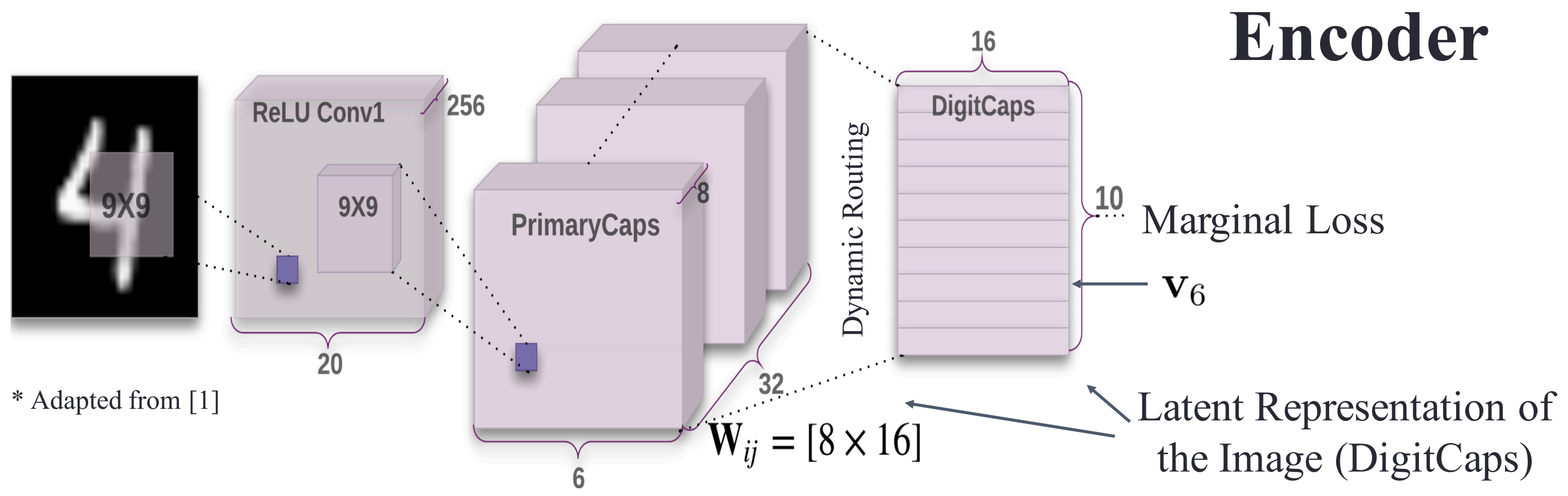
[9.520] Learning Disentangled Latent Representations with CapsNet

Chuanquan (Charles) Shu

OBJECTIVE STATEMENT

A distinguishing feature of Capsule Neural Network (CapsNet) is that the input and output neurons of a Caps Layer are vectors. The orientation of a neuron vector encodes features of a particular entity, but the semantic meanings of such orientation are entangled and thus uninterpretable. In this project, I seek to use a training procedure that enables CapsNet to learn disentangled and semantically interpretable latent representations. The result is that specified dimensions of a neuron vector will one-to-one encode only the chosen features.

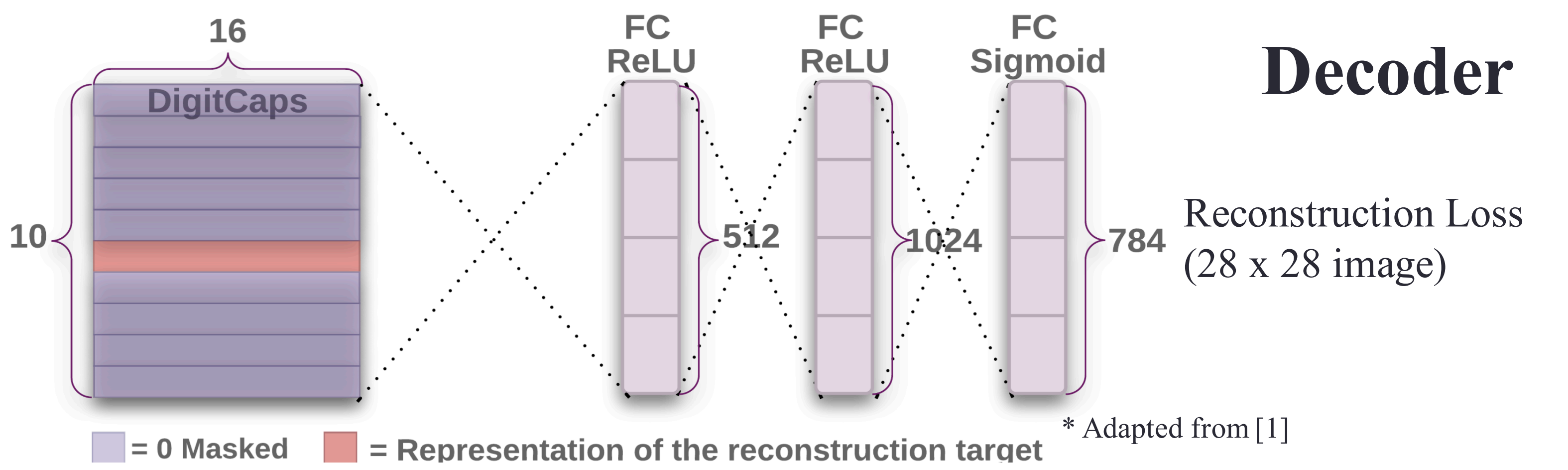
OVERVIEW OF CAPSNET STRUCTURE



* Adapted from [1]

Marginal Loss: $L_c = T_c\ max(0, 0.9 - ||\mathbf{v}_c||)^2 + 0.5(1 - T_c)\ max(0, ||\mathbf{v}_c|| - 0.1)^2$

$\mathbf{v}_c$:
- DigitCap c (or neuron vector c).
- $||\mathbf{v}_c||$ close to 1 if digit c exists in the image; close to 0 if not.


Decoder
16
DigitCaps
10
FC ReLU
512
FC ReLU
1024
FC Sigmoid
784
Reconstruction Loss (28 x 28 image)
= 0 Masked
= Representation of the reconstruction target

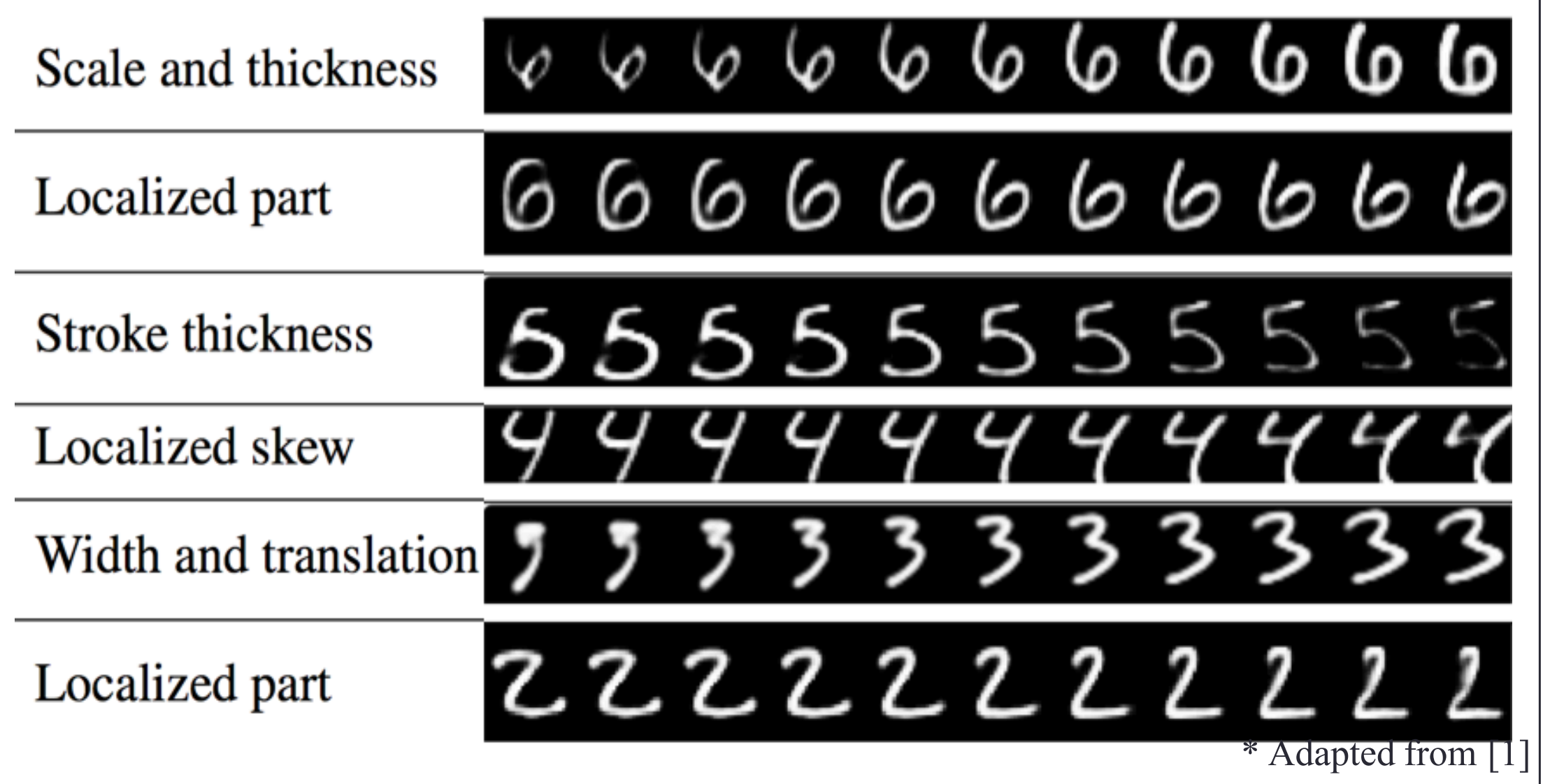
* Adapted from [1]

Mask (set to 0) DigitCaps according to the training image label (5 in the above illustration).

$$\text{Total Loss} = \sum_{c=0}^{9} L_c + \lambda \cdot \text{Reconstruction Loss}$$

Training with standard back propagation. Fully trained CapsNet can reconstruct input images well; the 16-dimensional DigitCap has learnt to span the space of variations of a given digit.

* Adapted from [1]



* Adapted from [1]

Each row shows the reconstruction when 1 of the 16 dimensions in the DigitCaps is perturbed by intervals of 0.05 in the range [-0.25, 0.25].

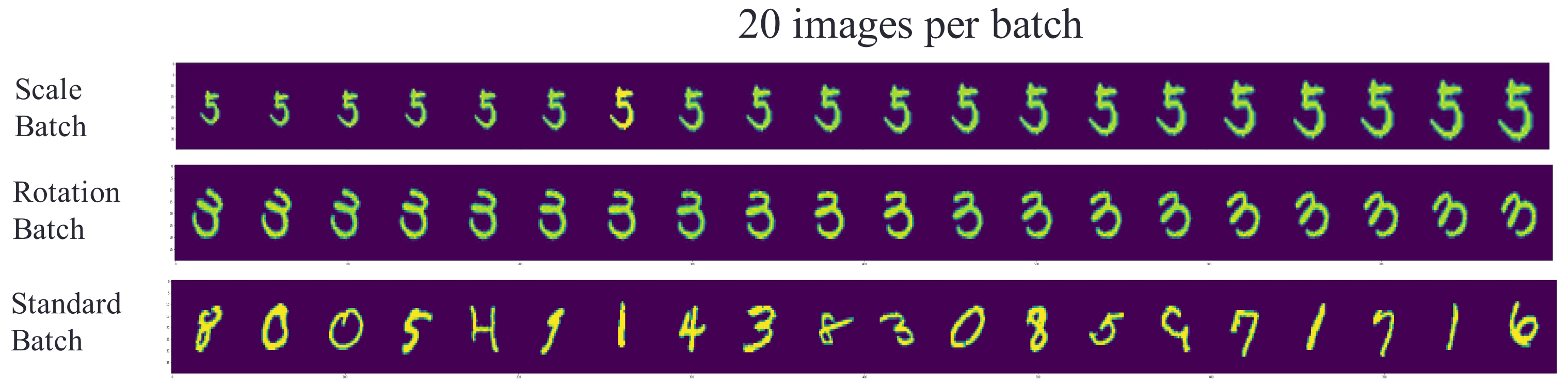
Issues

Entangled Semantic Meanings:
- The mapping between dimensions and features is not one-to-one.
- Dimension does not consistently map to a fixed feature of an image.

Hard to Interpret & to do Controlled Generation

TRAINING PROCEDURE

Disentanglement for Rotation and Scale

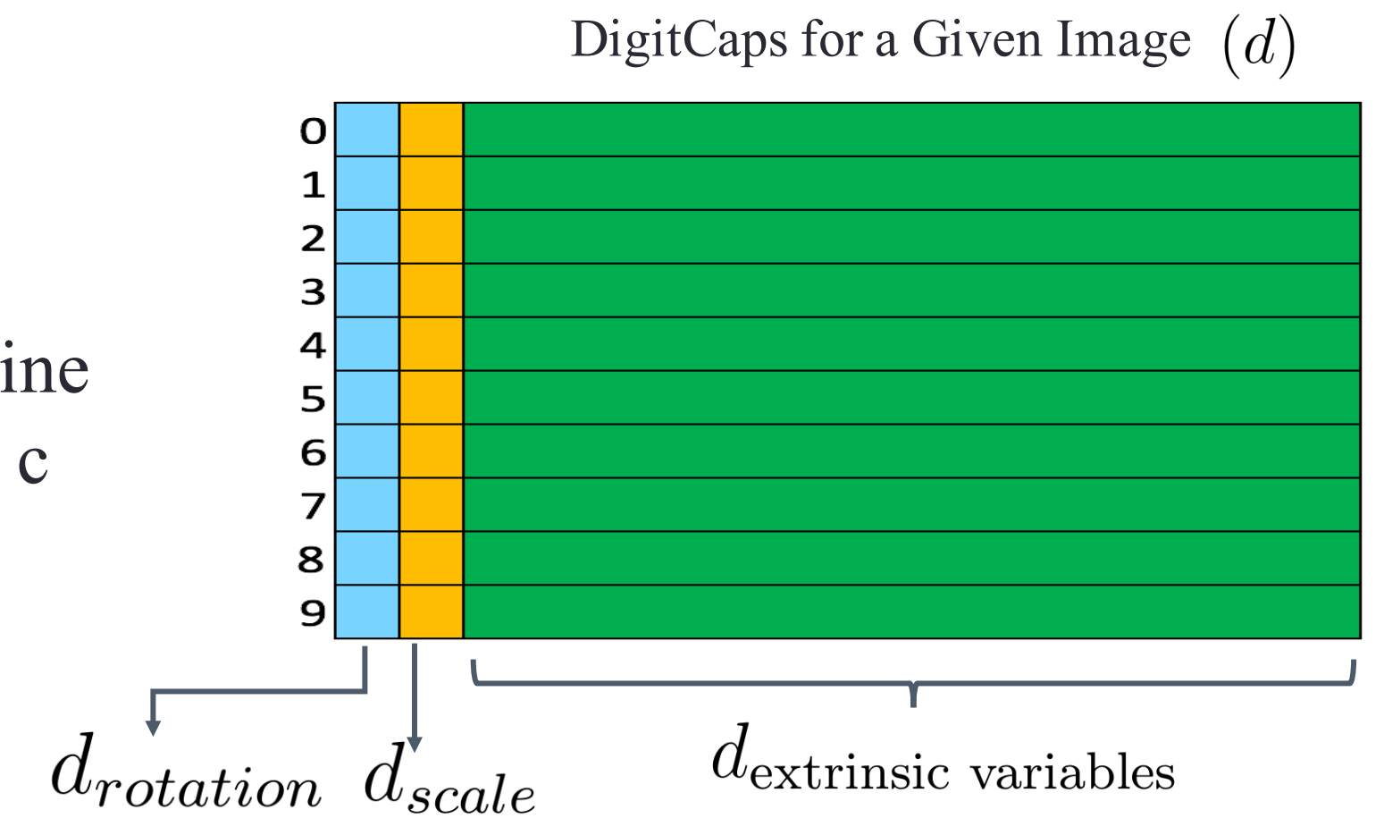


Standard batches from MNIST. Rotation and Scale batches are created: randomly selected 100 images per digit, apply 20 rotating angles and scaling factors.

$d_{c,scale}$ controls scaling for digit c

$d_{c,rotation}$ controls rotation for digit c

$d_{c,\text{extrinsic variables}}$ controls extrinsic variables (line thickness, style, etc) for digit c

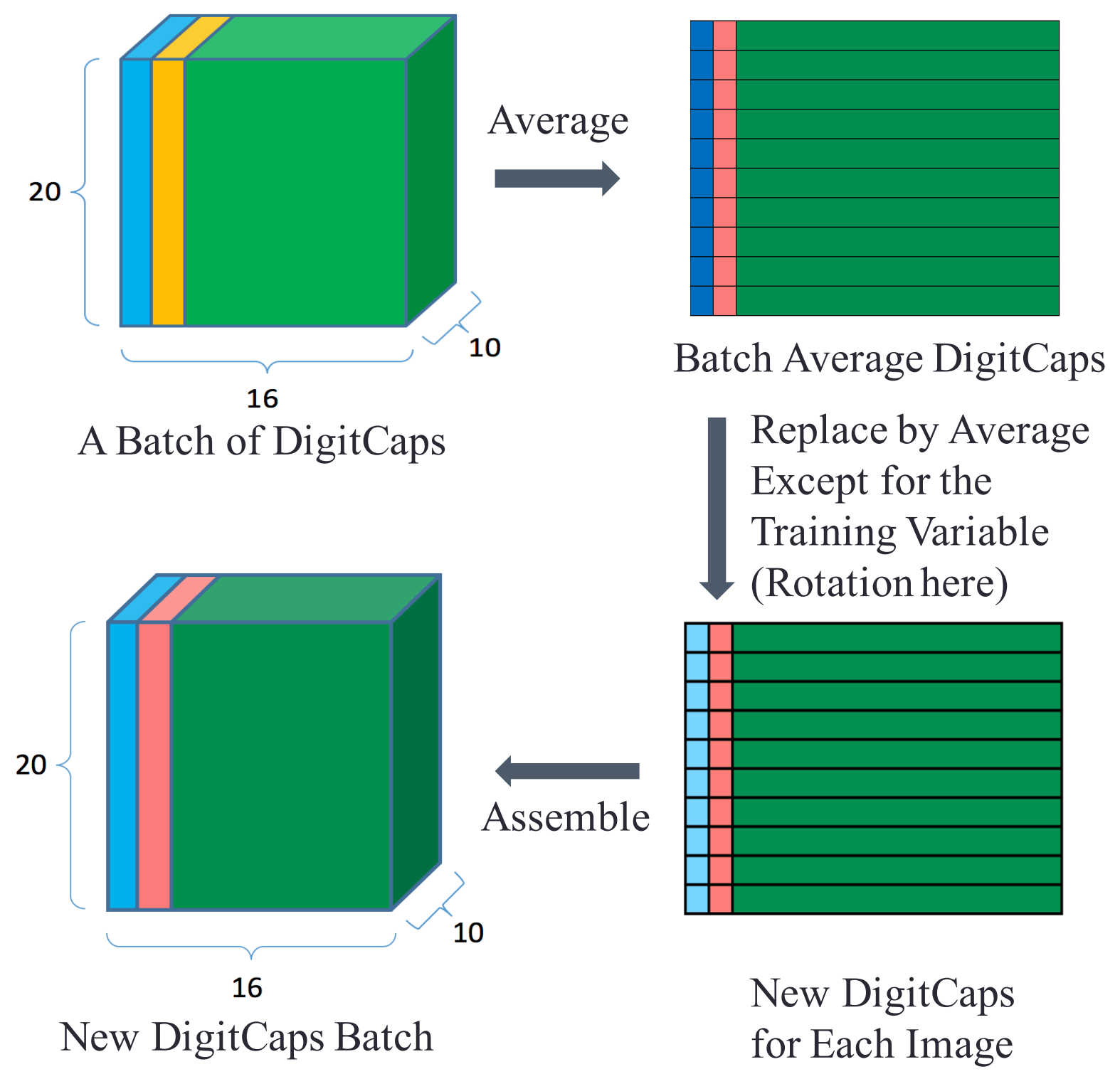

DigitCaps for a Given Image (d)
0
1
2
3
4
5
6
7
8
9
$d_{rotation}$
d_{scale}
$d_{\text{extrinsic variables}}$

Steps

1. Select randomly a group of variable(s) to train (rotation, scale, extrinsic variables) with probabilities (17%, 17%, 66%) because extrinsic variables has higher dimension.
2. Randomly select a batch in which only that group of variable(s) changes.

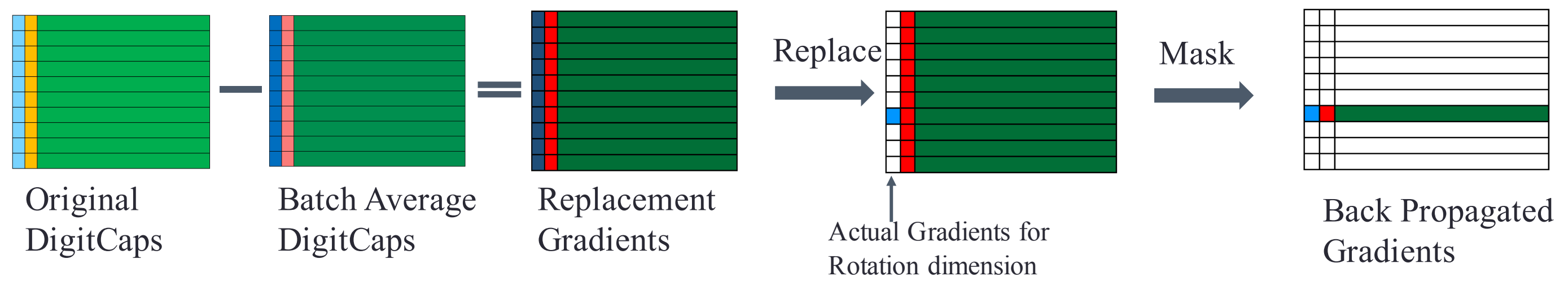
Forward

3. Use CapsNet to encode* each image in the batch into DigitCaps.
4. Calculate an "average" DigitCaps across the batch (reducing the cube into 16x10 matrix).
5. For the DigitCaps of each image, replace all values with the Batch Average values except for the dimension(s) corresponding to the training variable(s). Note the new cube (New DigitCaps Batch) has no variation across each image (height) except for the Rotate dimension.
6. Apply masking and send the new cube to the decoder, computing Total Loss.


20
16
10
A Batch of DigitCaps
Average
Batch Average DigitCaps
Replace by Average Except for the Training Variable (Rotation here)
New DigitCaps for Each Image
Assemble
20
16
10
New DigitCaps Batch

Backward

7. For each image's DigitCaps, replace their gradients with their difference from the mean. The gradient(s) of the training variable(s) is unchanged. Continue back propagation.



ACKNOWLEDGEMENTS & NOTES

❖ [1] Sara Sabour, Nicholas Frosst, Geoffrey E. Hinton. Dynamic Routing Between Capsules. *arXiv: 1710.09829v1*, 2017.

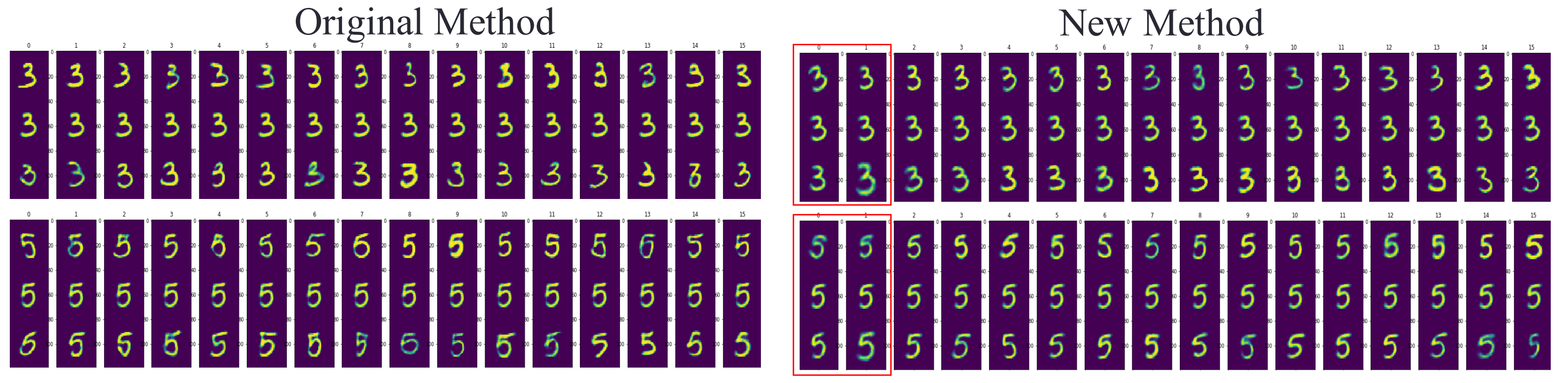
❖ [2] Tejas D. Kulkarni, Will Whitney, Pushmeet Kohli, Joshua B. Tenenbaum. Deep Convolutional Inverse Graphics Network. arXiv: 1503.03167v4, 2015.

❖ Special Thanks to Naturomics for making his implementation of CapsNet public on Github. https://github.com/naturomics/CapsNet-Tensorflow

(*) Custom modification to CapsNet encoding: Because first 2 dimensions are used to learn rotation and scale, probability of entity existence is measured using the length of the remaining 14 dimensions. The original model uses all 16 dimensions.

(**) Results and comparisons based on just 30,000 training steps of 20-image batches each, slightly modified hyperparameters (for original model) to work on 40x40 MNIST. Both models reach the state-of-art accuracy but reconstruction quality is not fabulous.

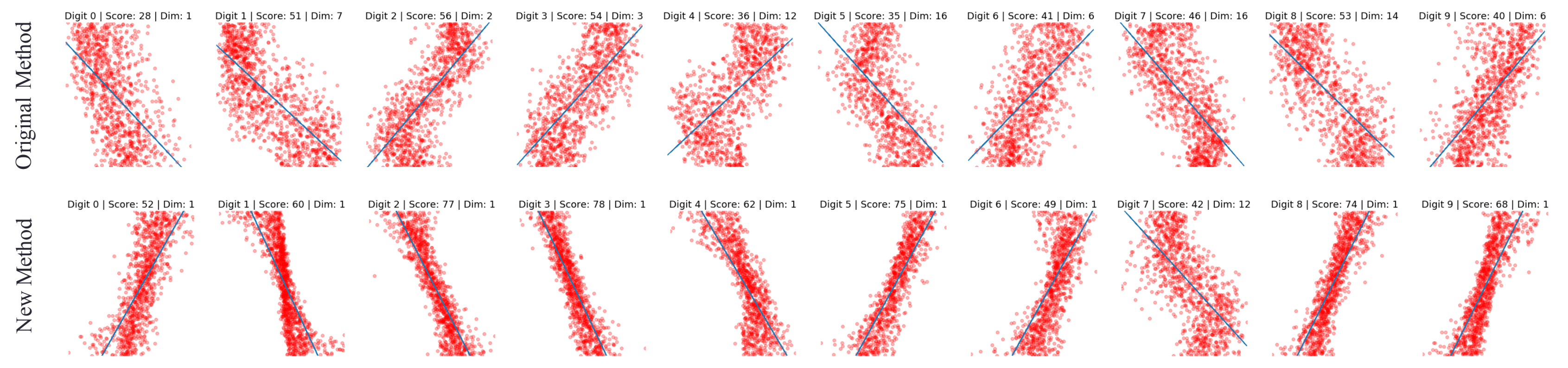
RESULTS**



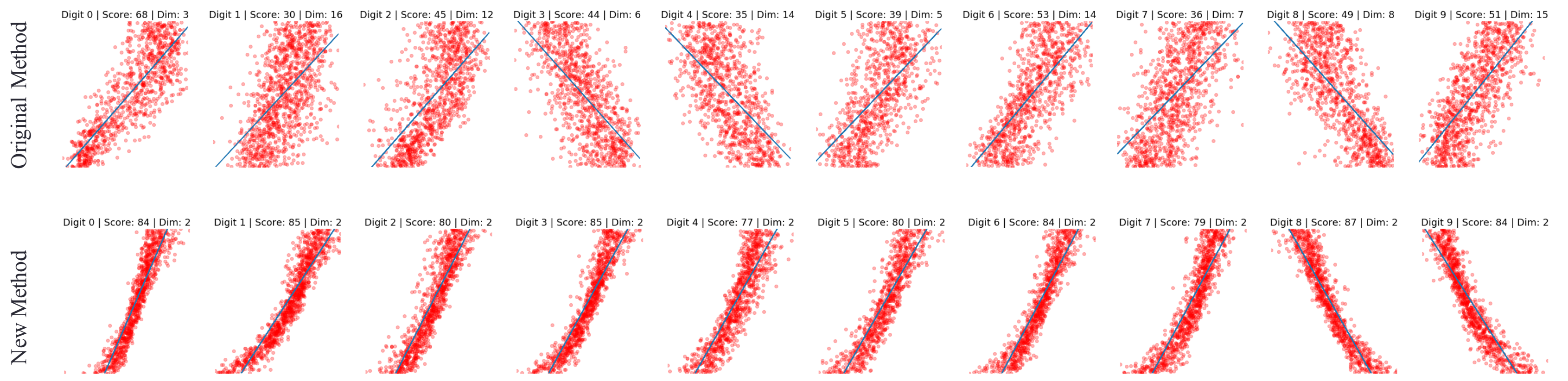
- Perturbation of the individual dimension by ±0.5 (except for the first 2 dimensions of the New Method with ±1) for randomly selected test set digits 3 and 5 under both methods.
- Under the Original Method, the dimensions affecting rotation and scale are not fixed. Perturbing candidate dimensions also affect features other than rotation and scale.
- Under the New Method, the dimensions affecting rotation and scale are 1 and 2, respectively. Other dimensions have limited to no relevance. Perturbing dimensions 1 and 2 largely affects rotation and scale only (no effect on other features). **Controlled Generation**.

Comprehensive Evaluation

Rotation



Scale



- Separately regress by digits the "true" rotation angle and the "true" scaling factor vs the best explaining dimension (among 16). Plots and scores (R-squared x 100) based on holdout dataset.
- Under original method, the best explaining dimension is uncertain across different digits for both rotation and scaling scenarios. Under the new method, for rotation, only digit 7 does not have the 1st dimension as the best explaining dimension. For scaling, all digits have the 2nd dimension as the best explaining dimension. **Exactly as Prescribed.**
- For rotation, the mean best score under the original method is 45 while that under the new method is 63. For scaling, the mean best score under the original method is 44 while that under the new method is 83. **Disentangled Semantic Interpretation.**

DISCUSSION

Why Does It Work

- When training using a non-standard batch (with 1 varying feature), only the values of the corresponding dimension varies while those in other dimensions are held fixed (forward). In addition, only the values of that dimension gets updated by the real gradients (backward). That dimension is hence forced to explain everything about the varying feature and nothing else.

Why Doesn't It Work Perfectly

- "Standard" batches not fully standardized; MNIST innate variance in rotation (huge) and scale.
- Decoder not flexible/powerful enough when the difference in a semantically meaningful feature is only encoded in 1 number.