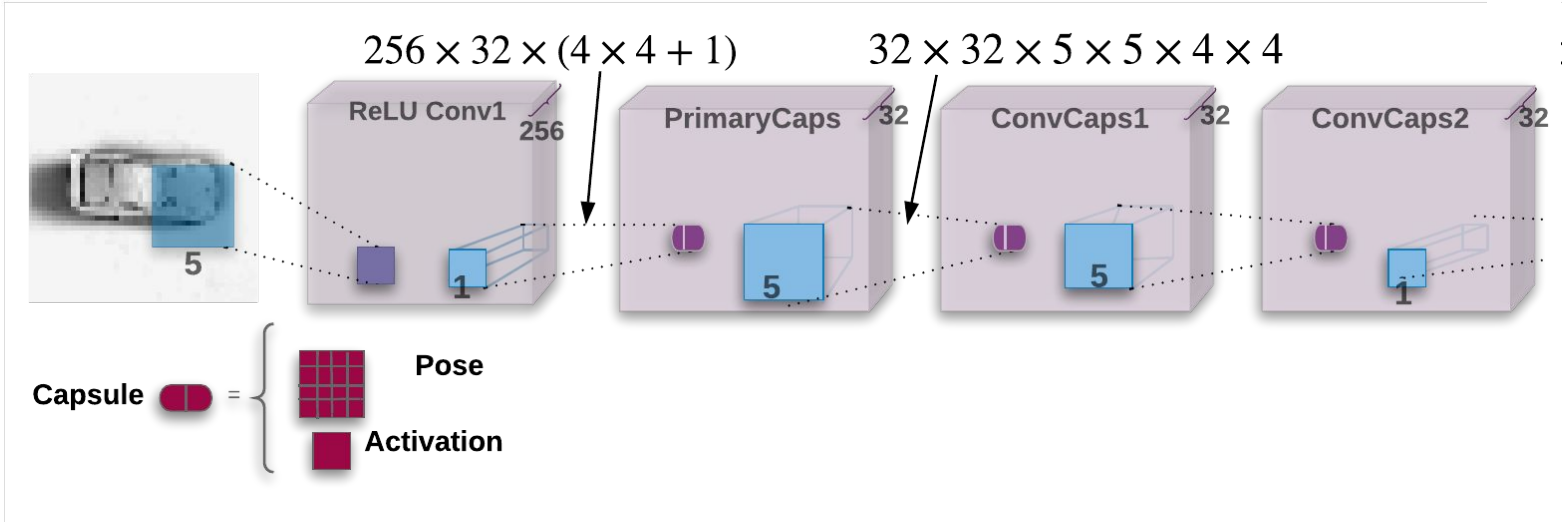
Capsule Architecture



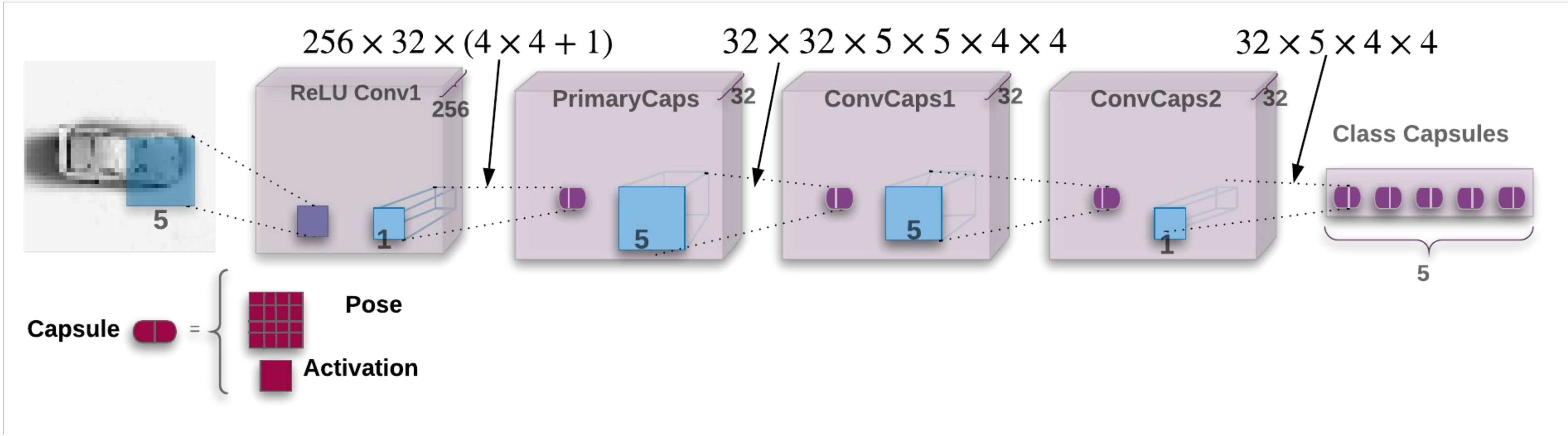
Capsule Architecture



Capsule Architecture



Capsule Architecture



Norb Classification Results

EM style version (100k trainable variables)

- smallNORB ~ 1.8% error (SOTA: 2.6%)
- fullNORB ~ 2.6% error (SOTA: 2.7%)

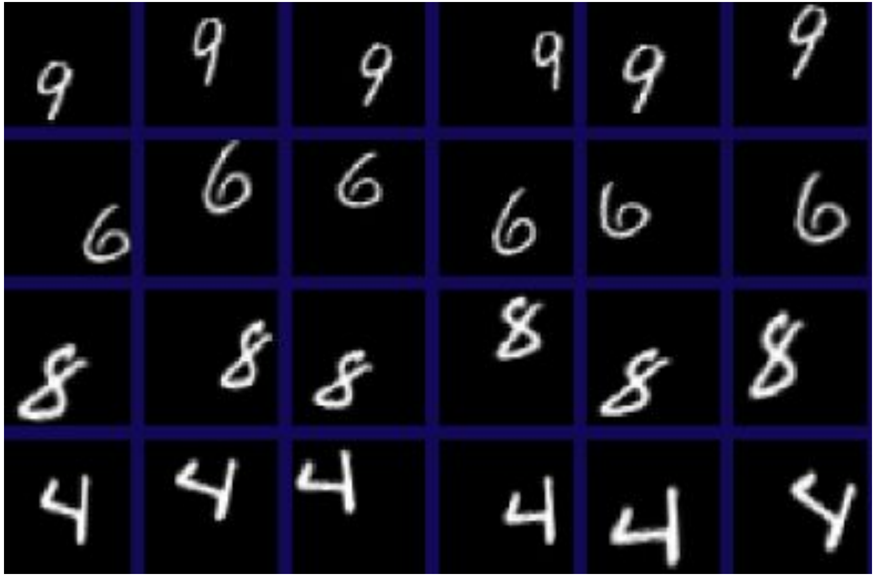
Norb transformation extrapolation

Test set	Azimuth		Elevation	
	CNN	Capsules	CNN	Capsules
Familiar viewpoints	3.7%	3.7%	4.3%	4.3%

Norb transformation extrapolation

Test set	Azimuth		Elevation	
	CNN	Capsules	CNN	Capsules
Novel viewpoints	20%	13.5%	17.8%	12.3%
Familiar viewpoints	3.7%	3.7%	4.3%	4.3%

Affnist Extrapolation

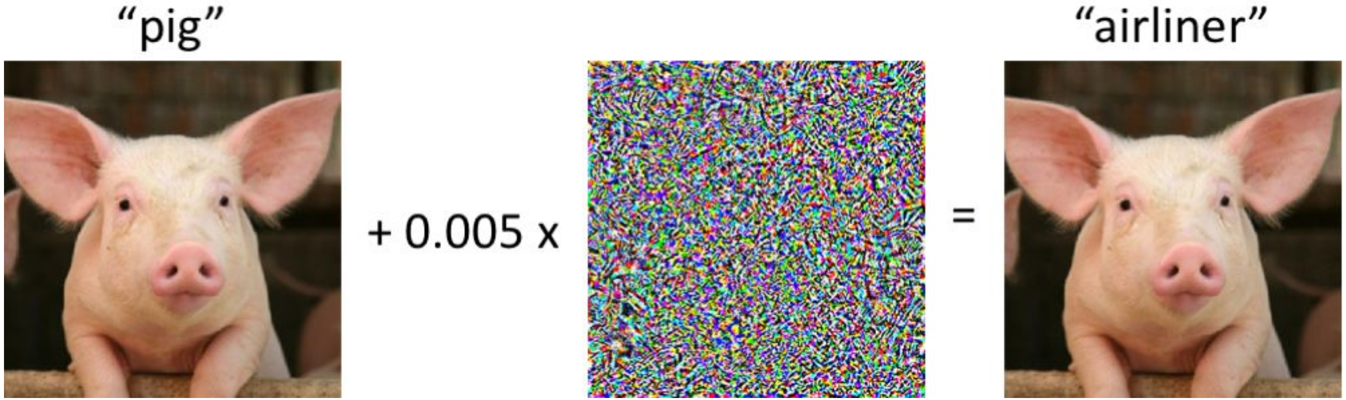


- Train on mnist, test on affnist
- CNN gets %66 test accuracy
- CapsNet get %79

Is there time left?

Adversarial Examples

- Calculus
- Attack Enginnering
- Failure



Source: http://people.csail.mit.edu/madry/lab/blog/adversarial/2018/07/06/adversarial_intro/

The Cycle of defense breaking

1. Propose a defense mechanism and claim to solve the problem

2. Propose a new attack that breaks the defense

3. Repeat

This has been mostly fruitless.
We don’t want models robust to adversarial attack, we want better models.

What exactly is an ‘Adversarial Examples’

- There is debate
  - ie I debate it

- Is it imperceptible changes?
  - Imperceptible to whom?

- Is it small changes?
  - How small?

- I posit that people are important to the definition
  - It’s really just intentionally crafted inputs we thought the network would get right that it doesn't't.

Are they a security risk?

Google

So should we even care about adversarial robustness?

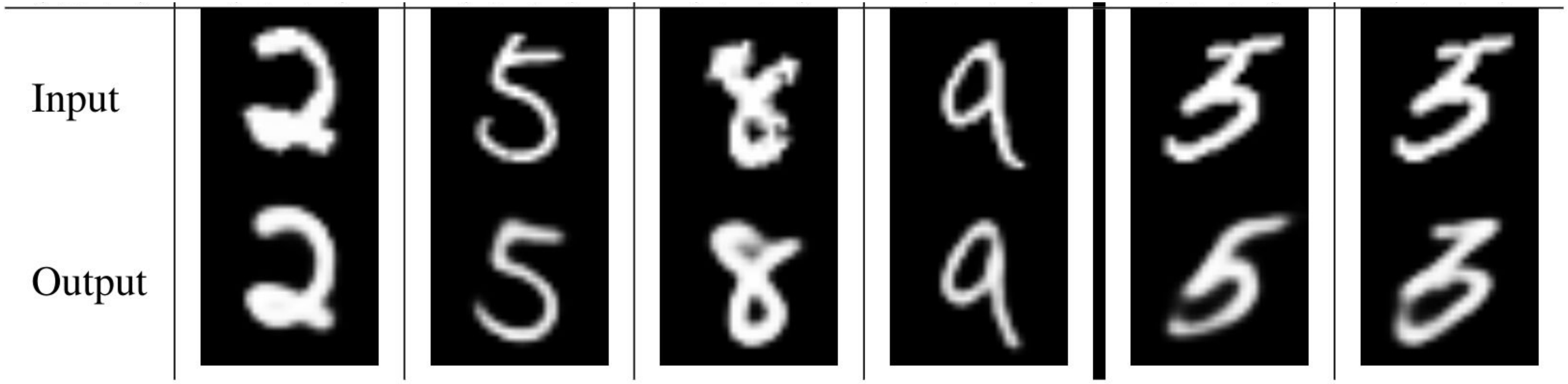
- Not really.

Are adversarial examples interesting and worth studying?

- yes!

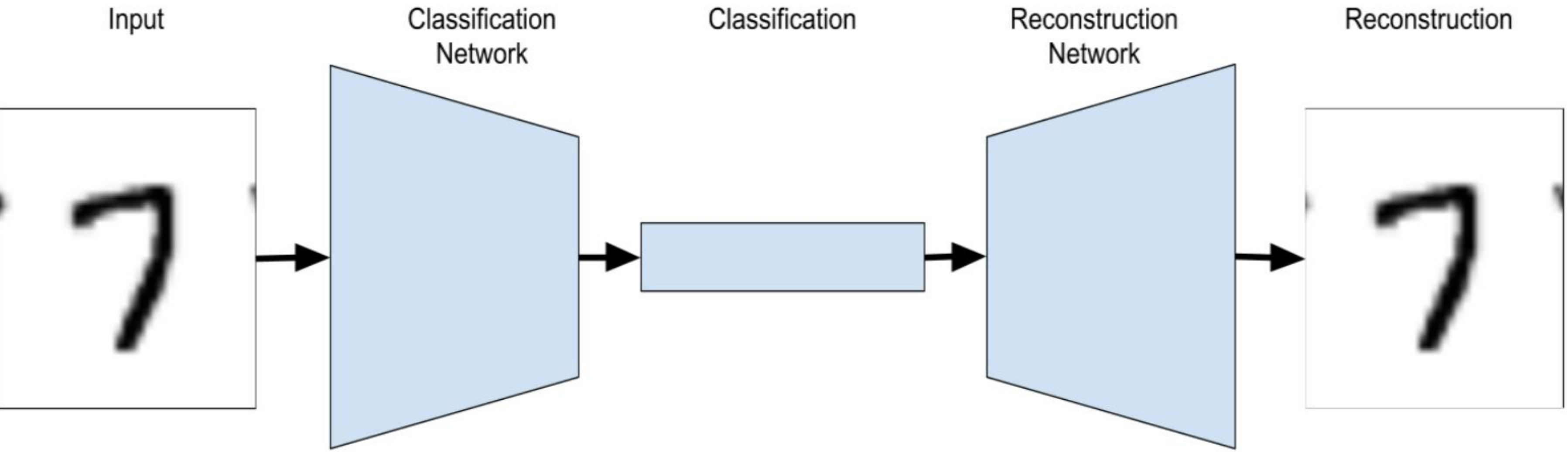
Capsule Reconstruction Network

- Take the class pose parameters
- And learn to reconstruct the input

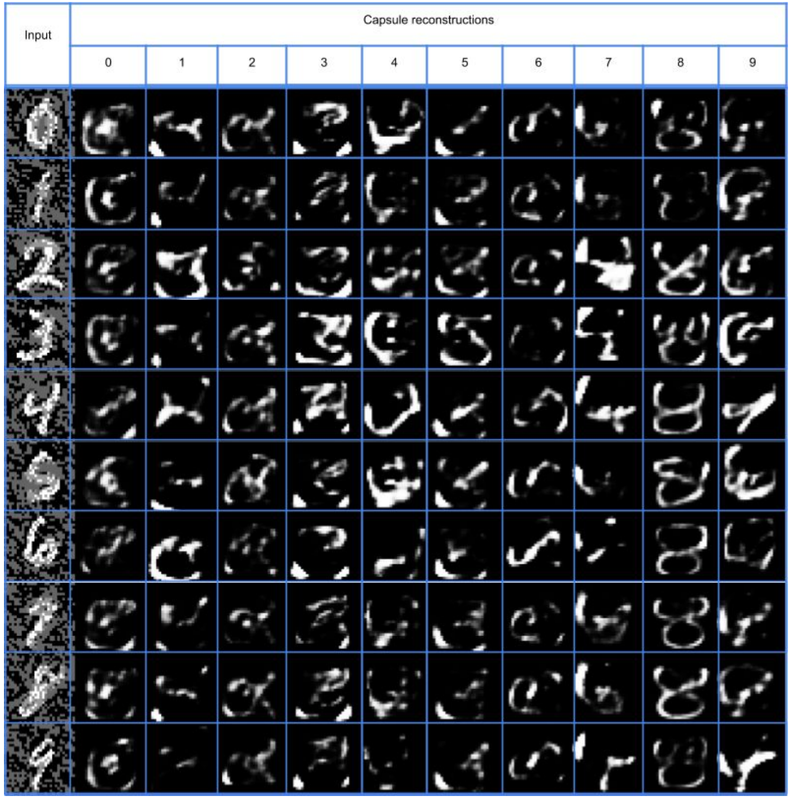


Capsule Networks now have two outputs

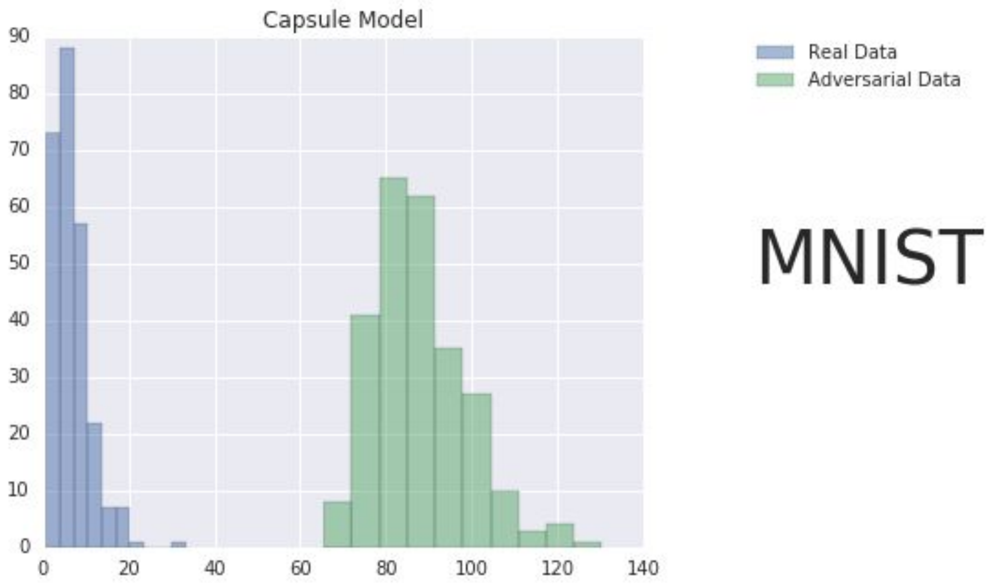
- A classification
- A reconstruction



Reconstruction form the wrong class

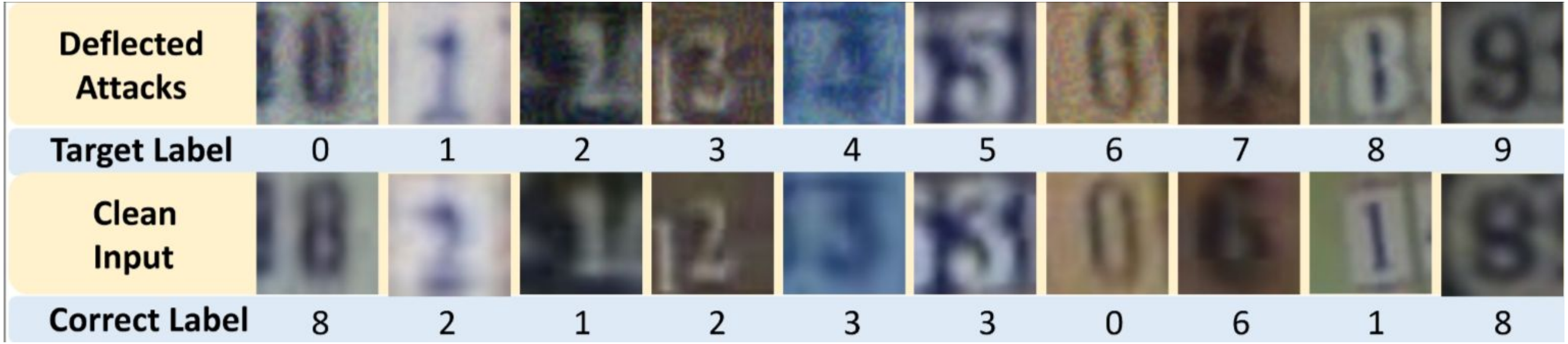


We can detect outlier data



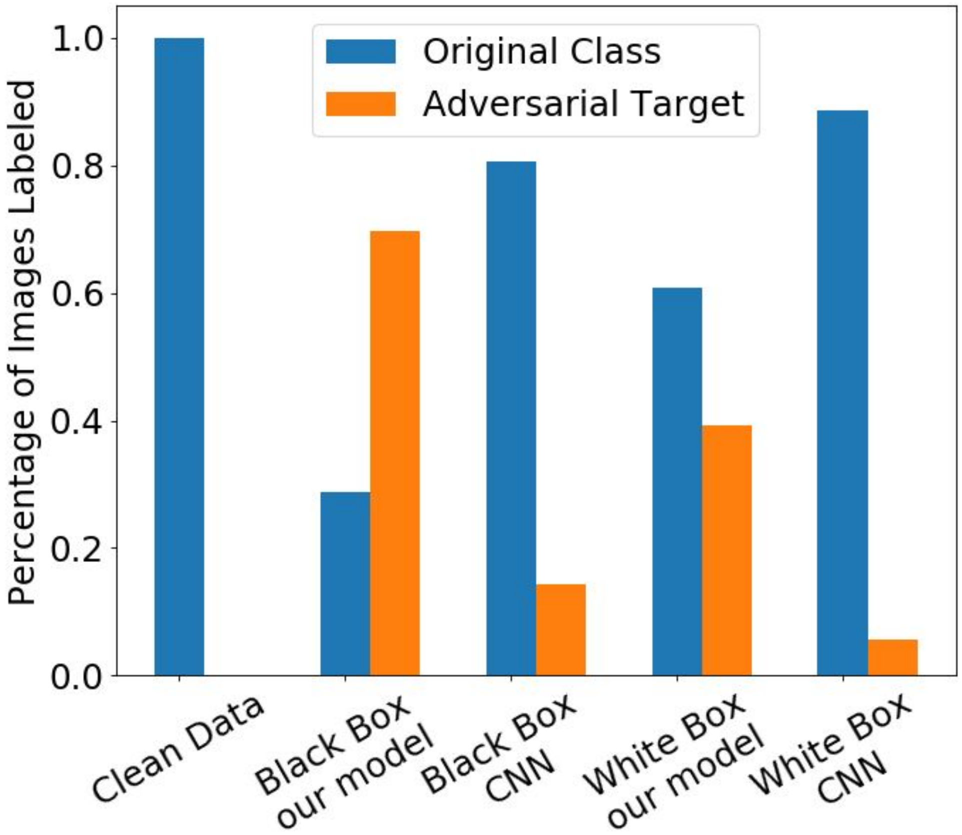
MNIST

Deflected attacks on SVHN



Deflected attacks on CIFAR10

Human Study



Google

Google

Brain Toronto

Thank you :)

email : frosst@google.com
twitter : @nickfrosst