# Week 7

# LayerNorm Optimization

- Implementation 1:
  - Loop 1: mean
  - Loop 2: variance
  - Loop 3: scaling

- Implementation 2:
  - Loop 1: mean & variance
    - Use current mean to calculate variance
  - Loop 2: scaling

```python
def forward(self, x):
    row = x.size()[-2]
    col = x.size()[-1]

    # Initialize mean and variacne with zero
    # mean = x[0, :, 0] * 0
    # var  = x[0, :, 0] * 0

    mean = x[:, 0] * 0
    var  = x[:, 0] * 0

    for i in range(col):
        # data = x[0, :, i];
        data = x[:, i];
        mean = mean + data
        var += torch.square(data - mean / (i + 1))
    var = var / (col - 1)
    mean = mean / col

    mean.unsqueeze_(1)
    var.unsqueeze_(1)

    return self.alpha * (x - mean) / torch.sqrt(var + self.eps) + self.beta
```

# LayerNorm Optimization (Accuracy Analysis)

- Approach 1:
  - Fine-tune a pretrained BERT-base model on CoLA classification dataset
  - Replace all LayerNorm with LayerNorm_v2 (keep the parameters)

```
# embedding layer
param = {}
# store the parameters
for name, data in new_model._modules['bert']._modules['embeddings']._modules['LayerNorm'].named_parameters():
    param[name] = data
# replace LayerNorm with LayerNorm_v2 (same parameters)
new_model._modules['bert']._modules['embeddings']._modules['LayerNorm'] = LayerNorm_v2(param['weight'], param['bias'])
```

- Evaluate with MCC metric (Matthews's correlation coefficient)
- The new model's predictions are stochastic (MCC = 0.1, baseline = 0.57)

Tutorial Source: Chris McCormick and Nick Ryan. (2019, July 22). BERT Fine-Tuning Tutorial with PyTorch. Retrieved from http://www.mccormickml.com

# LayerNorm Optimization (Accuracy Analysis)

- Approach 2:
  - Replace LayerNorm in the pretrained model
    - With trainable parameters (weight, bias)
  - Fine-tune the new model
    - No loss drop on validation set
  - Still very low MCC score

```
Running Validation...
  Accuracy: 0.71
  Validation Loss: 0.61
  Validation took: 0:00:25
```

```
Running Validation...
  Accuracy: 0.71
  Validation Loss: 0.61
  Validation took: 0:00:25
```

```
Running Validation...
  Accuracy: 0.71
  Validation Loss: 0.60
  Validation took: 0:00:24
```

```
Running Validation...
  Accuracy: 0.71
  Validation Loss: 0.60
  Validation took: 0:00:25
```

# LayerNorm Optimization (Accuracy Analysis)

- Approach 3 (transformer from scratch):
  - Build the model layer by layer, LayerNorm vs LayerNorm_v2
  - Extremely slow when training LayerNorm_v2

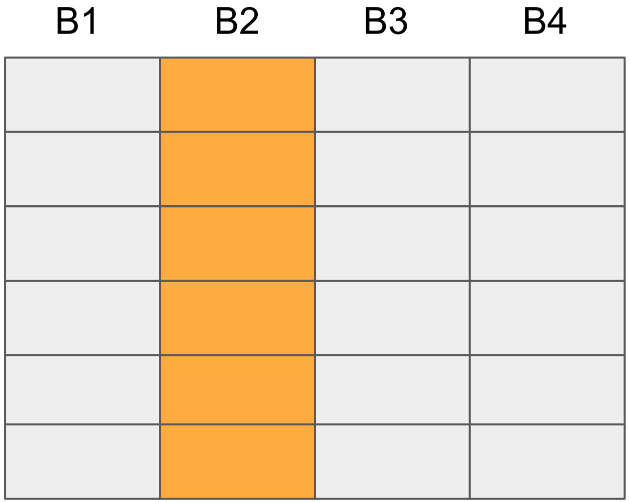
```
Epoch Step:      1 | Accumulation Step:   1 | Loss:   2.95 | Tokens / Sec:   132.1
Epoch Step:     41 | Accumulation Step:   5 | Loss:   2.73 | Tokens / Sec:   134.8
Epoch Step:     81 | Accumulation Step:   9 | Loss:   2.78 | Tokens / Sec:   134.5
Epoch Step:    121 | Accumulation Step:  13 | Loss:   2.83 | Tokens / Sec:   134.3
Epoch Step:    161 | Accumulation Step:  17 | Loss:   2.74 | Tokens / Sec:   132.7
Epoch Step:    201 | Accumulation Step:  21 | Loss:   2.68 | Tokens / Sec:   133.9
Epoch Step:    241 | Accumulation Step:  25 | Loss:   2.64 | Tokens / Sec:   134.9
Epoch Step:    281 | Accumulation Step:  29 | Loss:   2.86 | Tokens / Sec:   135.0
Epoch Step:    321 | Accumulation Step:  33 | Loss:   2.40 | Tokens / Sec:   134.2
Epoch Step:    361 | Accumulation Step:  37 | Loss:   2.64 | Tokens / Sec:   135.3
Epoch Step:    401 | Accumulation Step:  41 | Loss:   2.50 | Tokens / Sec:   133.9
Epoch Step:    441 | Accumulation Step:  45 | Loss:   2.62 | Tokens / Sec:   133.7
Epoch Step:    481 | Accumulation Step:  49 | Loss:   2.66 | Tokens / Sec:   134.6
Epoch Step:    521 | Accumulation Step:  53 | Loss:   2.52 | Tokens / Sec:   134.8
Epoch Step:    561 | Accumulation Step:  57 | Loss:   2.61 | Tokens / Sec:   134.9
Epoch Step:    601 | Accumulation Step:  61 | Loss:   2.58 | Tokens / Sec:   133.4
Epoch Step:    641 | Accumulation Step:  65 | Loss:   2.38 | Tokens / Sec:   135.1
Epoch Step:    681 | Accumulation Step:  69 | Loss:   2.49 | Tokens / Sec:   135.5
Epoch Step:    721 | Accumulation Step:  73 | Loss:   2.39 | Tokens / Sec:   132.5
Epoch Step:    761 | Accumulation Step:  77 | Loss:   2.64 | Tokens / Sec:   133.6
Epoch Step:    801 | Accumulation Step:  81 | Loss:   2.20 | Tokens / Sec:   134.5
Epoch Step:    841 | Accumulation Step:  85 | Loss:   2.48 | Tokens / Sec:   133.0
Epoch Step:    881 | Accumulation Step:  89 | Loss:   2.54 | Tokens / Sec:   136.0
```

```
Epoch Step:      1 | Accumulation Step:   1 | Loss:   2.05 | Tokens / Sec:  2080.8
Epoch Step:     41 | Accumulation Step:   5 | Loss:   2.18 | Tokens / Sec:  1715.0
Epoch Step:     81 | Accumulation Step:   9 | Loss:   2.30 | Tokens / Sec:  1721.8
Epoch Step:    121 | Accumulation Step:  13 | Loss:   2.06 | Tokens / Sec:  1725.1
Epoch Step:    161 | Accumulation Step:  17 | Loss:   2.00 | Tokens / Sec:  1745.6
Epoch Step:    201 | Accumulation Step:  21 | Loss:   1.98 | Tokens / Sec:  1717.5
Epoch Step:    241 | Accumulation Step:  25 | Loss:   1.60 | Tokens / Sec:  1738.4
Epoch Step:    281 | Accumulation Step:  29 | Loss:   1.79 | Tokens / Sec:  1733.4
Epoch Step:    321 | Accumulation Step:  33 | Loss:   1.96 | Tokens / Sec:  1719.1
Epoch Step:    361 | Accumulation Step:  37 | Loss:   1.85 | Tokens / Sec:  1703.4
Epoch Step:    401 | Accumulation Step:  41 | Loss:   2.07 | Tokens / Sec:  1720.6
Epoch Step:    441 | Accumulation Step:  45 | Loss:   2.03 | Tokens / Sec:  1731.8
Epoch Step:    481 | Accumulation Step:  49 | Loss:   1.88 | Tokens / Sec:  1733.2
Epoch Step:    521 | Accumulation Step:  53 | Loss:   2.20 | Tokens / Sec:  1726.6
Epoch Step:    561 | Accumulation Step:  57 | Loss:   1.61 | Tokens / Sec:  1694.6
Epoch Step:    601 | Accumulation Step:  61 | Loss:   1.92 | Tokens / Sec:  1735.5
Epoch Step:    641 | Accumulation Step:  65 | Loss:   2.01 | Tokens / Sec:  1711.6
Epoch Step:    681 | Accumulation Step:  69 | Loss:   1.65 | Tokens / Sec:  1714.4
Epoch Step:    721 | Accumulation Step:  73 | Loss:   1.69 | Tokens / Sec:  1719.7
Epoch Step:    761 | Accumulation Step:  77 | Loss:   1.81 | Tokens / Sec:  1732.5
Epoch Step:    801 | Accumulation Step:  81 | Loss:   1.49 | Tokens / Sec:  1730.9
Epoch Step:    841 | Accumulation Step:  85 | Loss:   1.82 | Tokens / Sec:  1717.7
Epoch Step:    881 | Accumulation Step:  89 | Loss:   1.63 | Tokens / Sec:  1726.5
```

- Significantly higher loss (cannot finish the training due to colab GPU limitation)

Tutorial Source: The Annotated Transformer, Austin Huang, Suraj Subramanian, Jonathan Sum, Khalid Almubarak, and Stella Biderman
http://nlp.seas.harvard.edu/annotated-transformer/

# Bias & Concatenation




- Bias weight size: (feature_size, 1)
    - One bias value for a column

- Embedding concatenation in MatMul:
    - Pass the head id to each head
    - Store the result vector to the position after concatenation based on head id

```c
#ifdef SELF_ATTN_TEST
  vse32_v_f32m1(&o[i * dk + j], partial_sum, vl);
#else
  vse32_v_f32m1(&o[i * d_model + k * dk + j], partial_sum, vl);
#endif /* SELF_ATTN_TEST */
```