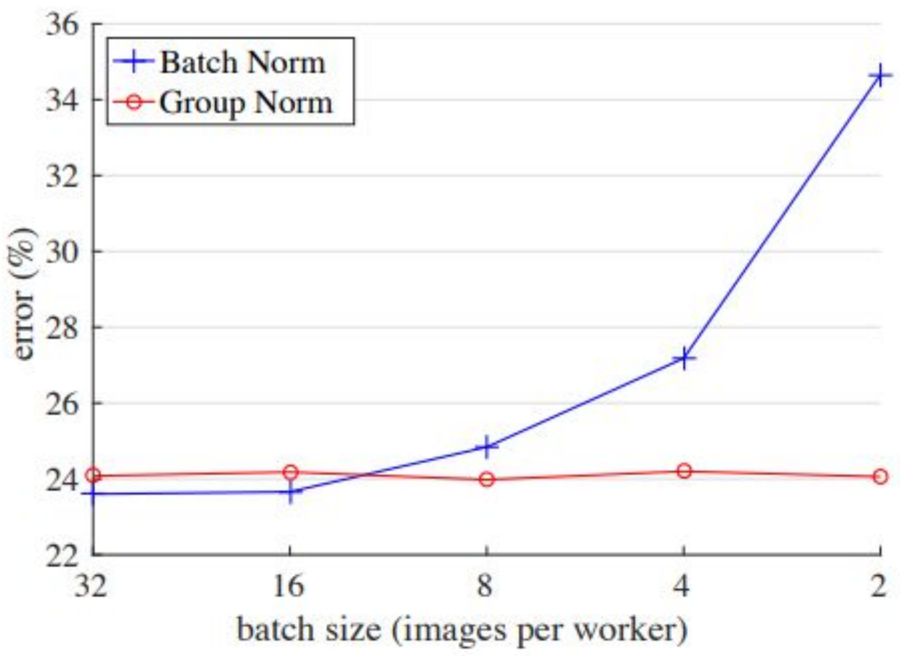
Group Normalization

Presenter：Baosong Yang

Motivation

- Batch Normalization:
  - Error increases rapidly when the batch size becomes smaller (OOM)
  - The batch sizes are inconsistent in training and testing (mean and variance)
  - How to avoid normalizing the batch?



Normalization

- Shift and scale to get standard distribution

$$\hat{x}_i = \frac{1}{\sigma_i}(x_i - \mu_i). \qquad \mu_i = \frac{1}{m}\sum_{k\in\mathcal{S}_i} x_k, \quad \sigma_i = \sqrt{\frac{1}{m}\sum_{k\in\mathcal{S}_i}(x_k - \mu_i)^2 + \epsilon},$$

- Re-shift and re-scale to guarantee the expressive

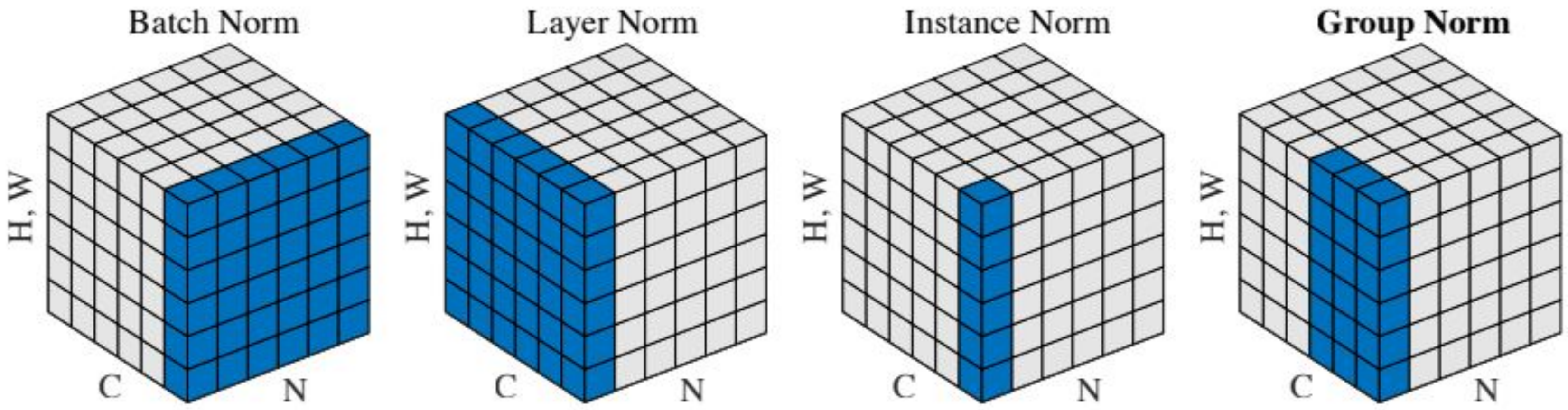
$$y_i = \gamma\hat{x}_i + \beta,$$

GN

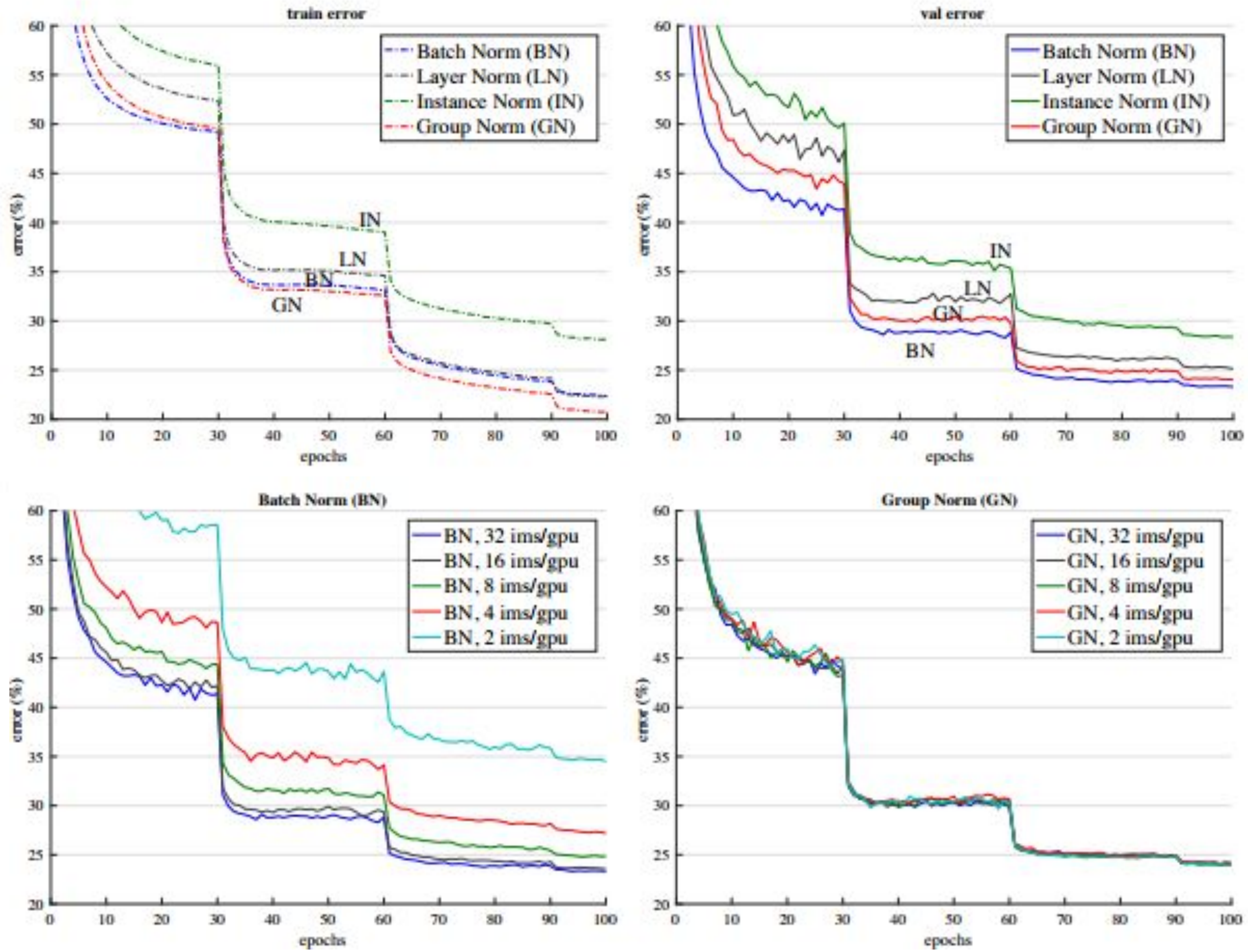
- GN divides the channels into groups.
- The mean and variance for normalization are computed within each group.
- GN's computation is independent of batch sizes.

Comparing with other nomalization

- Batch Normalization: [N， H， W] $\mathcal{S}_i = \{k \mid k_C = i_C\},$
- Layer Normalization: [C， H， W] $\mathcal{S}_i = \{k \mid k_N = i_N\}$
- Instance Normalization: [H， W] $\mathcal{S}_i = \{k \mid k_N = i_N, k_C = i_C\}$
- Group Normalization: [C//G , H, W] $\mathcal{S}_i = \{k \mid k_N = i_N, \lfloor \frac{k_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor\}.$



Experiments



Experiments

# groups (G)						
64	32	16	8	4	2	1 (=LN)
24.6	**24.1**	24.6	24.4	24.6	24.7	25.3
0.5	-	*0.5*	*0.3*	*0.5*	*0.6*	*1.2*

# channels per group						
64	32	16	8	4	2	1 (=IN)
24.4	24.5	**24.2**	24.3	24.8	25.6	28.4
0.2	*0.3*	-	*0.1*	*0.6*	*1.4*	*4.2*

Table 3. **Group division**. We show ResNet-50's validation error (%) in ImageNet, trained with 32 images/GPU. (Top): a given number of groups. (Bottom): a given number of channels per group. The last rows show the differences with the best number.

Conclusion

- GN is less restricted than LN (Why multi-head is better than single-head)
- IN misses the opportunity of exploiting the channel dependence. (A support for CSAN)
- Improving Transformer with Head Normalization？
- Why not normalize along length and batch?

```
def layer_norm(inputs, epsilon=1e-6, dtype=None, scope=None):
    """
    Layer Normalization
    :param inputs: A Tensor of shape [..., channel_size]
    :param epsilon: A floating number
    :param dtype: An optional instance of tf.DType
    :param scope: An optional string
    :returns: A Tensor with the same shape as inputs
    """
    with tf.variable_scope(scope, default_name="layer_norm", values=[inputs],
                           dtype=dtype):
        channel_size = inputs.get_shape().as_list()[-1]

        scale = tf.get_variable("scale", shape=[channel_size],
                                initializer=tf.ones_initializer())

        offset = tf.get_variable("offset", shape=[channel_size],
                                 initializer=tf.zeros_initializer())

        mean = tf.reduce_mean(inputs, axis=-1, keep_dims=True)
        variance = tf.reduce_mean(tf.square(inputs - mean), axis=-1,
                                  keep_dims=True)

        norm_inputs = (inputs - mean) * tf.rsqrt(variance + epsilon)

        return norm_inputs * scale + offset
```