# CapsuleNet

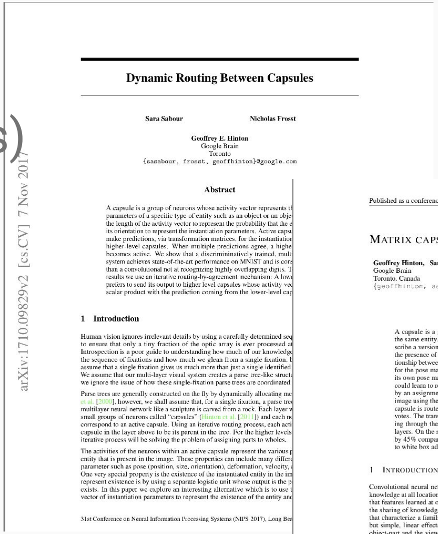
Capsules to rescue CNN: informative vectors instead of a single scalar output!

# Outline

- Introduction
- Motivation - Challenges of CNNs
- Capsules
- Dynamic Routing
- Squashing function
- CapsNet Architecture
- Results
  - Mnist
  - CIFAR10
  - Extra: smallNOPRB
- Conclusions

# CapsuleNet(2017)

- Geoffrey Hinton & Google et al.
- Dynamic routing between capsules (588 citations)
- Matrix capsules with EM routing (110 citations)
- MNIST, Cifar-10 classification and reconstruction




# Objective

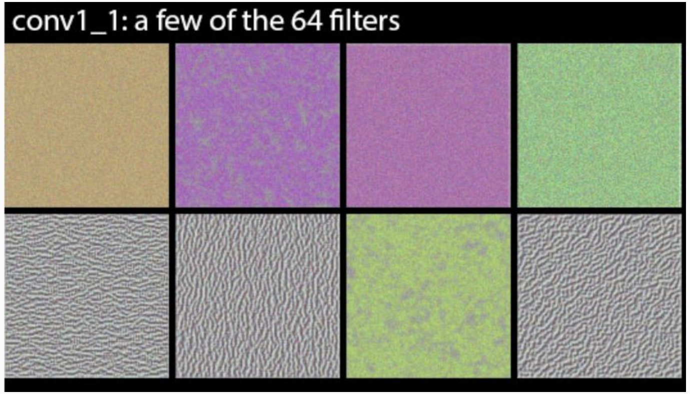
Capsule is a better representation of neurons than convolution.

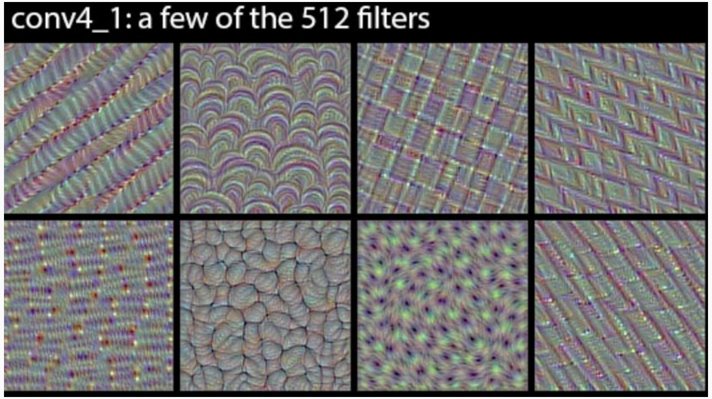
Because you achieve viewpoint invariance in the activities of neurons.

In English,
when you see a car, you should be able to tell that it is a car from an arbitrary viewpoint.

# Motivation – Challenges of CNNs

- Kernels filter features from input

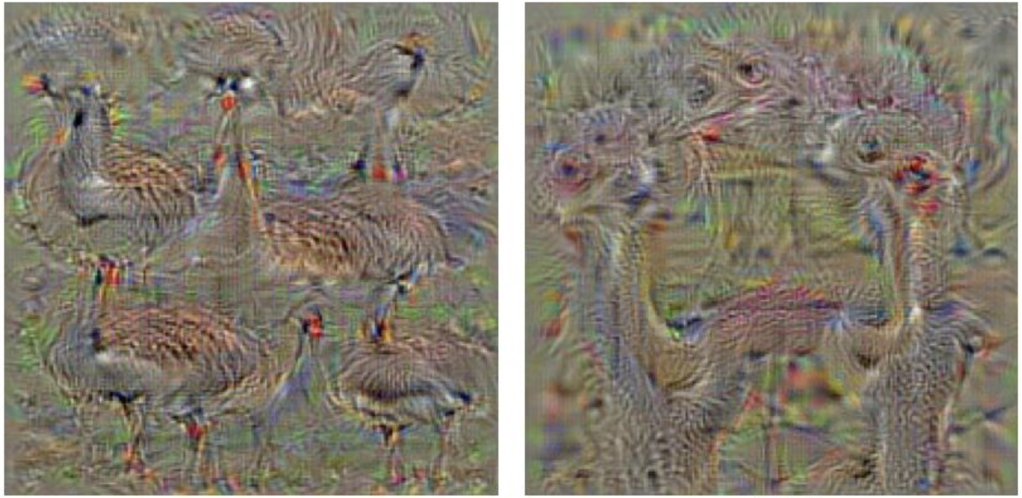



- Lower layers learn basic features, such as edges, cornes

- Higher layers learn complex features

# Motivation – Challenges of CNNs



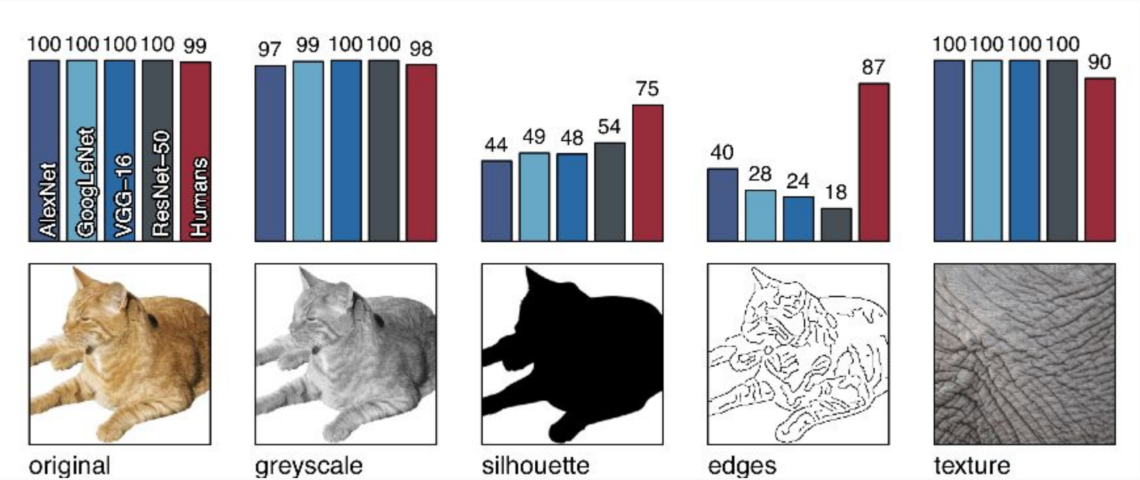
goose ostrich

Figure 1: **Numerically computed images, illustrating the class appearance models, learnt by a ConvNet, trained on ILSVRC-2013.** Note how different aspects of class appearance are captured in a single image. Better viewed in colour.

- Input that maximizes a specific class
- Does not look like a real image at all

Simonyan, et al

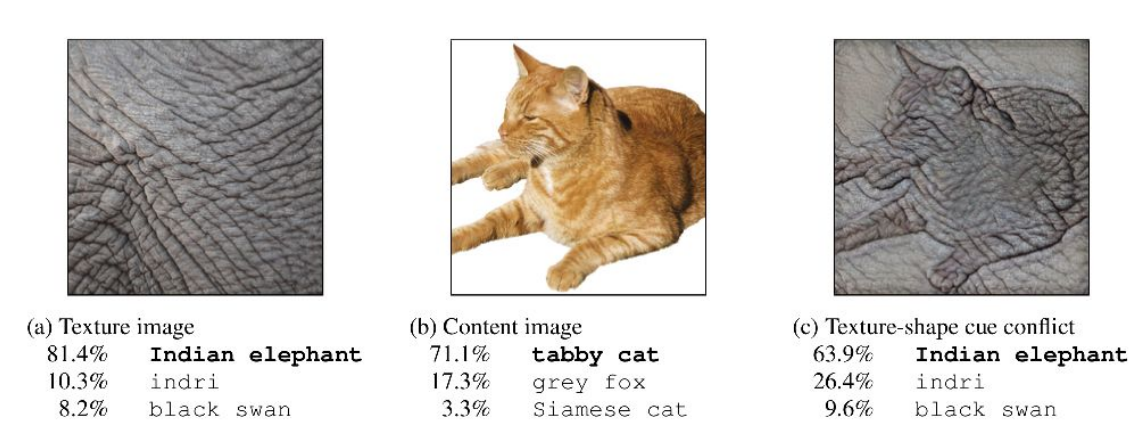
# Motivation – Challenges of CNNs

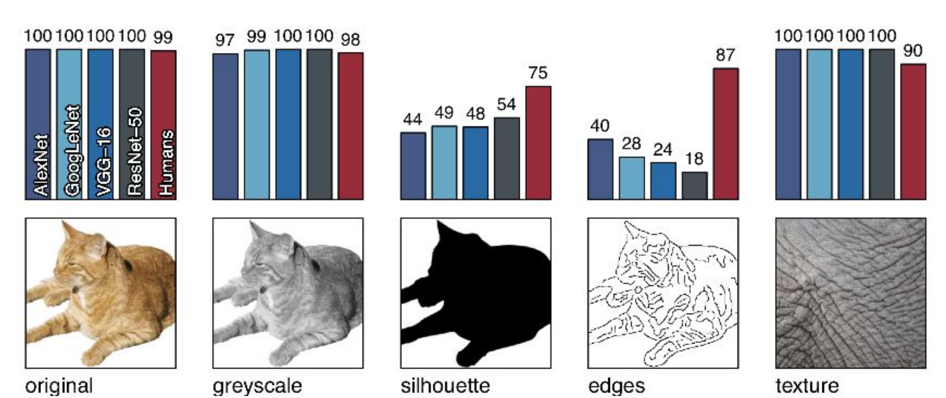



Data augmentation can help:

- Flip
- Rotation
- Translation
- Crop
- Added noise
- Contrast
- Brightness
- Shear angle
- Style transfer
- ...

# Motivation – Challenges of CNNs



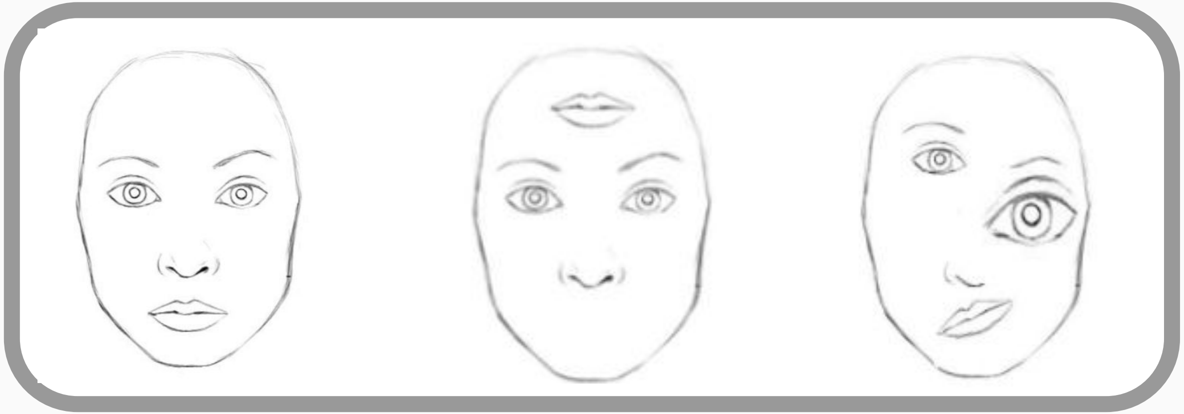




- CNNs rely on texture too much

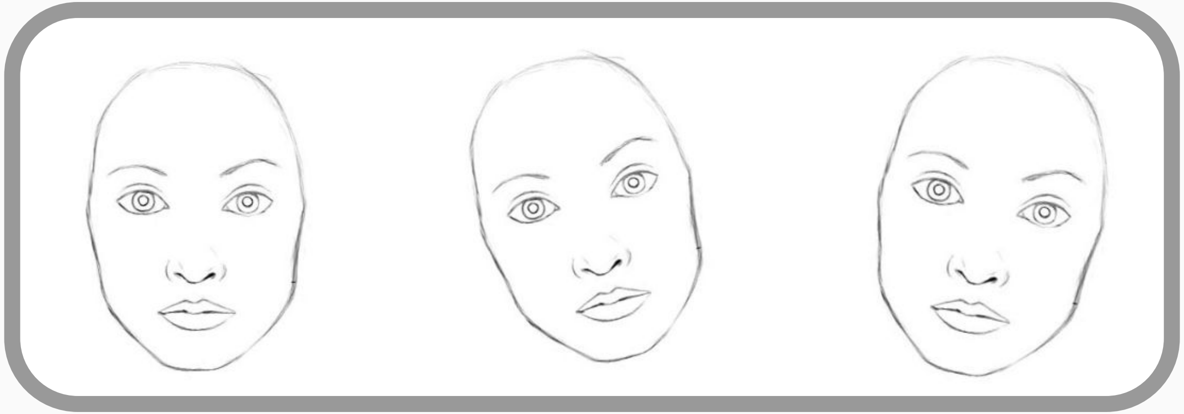
Data augmentation can help:

- Flip
- Rotation
- Translation
- Crop
- Added noise
- Contrast
- Brightness
- Shear angle
- Style transfer
- ...

# Motivation – Challenges of CNNs



- CNN are easily fooled. All of these are recognized as faces



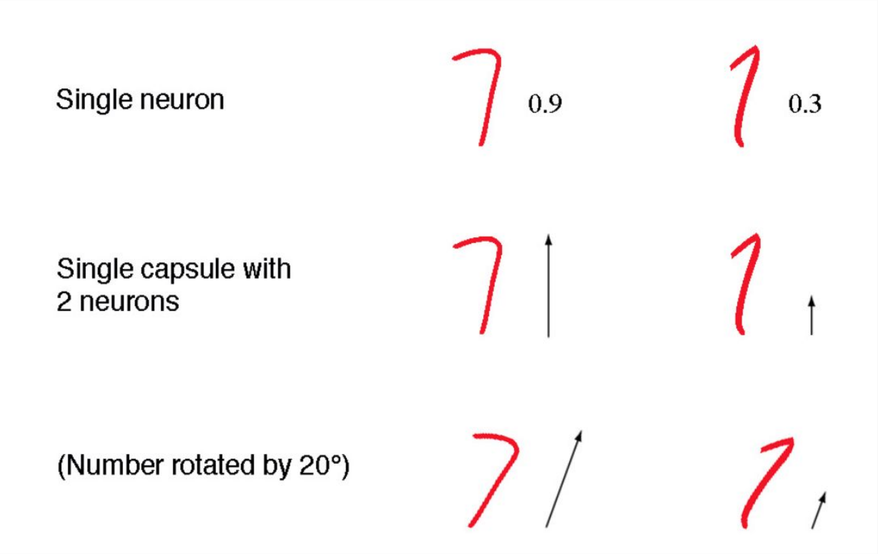
- CNN cannot easily extrapolate. This requires augmentation.

# Motivation – Challenges of CNNs

- Kernels output scalars
  - Little orientational and relative spatial relationships between features
- Max pooling loses valuable information
  - Weak spatial hierarchies between simple and complex object

# Capsules – emulate neurons better with a capsule!

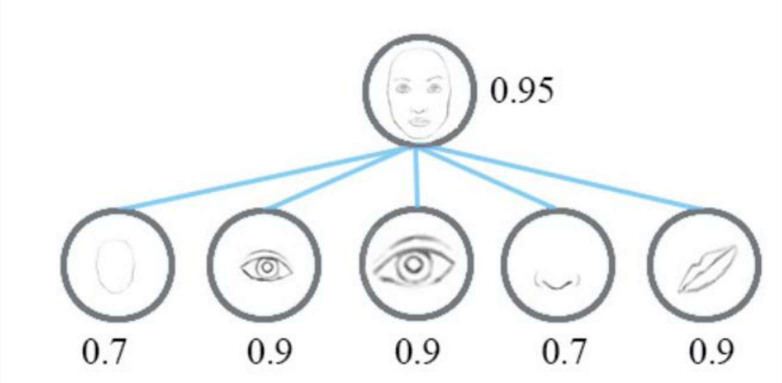
- Activation outputs a vector instead of a scalar
  - Length: probability that the entity is present within its limited domain
  - Direction: “instantiation parameters” of the input (e.g. pose, lighting and etc.)
  - Even if the direction (pose) changes, the length (probability) may stay the same.
    - Activity Equivariance



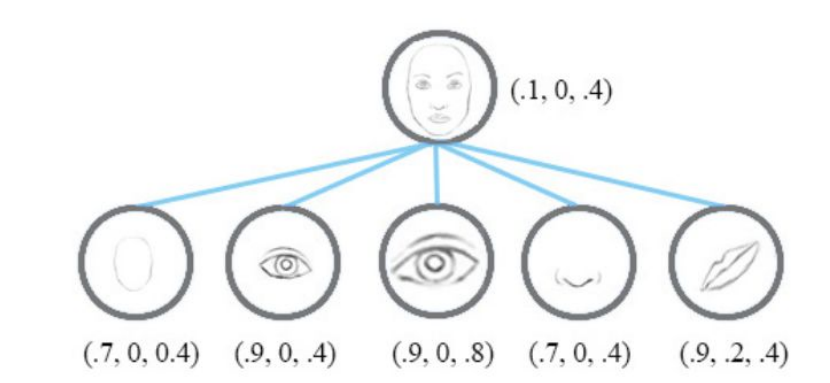
# Capsules – how does our brains work?

- We decompose hierarchical representations and do pattern matching.
- The representation is view-angle invariant.

- Takes a vector as input and outputs a vector
- Output vector encodes information about feature transformations




Traditional Convolutional Layer
(scalar output)



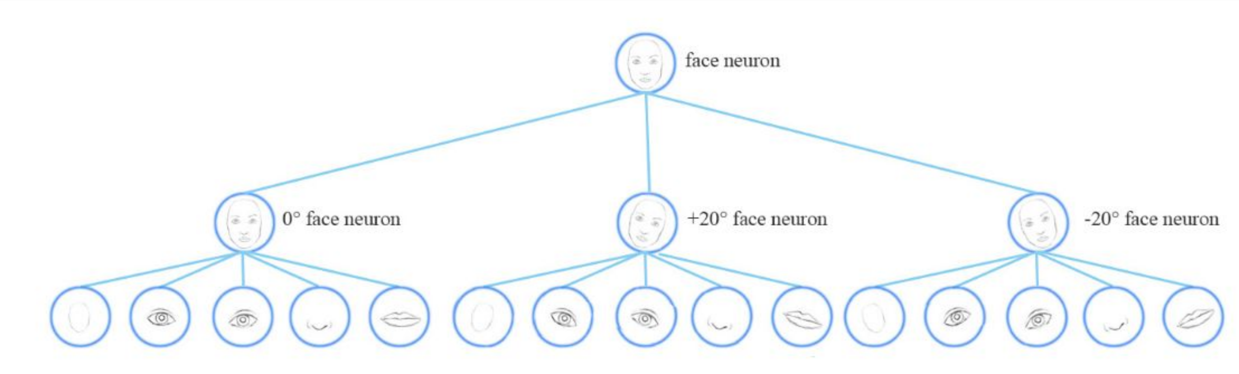

Capsule Layer
(vector output)

# Capsule in a nutshell

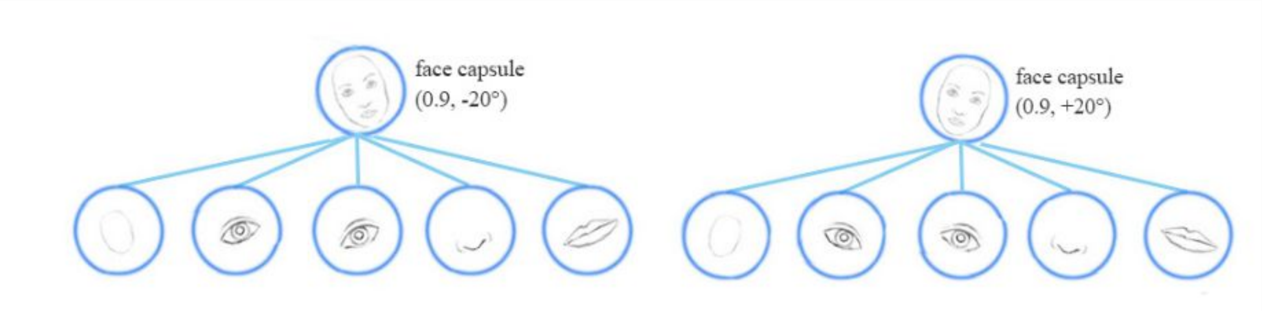
Takes a vector as input and outputs a vector

# Capsules – how does our brains work?

- We decompose hierarchical representations and do pattern matching.
- The representation is view-angle invariant.




Traditional Convolutional Layer
(scalar output)



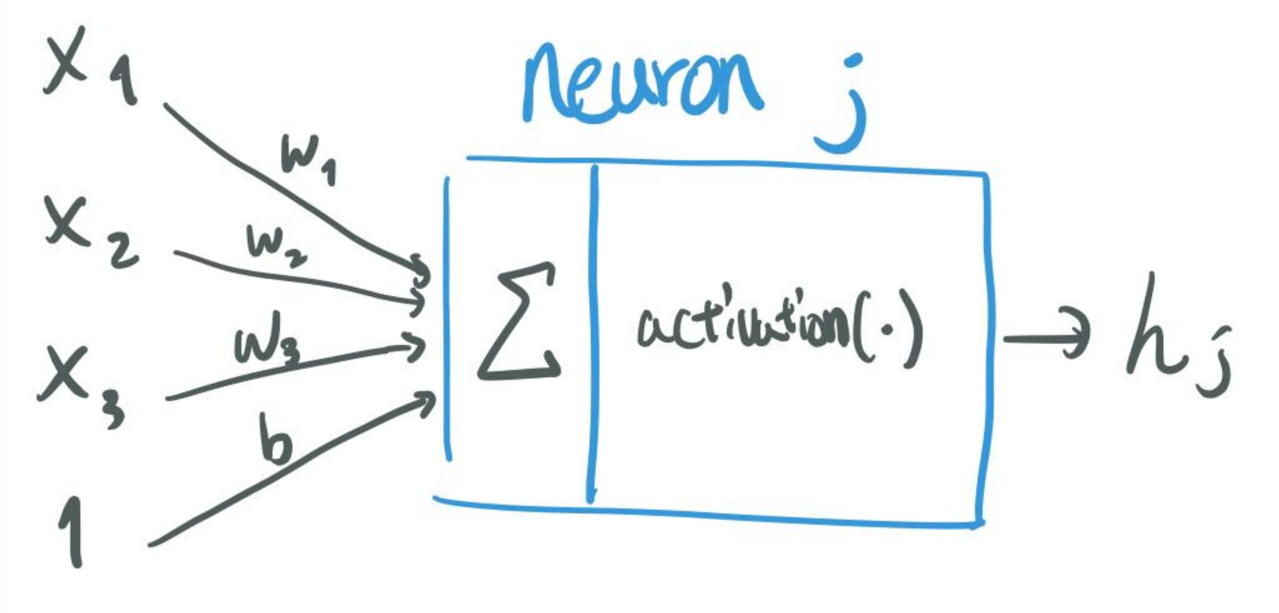

Capsule Layer
(vector output)

Less parameters

# Intuition of Capsule

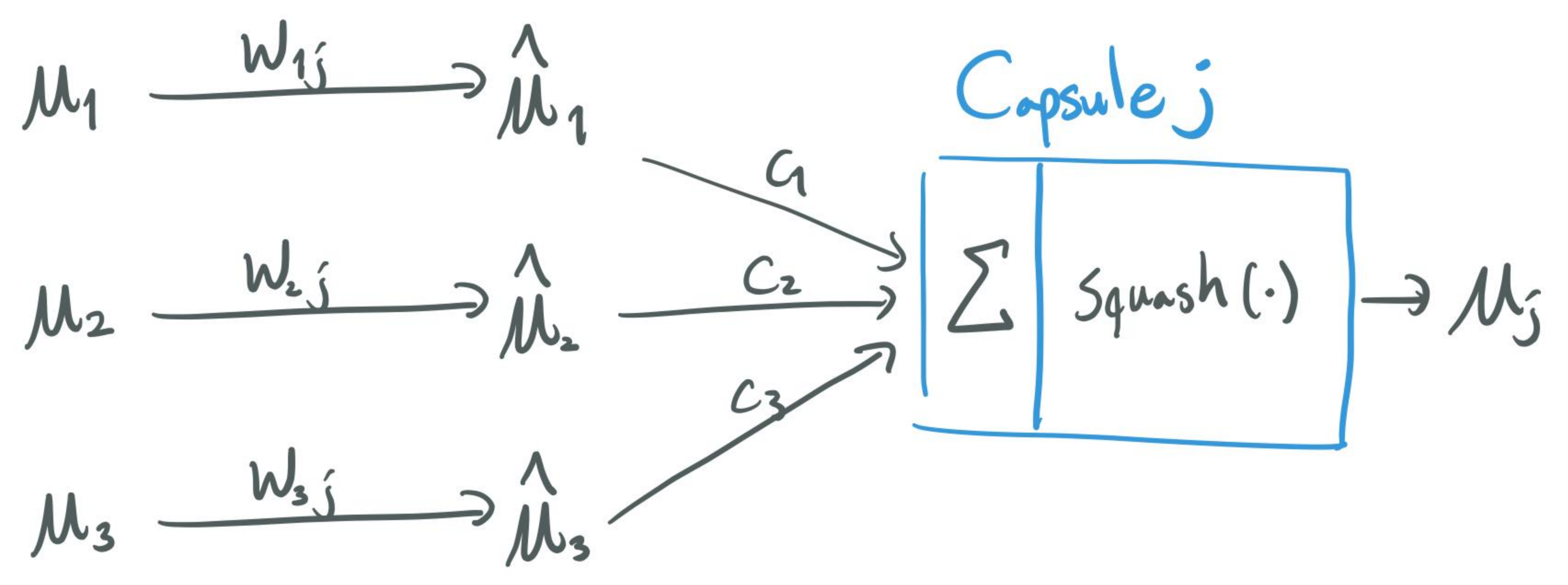
- 

# How does a capsule work? – Traditional Neuron



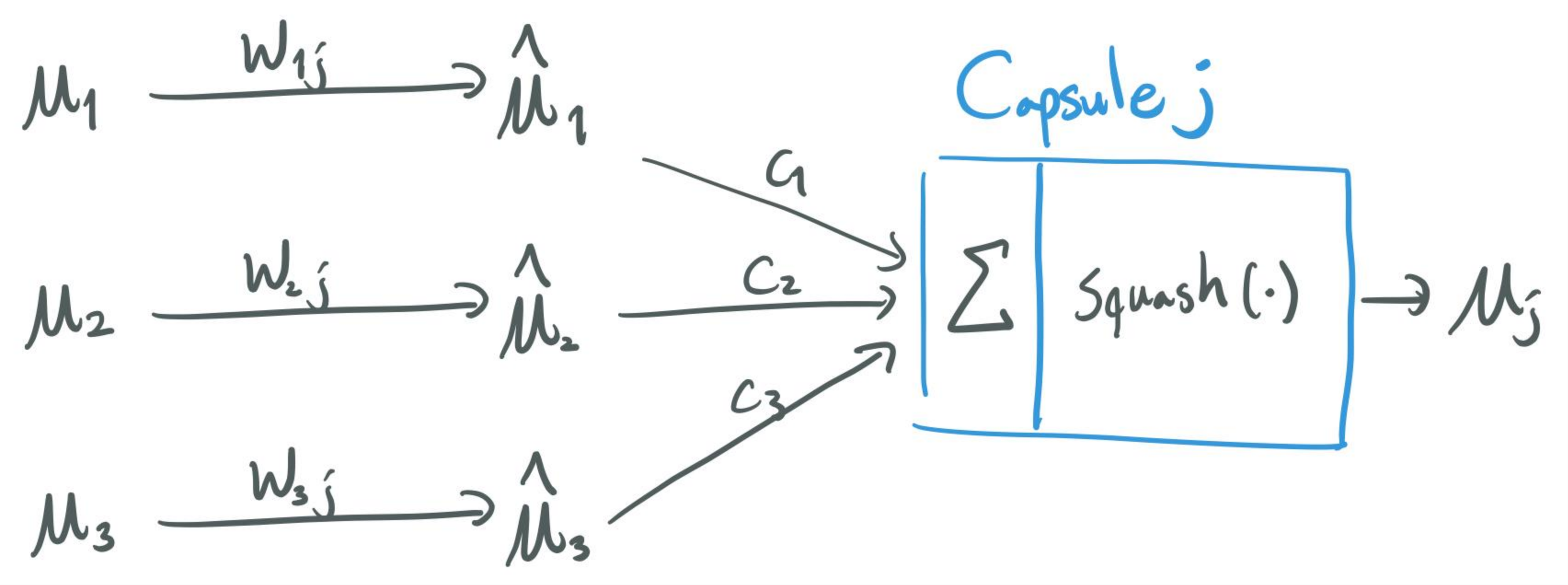

X_n: a scalar from previous convolution layer. Represents feature activation level

# How does a capsule work? – Capsule




- Weight matrices encode spatial and other relationships between lower level features and higher level features.
- Output vector is a predicted position of the higher level feature given the lower level feature.

# How does a capsule work?




- Non-negative scalar weight c_n is determined using “dynamic routing”.
- Sum([c_1, …, c_n]) = 1
- Len([c_1, …, c_n]) = #Number of the next level capsules

# How does a capsule work? – Capsule vs. Traditional Neuron

| Capsule vs. Traditional Neuron | | | |
|---|---|---|---|
| Input from low-level capsule/neuron | | vector($\mathbf{u}_i$) | scalar($x_i$) |
| Operation | Affine Transform | $\hat{\mathbf{u}}_{j\|i} = \mathbf{W}_{ij}\mathbf{u}_i$ | – |
| | Weighting | $\mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j\|i}$ | $a_j = \sum_i w_i x_i + b$ |
| | Sum | | |
| | Nonlinear Activation | $\mathbf{v}_j = \frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}$ | $h_j = f(a_j)$ |
| Output | | vector($\mathbf{v}_j$) | scalar($h_j$) |

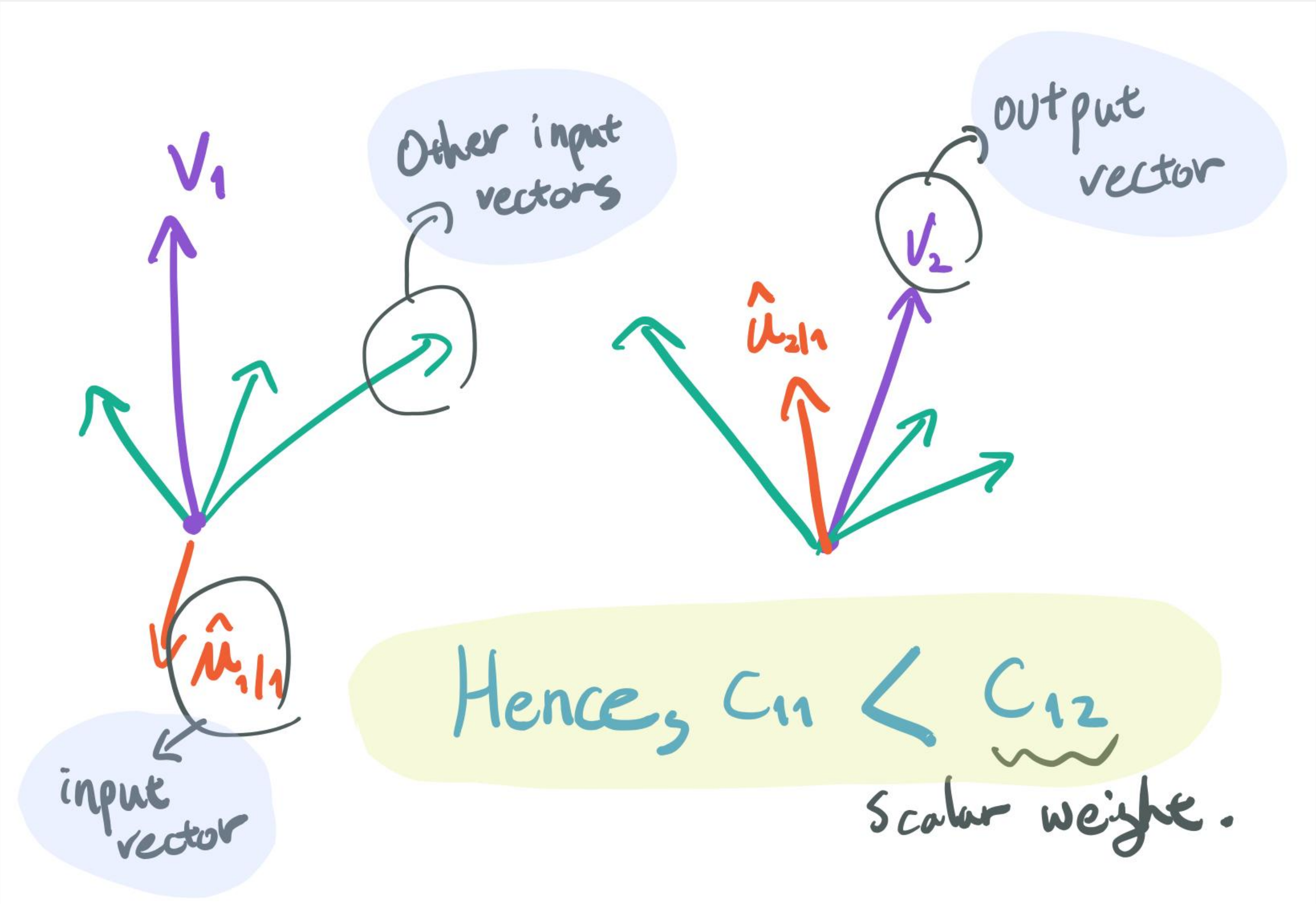
# How does a capsule work? – Dynamic Routing

**Procedure 1** Routing algorithm.

```
procedure ROUTING(û_{j|i}, r, l)
    for all capsule i in layer l and capsule j in layer (l + 1): b_ij ← 0.
    for r iterations do
        for all capsule i in layer l: c_i ← softmax(b_i)                  ▷ softmax computes Eq. 3
        for all capsule j in layer (l + 1): s_j ← Σ_i c_ij û_{j|i}
        for all capsule j in layer (l + 1): v_j ← squash(s_j)             ▷ squash computes Eq. 1
        for all capsule i in layer l and capsule j in layer (l + 1): b_ij ← b_ij + û_{j|i}.v_j
    return v_j
```

- u_hat: output of previous level capsule
- r: routing iteration, (3 is recommended)
- l: previous level
- v_j: output of next level capsule
- b_ij: temporary coefficient holder. At the end, it's stored to c_ij

# How does a capsule work? – Dynamic Routing



# How does a capsule work? – Squashing as nonlinearity

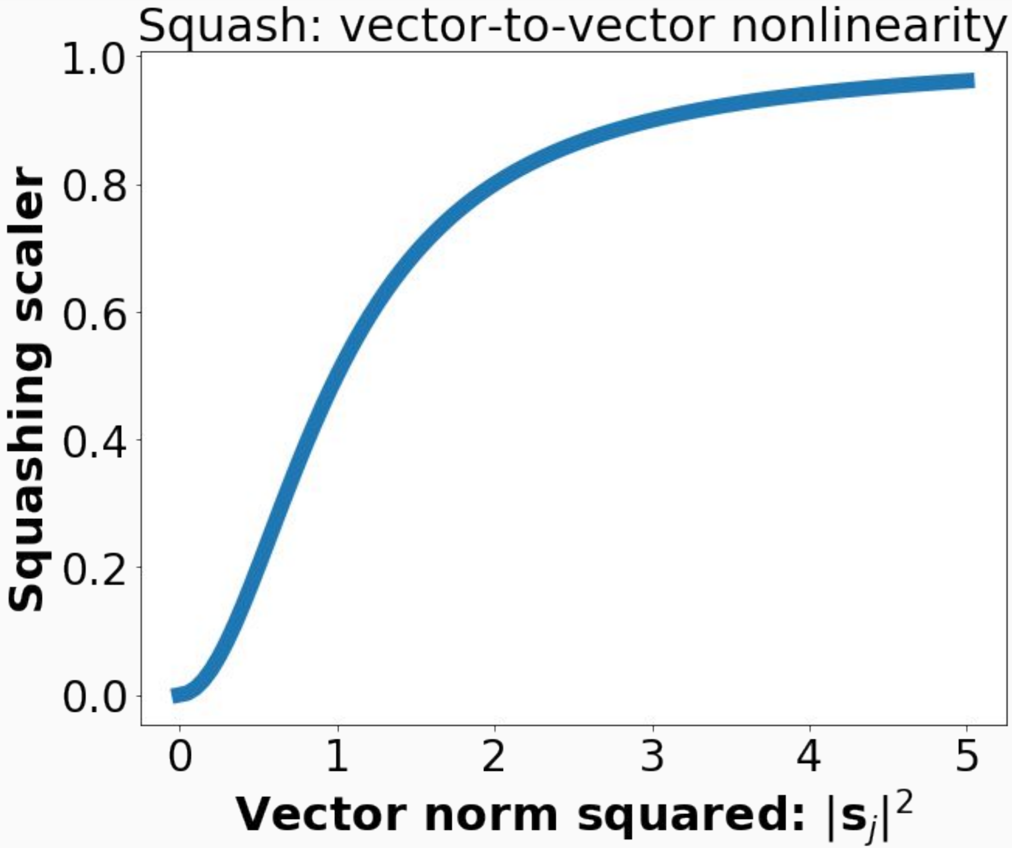
$$\mathbf{v}_j = \underbrace{\boxed{\frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}}}_{\text{additional “squashing”}}\ \underbrace{\boxed{\frac{\mathbf{s}_j}{\|\mathbf{s}_j\|}}}_{\text{unit scaling}}$$

- Length of short vectors => ~0
- Length of long vectors => ~1

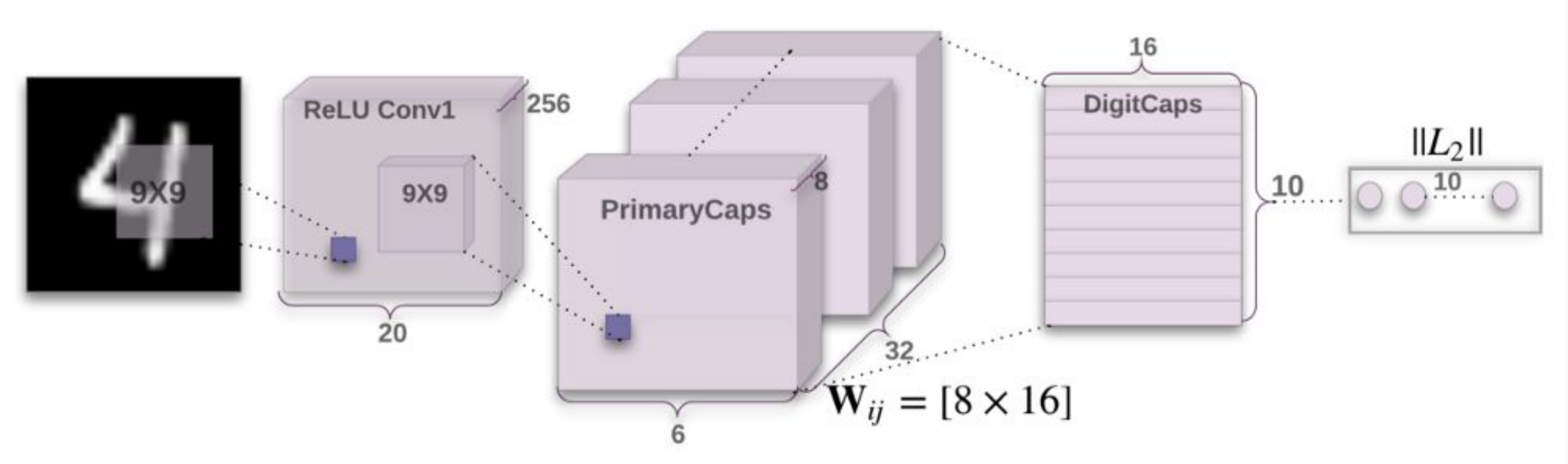
# How does a capsule work? – Squashing as nonlinearity

$$\boxed{\frac{\|\mathbf{s}_j\|^2}{1+\|\mathbf{s}_j\|^2}}$$

additional "squashing"

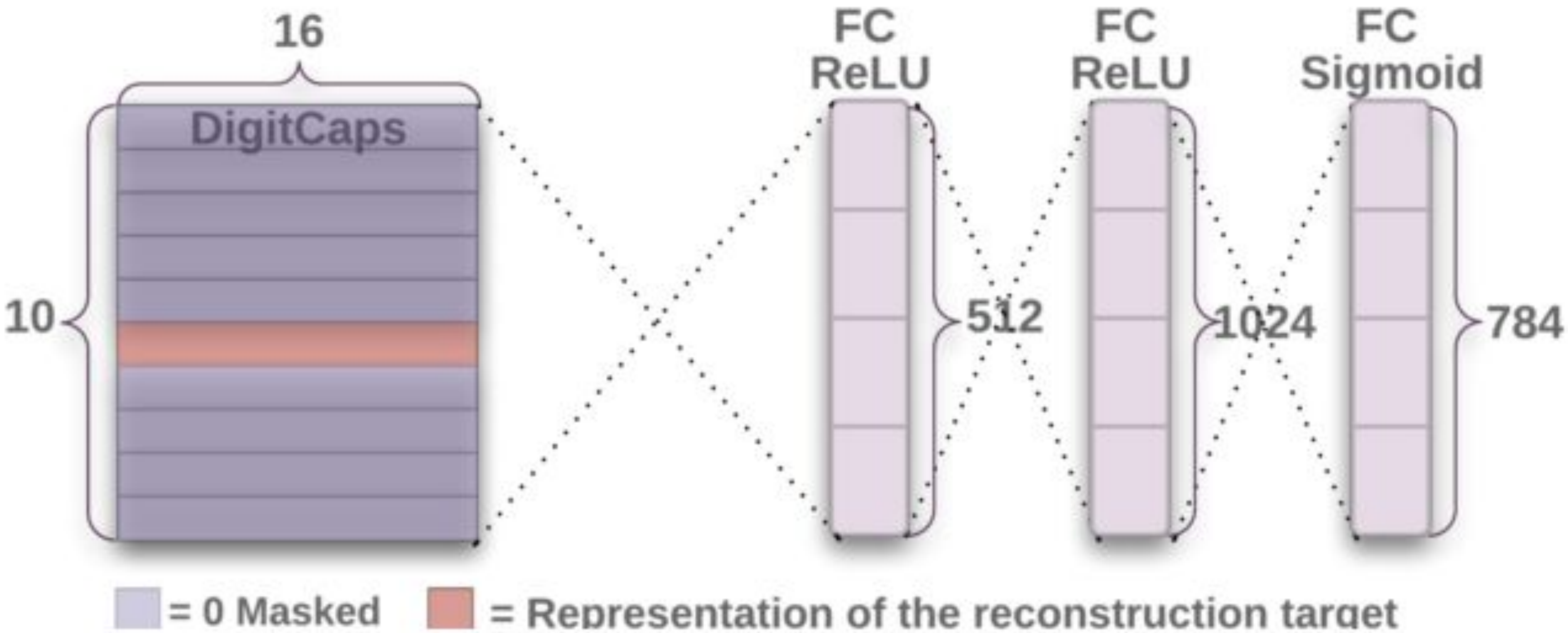


# CapsNet architecture – Encoder (classifier)




- 2D Convolutional  layer: Convert pixel intensities to the activities of local feature detectors
- PrimaryCaps layer (convolutional): Invert rendering process
- DigitCaps layer (fully connected):

# CapsNet architecture – Decoder (reconstruction)



# CapsNet architecture – Margin Loss

$$L_k = T_k \max(0, m^+ - ||\mathbf{v}_k||)^2 + \lambda \, (1 - T_k) \max(0, ||\mathbf{v}_k|| - m^-)^2$$

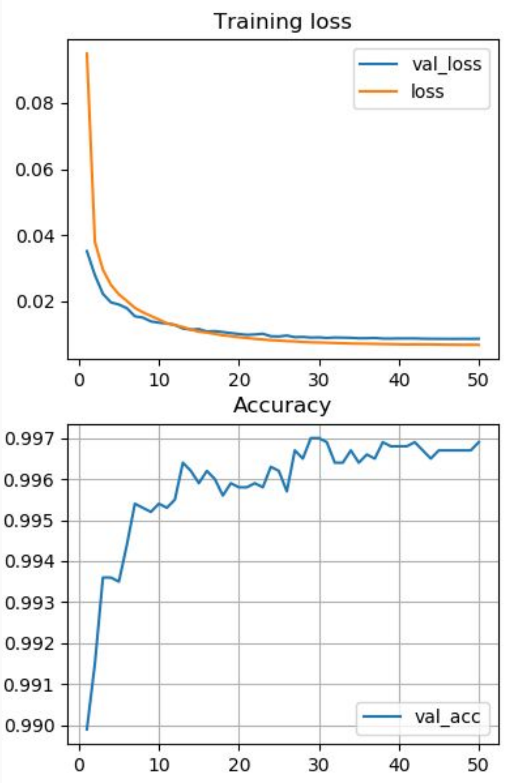
- Calculate loss for each capsule at the top-level digit capsule,
  - i.e. for each class
- T_k = 1 iff a class exists in an image
- m+: 0.9
- m-: 0.1
- Lambda: down-weighting for initial learning iterations
- Total loss: Sum([L_1, …, L_k])

# CapsNet architecture – Total Loss

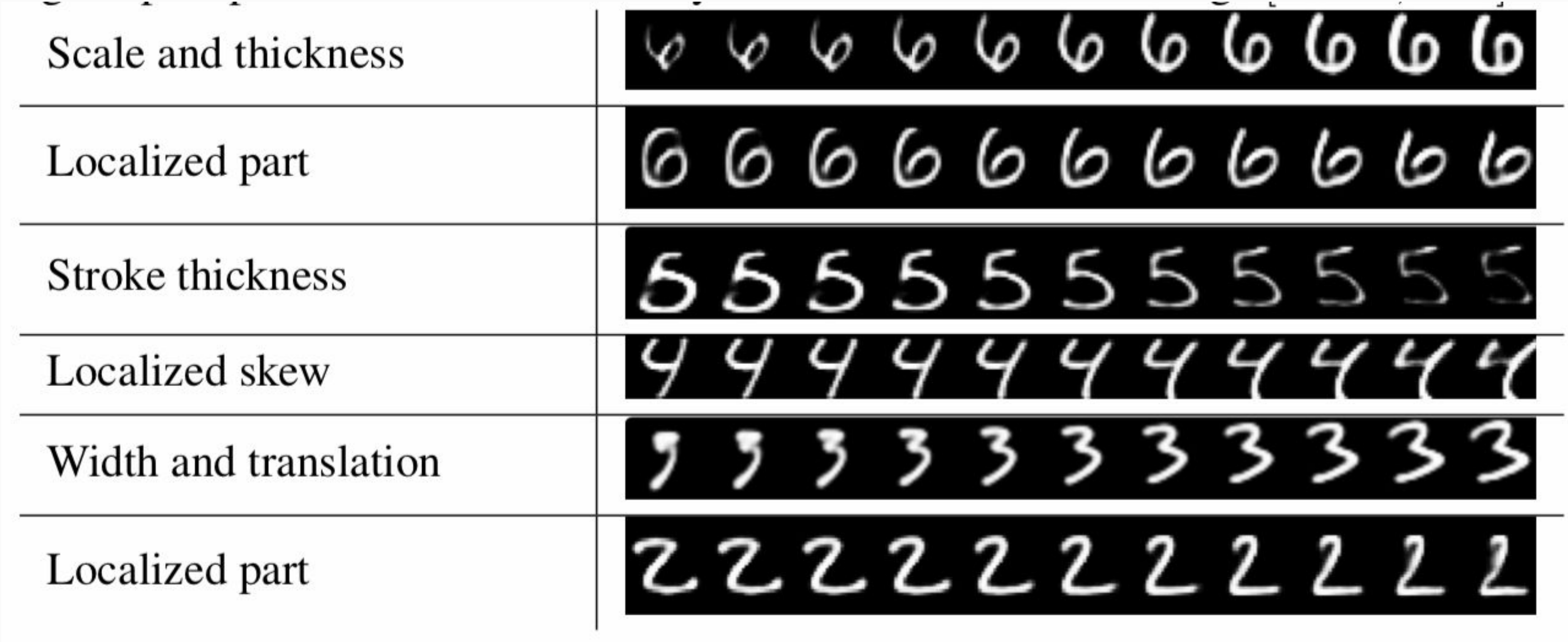
Loss = Loss_margin + 0.0005 * MSE(reconstructed_image, input_image)

# Experiment – MNIST

- accuracy: 99.7%
- loss: 0.00855



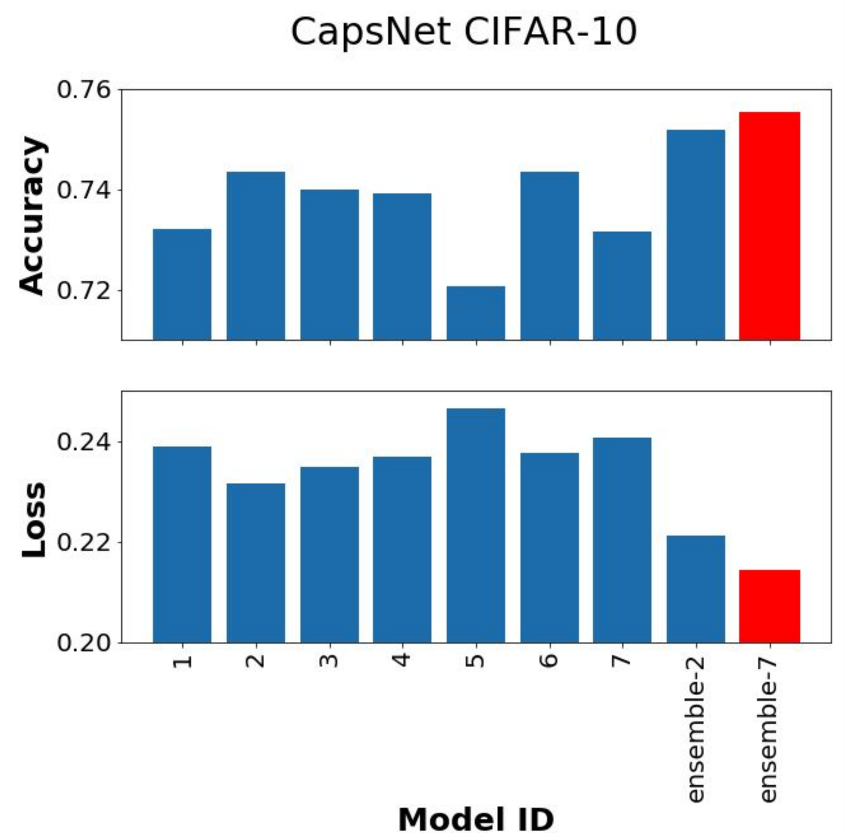

# CapsNet architecture – Interpretable activation vectors



# CapsNet architecture – Robustness to affine transformation

# Experiment – CIFAR-10

- 32x32x3 image classification
- 10 classes, SOTA: 99%, Paper: 89.4%

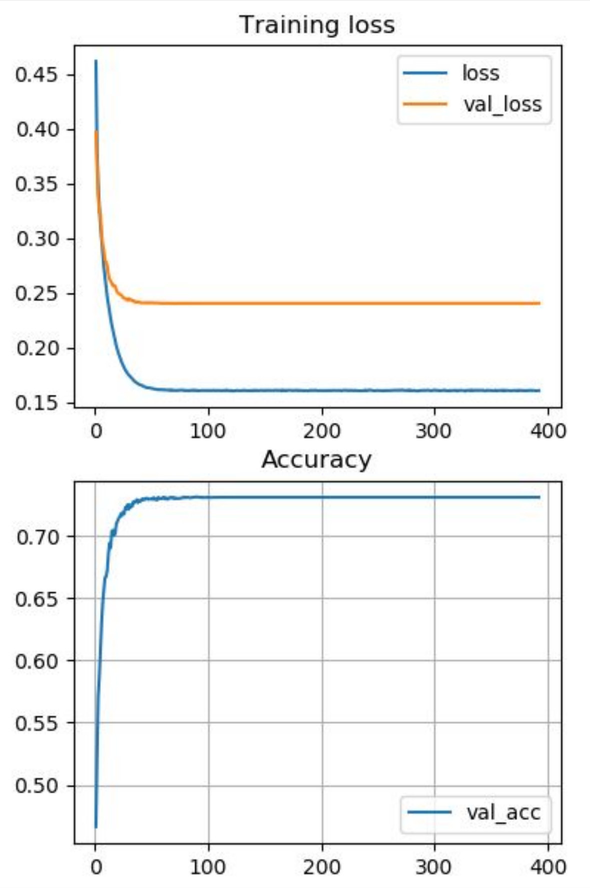



# Experiment – CIFAR-10

1st Epoch

1000th Epoch

# Experiment – CIFAR-10



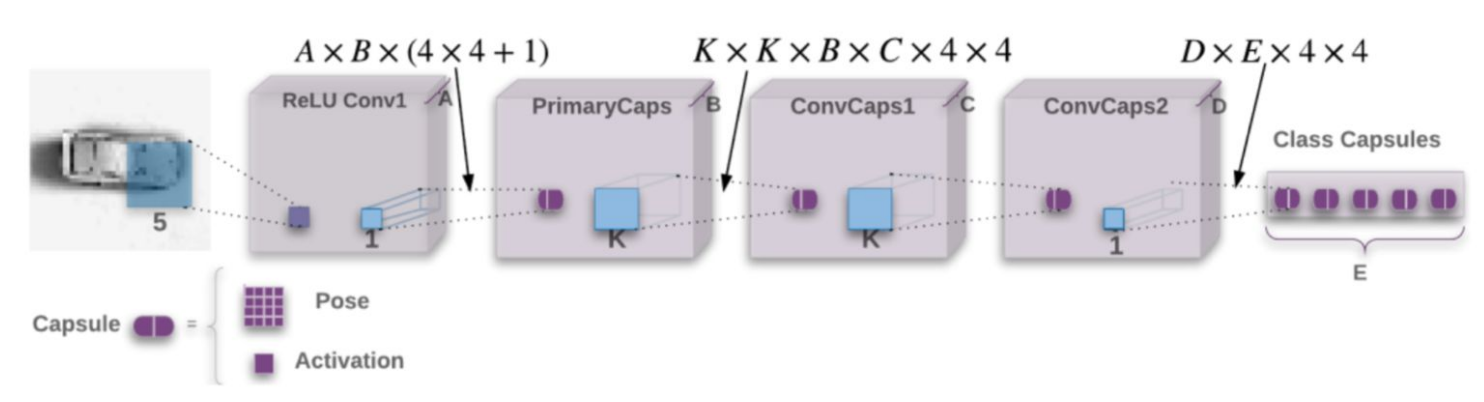

1000th Epoch

# Extra - smallNORB (Dynamic Routing with EM)

- *smallNORB* dataset (48 600 images)
  - 96x96 images
  - 5 classes
  - 10 instances per class
  - 18 azimuths per instance
  - 9 elevations per instance
  - 6 lightning conditions

# Extra - Dynamic Routing with EM

- EM algorithm instead of dynamic routing




| 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% |

# Conclusion

- Capsules are convolutions with block non-linearity and routing
- Capsules require less parameters than conv (6.8M vs. 35.4M)
  - However, the routing procedure involves slow iterations
- Capsules try to build better model hierarchical relationships inside of internal knowledge representation of an NN.
- Nonetheless, capsule networks are not very popular yet.

# References


- https://pechyonkin.me/capsules-4/
- https://www.kaggle.com/fizzbuzz/beginner-s-guide-to-capsule-networks
- https://github.com/sekwiatkowski/awesome-capsule-networks
- https://www.youtube.com/watch?v=pPN8d0E3900
- https://jhui.github.io/2017/11/03/Dynamic-Routing-Between-Capsules/
- https://jhui.github.io/2017/11/14/Matrix-Capsules-with-EM-routing-Capsule-Network/
- Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. “Dynamic Routing Between Capsules.” *ArXiv:1710.09829 [Cs]*, October 26, 2017. http://arxiv.org/abs/1710.09829.
- Hinton, Geoffrey, Sara Sabour, and Nicholas Frosst. “MATRIX CAPSULES WITH EM ROUTING,” 2018, 15.

  https://openreview.net/pdf?id=HJWLfGWRb
- Geirhos, Robert, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. “ImageNet-Trained CNNs Are Biased towards Texture; Increasing Shape Bias Improves Accuracy and Robustness,” September 27, 2018. https://openreview.net/forum?id=Bygh9j09KX.
- Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps.” *ArXiv:1312.6034 [Cs]*, December 20, 2013. http://arxiv.org/abs/1312.6034.
- Hernández-García, Alex, and Peter König. “Do Deep Nets Really Need Weight Decay and Dropout?” *ArXiv:1802.07042 [Cs]*, February 20, 2018. http://arxiv.org/abs/1802.07042.
- https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html