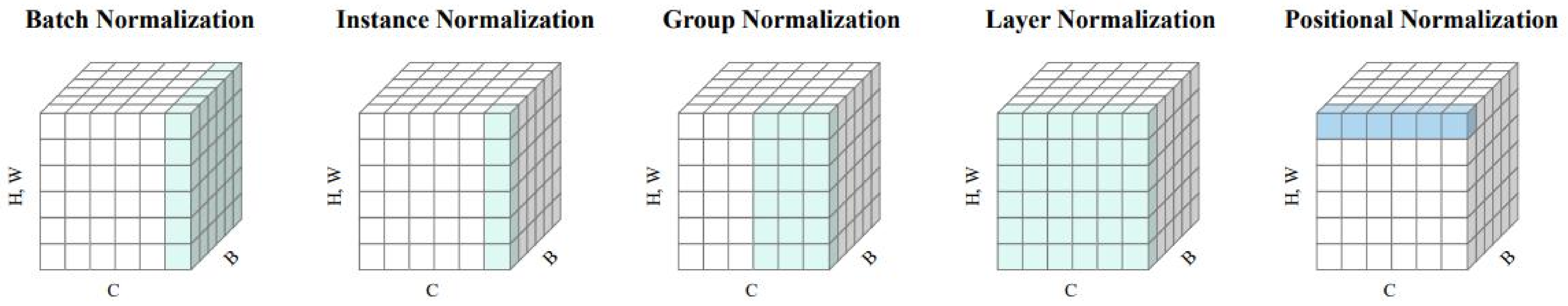
POSITIONAL NORMALIZATION

Normalization attribute

- reducing internal covariate shift
- mean = 0, variance = 1

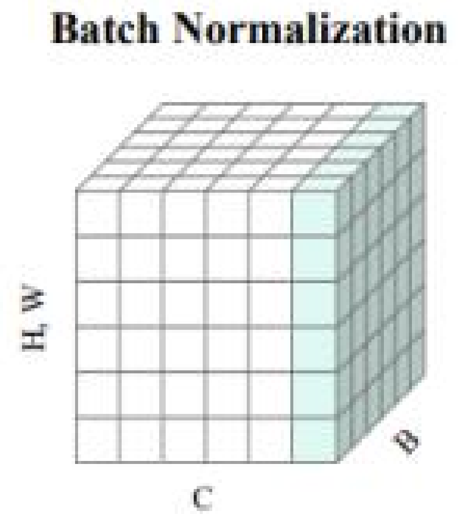


Normalization benefit

- Batch normalization: benefits training of deep networks primarily in computer vision tasks
- Group normalization: the first choice for small mini-batch settings such as object detection and instance segmentation tasks
- Layer Normalization: suited to NLP tasks
- Instance normalization: used in image synthesis owing to its apparent ability to remove style information from the inputs
- PONO: deal with spatial information, primarily targeted at generative and sequential models

Batch Normalization

$$X'_{b,c,h,w} = \gamma\left(\frac{X_{b,c,h,w} - \mu}{\sigma}\right) + \beta.$$



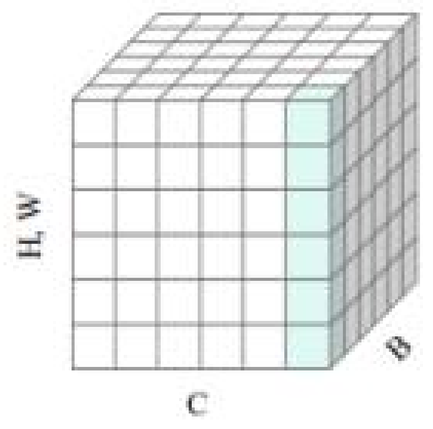
Batch Normalization (BN) (Ioffe and Szegedy, 2015) computes μ and σ *across the B, H, and W dimensions*. BN increases the robustness of the network with respect to high learning rates and weight initializations (Bjorck et al., 2018), which in turn drastically improves the convergence rate.

Synchronized Batch Normalization treats features of mini-batches across multiple GPUs like a single mini-batch

Instance Normalization

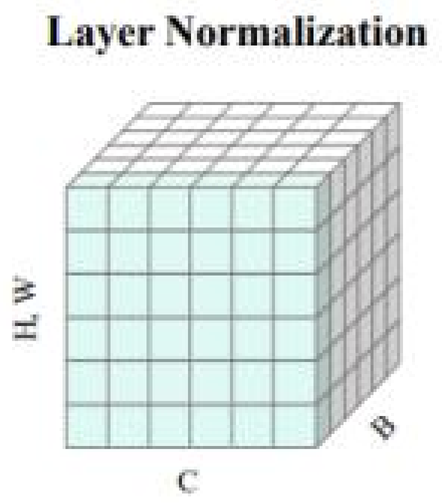
- Instance Normalization (IN) (Ulyanov et al., 2016) treats each instance in a mini-batch independently and computes the statistics across only spatial dimensions (H and W).
- IN aims to make a small change in the stylization architecture results in a significant qualitative improvement in the generated images.

Instance Normalization



Layer Normalization

- Layer Normalization (LN) normalizes all features of an instance within a layer jointly, i.e., calculating the statistics over the C, H, and W dimensions.
- LN is beneficial in natural language processing applications (Lei Ba et al., 2016; Vaswani et al., 2017).



- Conditional instance normalization (CIN) (Dumoulin et al., 2017) keeps a set parameter of pairs $\{(\beta i , \gamma i) | i \in \{1, . . . , N\}\}$ which enables the model to 2 have N different behaviors conditioned on a style class label i.
- Adaptive instance normalization (AdaIN) (Huang and Belongie, 2017) generalizes this to an infinite number of styles by using the μ and σ of IN borrowed from another image as the β and γ.
- Dynamic Layer Normalization (DLN) (Kim et al., 2017) relies on a neural network to generate the β and γ. Later works (Huang et al., 2018; Karras et al., 2018) refine AdaIN and generate the β and γ of AdaIN dynamically using a dedicated neural network.
- Conditional batch normalization (CBN) (De Vries et al., 2017) follows a similar spirit and uses a neural network that takes text as input to predict the residual of β and γ, which is shown to be beneficial to visual question answering models.

- Spatially Adaptive Denormalization (SPADE) (Park et al., 2019), *an extension of Synchronized Batch Normalization* with dynamically predicted weights, generates the spatially dependent β, γ ∈ R B×C×H×W using a two-layer ConvNet with raw images as inputs.

PONO

$$\mu_{b,h,w} = \frac{1}{C}\sum_{c=1}^{C} X_{b,c,h,w}, \quad \sigma_{b,h,w} = \sqrt{\frac{1}{C}\sum_{c=1}^{C}\left(X^2_{b,c,h,w} - \mu_{b,h,w}\right) + \epsilon}$$

where ϵ is a tiny stability constant (*e.g.*, 10^{-5}).

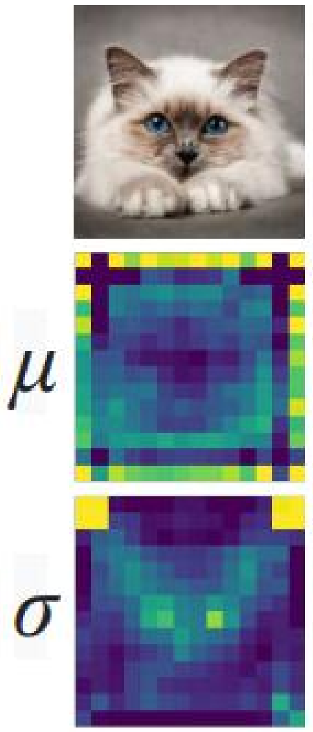


Figure 3: PONO statistics of DenseBlock-3 of a pretrained DenseNet-161.

PONO-MS

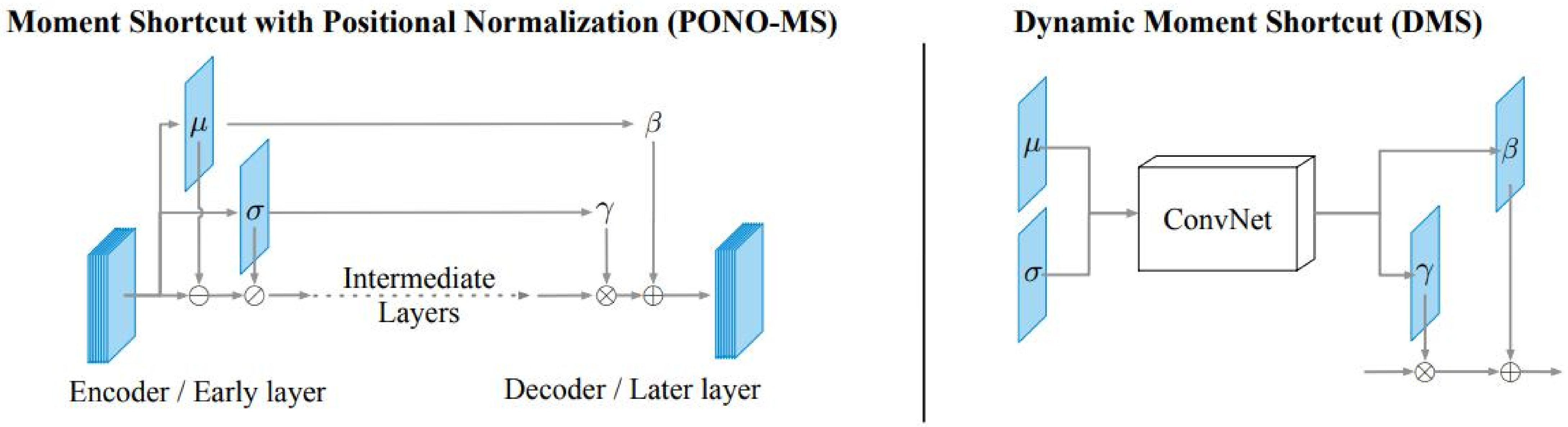
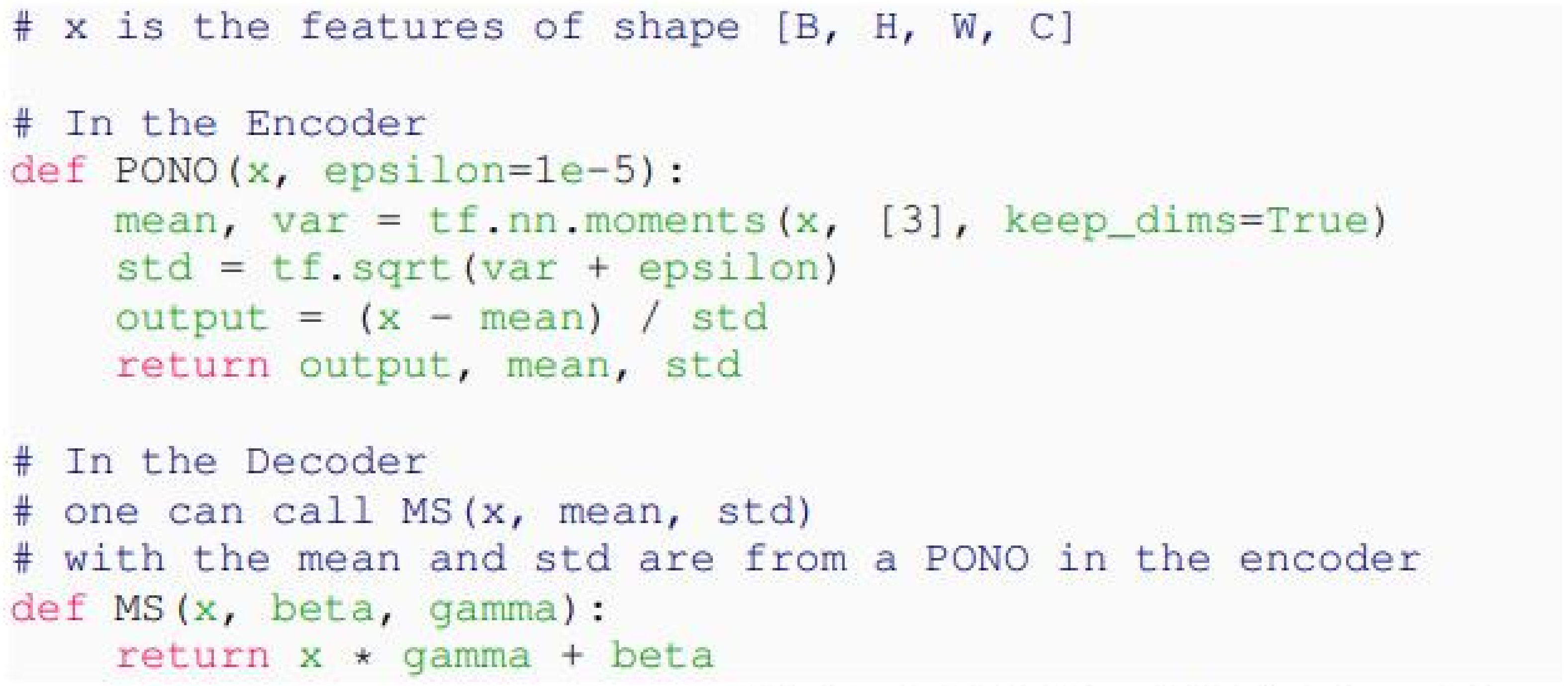


Figure 4: Left: PONO-MS directly uses the extracted mean and standard deviation as β and γ. Right: Optionally, one may use a (shallow) ConvNet to predict β and γ dynamically based on μ and σ.

PONO in Tensorflow

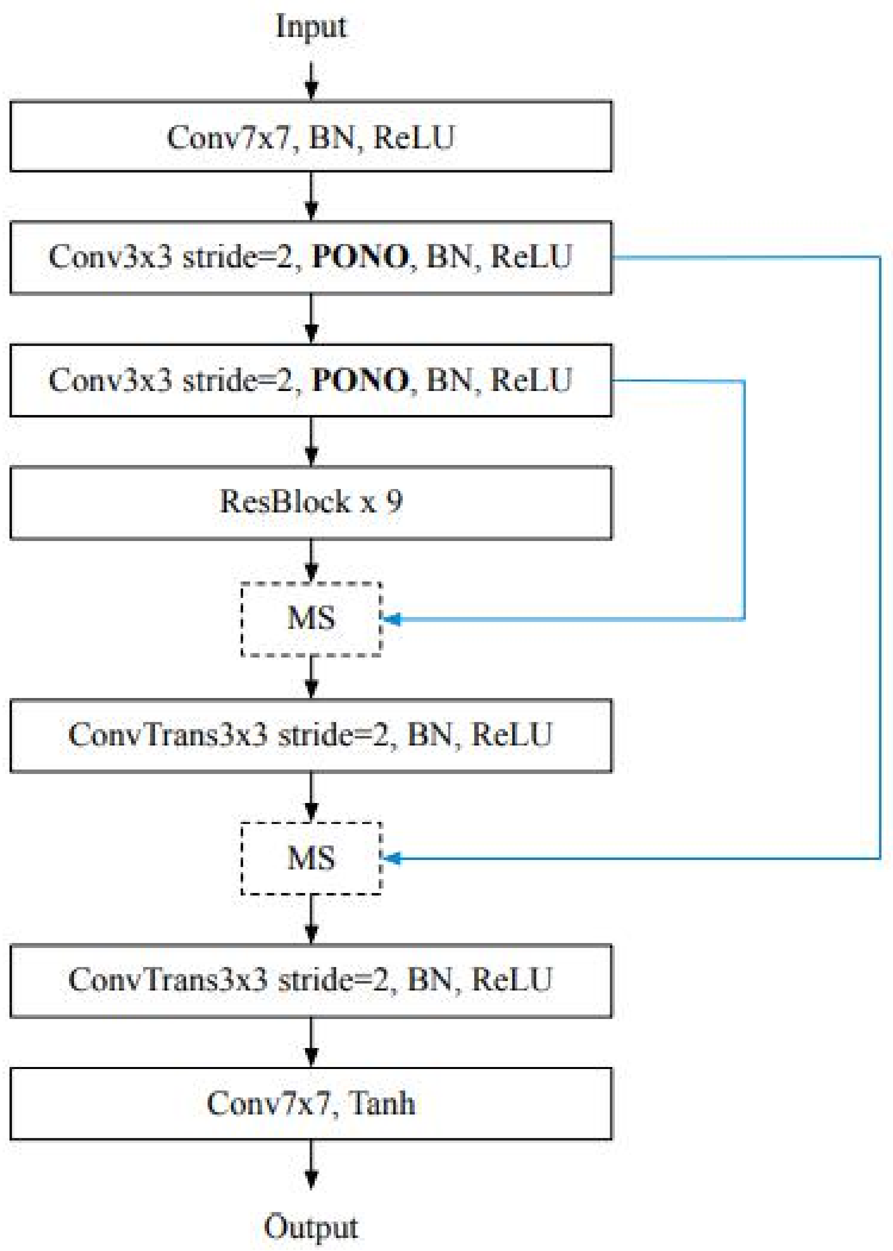


```
# x is the features of shape [B, H, W, C]

# In the Encoder
def PONO(x, epsilon=1e-5):
    mean, var = tf.nn.moments(x, [3], keep_dims=True)
    std = tf.sqrt(var + epsilon)
    output = (x - mean) / std
    return output, mean, std

# In the Decoder
# one can call MS(x, mean, std)
# with the mean and std are from a PONO in the encoder
def MS(x, beta, gamma):
    return x * gamma + beta
```

Listing 1: PONO and MS in TensorFlow



Input
Conv7x7, BN, ReLU
Conv3x3 stride=2, **PONO**, BN, ReLU
Conv3x3 stride=2, **PONO**, BN, ReLU
ResBlock x 9
MS
ConvTrans3x3 stride=2, BN, ReLU
MS
ConvTrans3x3 stride=2, BN, ReLU
Conv7x7, Tanh
Output