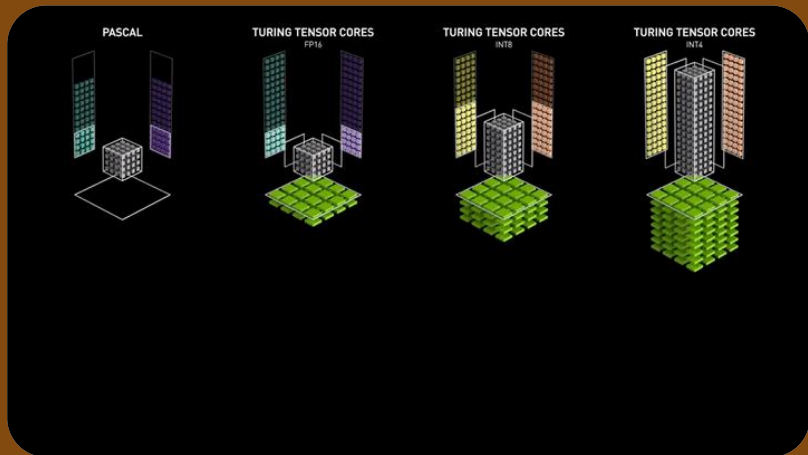


Outlines



Quantization
(TF32, FP32, FP16, BFP16)

Automatic Mixed Precision
(PyTorch)

PyTorch Compiler
Sophia Optimizer

Training BERT Model
(MLM, NSP)

Outlines

Part 1 and 2



References

My GitHub; Codes and Examples

https://github.com/HFarkhari/Mixed_Precision_Training

Mixed-Precision using Tensor Cores Series

<https://youtube.com/playlist?list=PL5B692fm6--vi9vC5EDBFsfTBnrvVbl40>

Training BERT by James Briggs

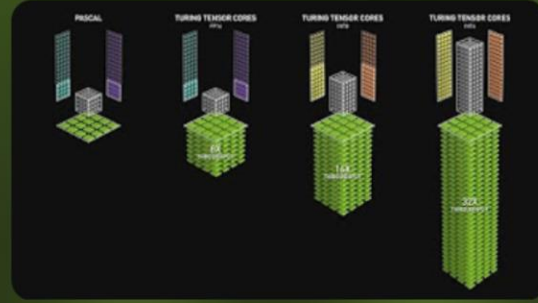
<https://youtube.com/playlist?list=PLIUOU7oqGTLgQ7tCdDT0ARlRoh1127NSO>



Hamed Farkhari
HFarkhari

Follow

Interested in Machine Learning, Deep Learning, Transformers, NLP, Time Series, Mathematics, Statistics, and Engineering.



**Mixed-Precision using
Tensor Cores Series**

NVIDIA Developer

8 videos 1,920 views Last updated on Nov 17, 2019



MLM with BERT

1

Training BERT

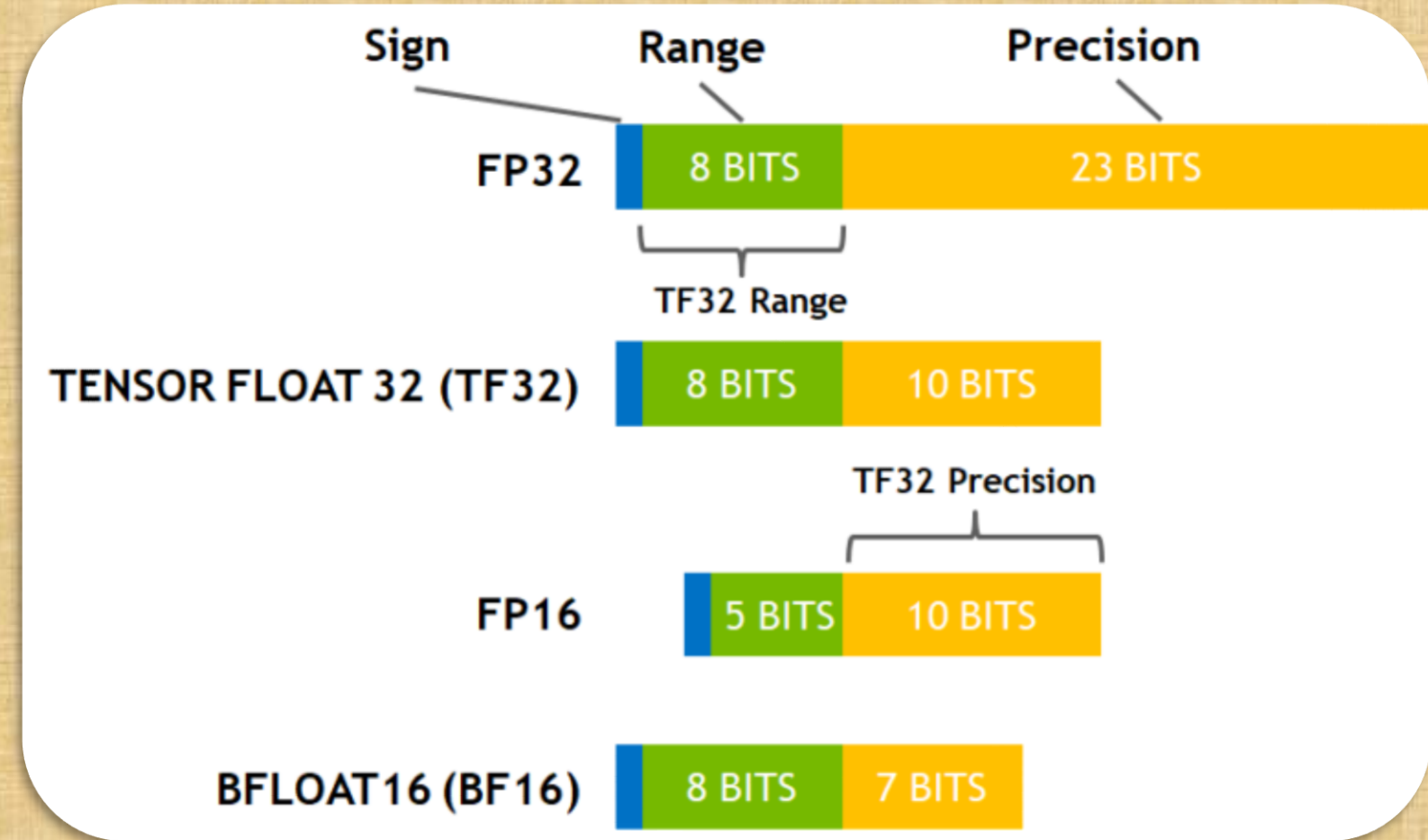
James Briggs

5 videos 7,603 views Last updated on Sep 16, 2021

FP32, TF32, FP16, BF16

<https://developer.nvidia.com/blog/accelerating-ai-training-with-tf32-tensor-cores/>

Quantization
(TF32, FP32,
FP16, BFP16)



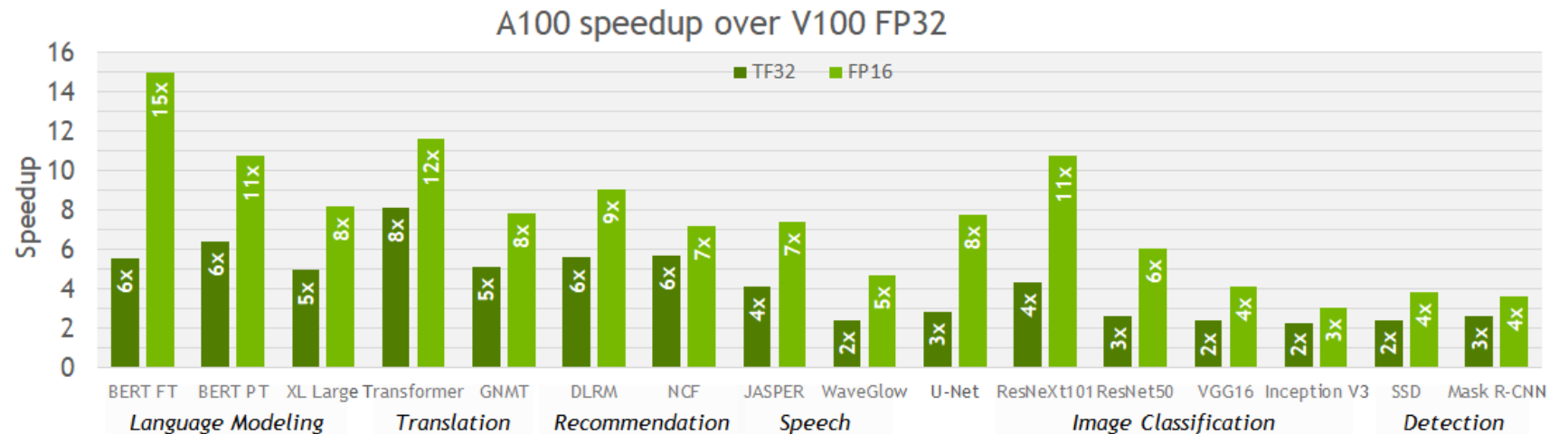
FP32, TF32, FP16, BF16

<https://developer.nvidia.com/blog/accelerating-ai-training-with-tf32-tensor-cores/>

TensorFloat32 (TF32) support

- **FP16** gives a further **speedup of up to ~2x**, as 16-bit Tensor Cores are 2x faster than **TF32 mode**

Quantization
(TF32, FP32,
FP16, BFP16)



FP32, TF32, FP16, BF16

<https://developer.nvidia.com/blog/accelerating-ai-training-with-tf32-tensor-cores/>

TensorFloat32 (TF32) support

- TF32 mode accelerates Only convolution and matrix-multiply layers, including linear and fully connected layers, recurrent cells, and attention blocks.
- TF32 does not apply to layers that are not convolution or matrix-multiply operations (for example, batch normalization), as well as optimizer or solver operations.
- All storage in memory and other operations remain completely in **FP32**, only **convolutions** and **matrix-multiplications** convert their inputs to **TF32** right before multiplication.
- Tensor storage is not changed when training with TF32. Everything remains in FP32, or whichever format is specified in the script.

Quantization
(TF32, FP32,
FP16, BFP16)

TensorFloat32 (TF32) Tips

- TF32 is the **default mode** for AI on A100 when using the NVIDIA optimized deep learning framework containers for TensorFlow, PyTorch, and MX Net, starting with the **20.06**.
- PyTorch 1.7, TensorFlow 2.4, nightly builds for MXNet 1.8 (minimum version).
- cuDNN version 8.0 and greater.
- cuBLAS version 11.0 and greater.
- cuSOLVER
- cuTENSOR version 1.1.0 and greater.

TensorFloat32 (TF32) Tips

- Tensor cores are programmable, with CUDA code and WMMA.
- TF32 mode is the default option for AI training with 32-bit variables on **Ampere GPU architecture (A100, RTX 30 series, RTX 40 series)**.
- **TF32** is only exposed as a Tensor Core operation mode, **not a type**.
- Achieves the **same accuracy** as FP32 training, requires **no changes** to **hyperparameters** for **training scripts**.
- **BF16** is introduced as Tensor Core math mode in **cuBLAS 11.0** and as a numerical type in **CUDA 11.0**.
- Deep learning frameworks and AMP support **BF16**.

TensorFloat32 (TF32) Tips

PERFORMANCE

How does one know tensor cores were used?

```
import torch
import torch.nn
```

```
bsz, inf, outf = 256, 1024, 2048
```

```
tensor = torch.randn(bsz, inf).cuda().half()
layer = torch.nn.Linear(inf, outf).cuda().half()
layer(tensor)
```

FP16 input
FP16 output

Running with:

```
nvprof python test.py
```

Produces (among with other output):

```
37.024us  1  37.024us  37.024us  37.024us  volta_fp16_s884_gemm_fp16.
```

Tensor core 884



<https://youtu.be/tAlakfEt-tI>

Quantization
(TF32, FP32,
FP16, BFP16)

TensorFloat32 (TF32) Tips

```
nvprof --log-file results/nvprof_log.txt python train.py
```

Avg	Min	Max	Name
1.898us	27.456us	3.7446ms	volta_fp16_s884gemm_fp16_128x64_ldg8_f2f_nn
3.74us	58.335us	6.2330ms	volta_fp16_s884gemm_fp16_256x128_ldg8_f2f_nn
1619ms	152.83us	5.5497ms	volta_fp16_s884gemm_fp16_256x128_ldg8_f2f_nt
1.079us	41.760us	1.0913ms	volta_fp16_s884gemm_fp16_256x64_ldg8_f2f_nn
1.27us	36.767us	1.4038ms	volta_fp16_s884gemm_fp16_128x128_ldg8_f2f_nt
3465ms	42.560us	4.1843ms	volta_fp16_sgemm_fp16_128x64_nt

GPU Compute Capability < 8

<https://developer.nvidia.com/cuda-gpus>

Quantization
(TF32, FP32,
FP16, BFP16)

TensorCore Performance Guidance

- **Requirements to trigger TensorCore operations:**
 - Convolutions:
 - Number of input channels a multiple of 8
 - Number of output channels a multiple of 8
 - Matrix Multiplies:
 - M, N, K sizes should be multiples of 8
 - Larger K sizes make multiplications more efficient (amortize the write overhead)
 - Makes wider recurrent cells more practical (K is input layer width)
- **If you're designing models**
 - Make sure to choose layer widths that are multiples of 8
 - Pad input/output dictionaries to multiples of 8
 - Speeds up embedding/projection operations
- **If you're developing new cells**
 - Concatenate cell matrix ops into a single call

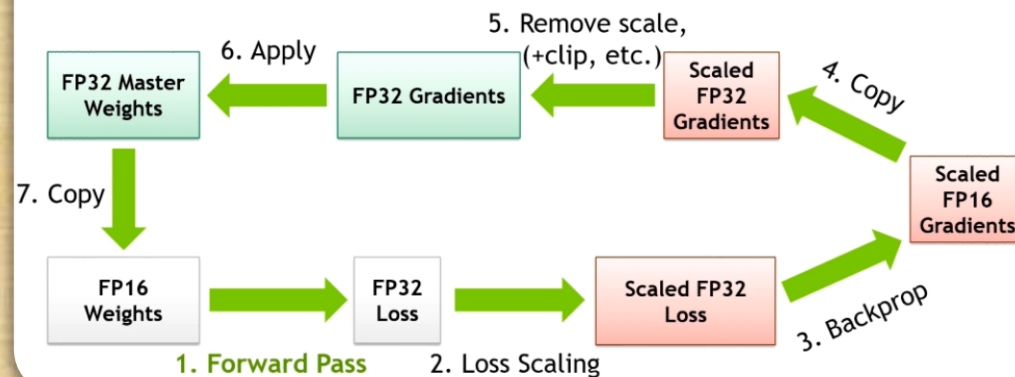
<https://youtu.be/i8-Jw48Cp8w>

FP32, FP16, BFP16

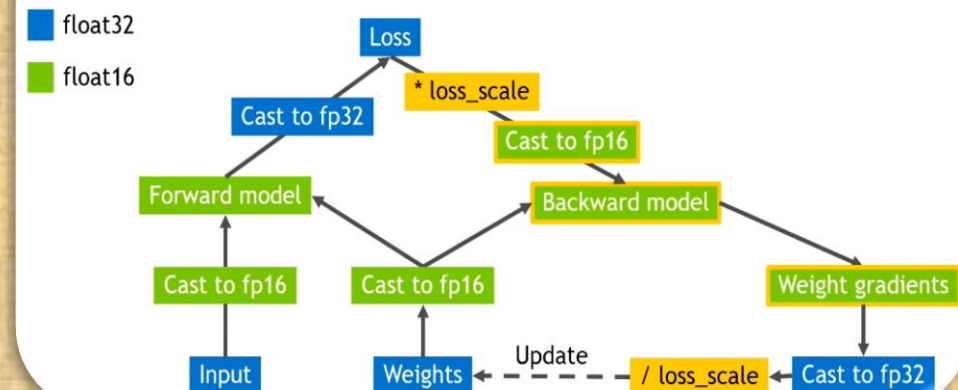
Mixed Precision Process

1. FP16 weights
2. Compute loss fp16
3. Convert loss fp16 (single value) to fp32 just for scaling loss in fp32
4. Scale loss fp32 (*128.0)
5. Compute scaled gradient on fp16
6. Create a copy of scaled gradient from fp16 to fp32
7. Divide scaled gradient (fp32) by scale factor (/128.0)
8. Clip gradient
9. Apply gradient fp32 on fp32 weights (Master weights which is a copy of weights in fp32)
10. Convert fp32 weights to fp16
11. Transfer fp16 weights to the model.

MIXED PRECISION TRAINING



NEW GRAPH

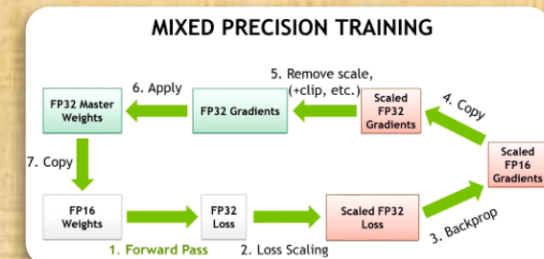
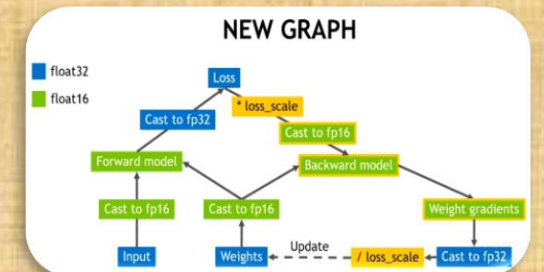


Quantization
(TF32, FP32,
FP16, BFP16)

FP32, FP16, BFP16

Quantization
(TF32, FP32,
FP16, BFP16)

- ### Mixed Precision Process Tips
- Prevent overflow
 - Loss should calculate in fp32
 - logits in fp16 --> should convert to fp32,
 - labels in int32,
 - Calc cross entropy loss(logit(fp32), labels(int32)) --> loss in fp32.
 - You can convert your input tensors from fp32 to fp16.
 - NLP, input tensors are integer (Tokenizer output),
 - Not possible to convert INT32 to fp16.
 - Reductions that make overflow and can not recover from it (should be in fp32)
 - Batch normalization layers, SoftMax layers, Pow, Exp, range expanding math functions.
 - Prevent gradient underflow
 - scale the loss value.
 - Save
 - save master weights gradient (model weights in fp32)
 - scale factor (i.e., 128.0)
 - **BFP16**
 - No need for scaling loss

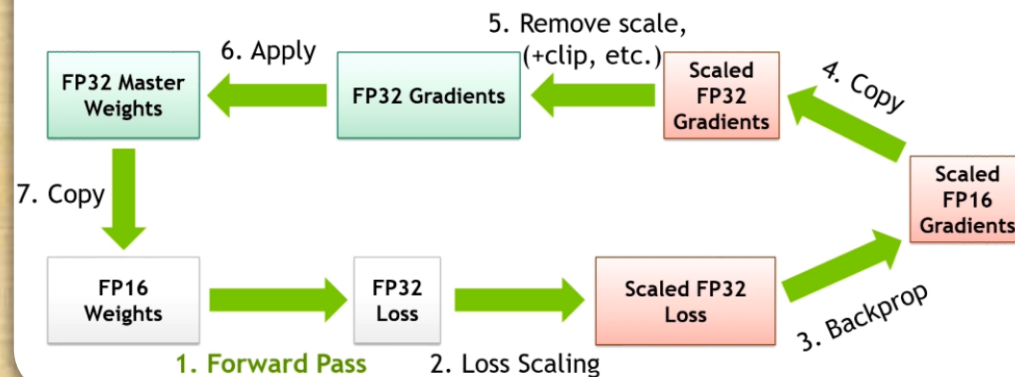


FP32, FP16,
BFP16

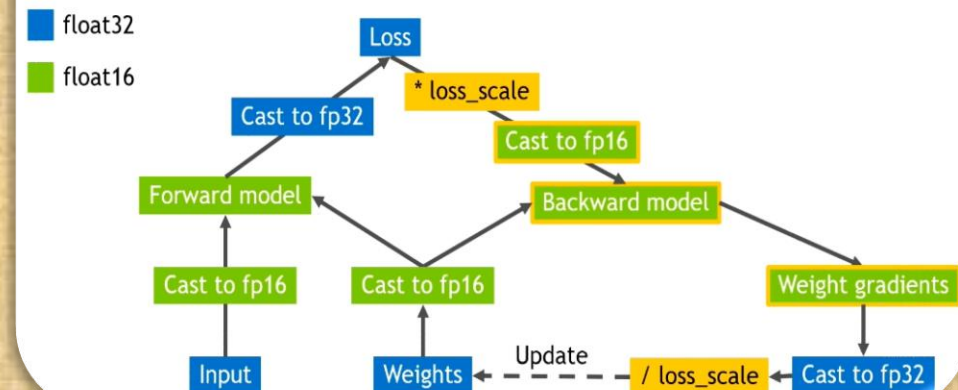
Mixed Precision Process

1. FP16 weights (change only BN, SoftMax layers, ... → from fp16 to fp32)
2. Compute **logits** fp16
3. Convert **logits** fp16 to fp32 → Calculate loss in fp32
4. Scale loss fp32 (*128.0)
5. Compute scaled gradient on fp16
6. Create a copy of scaled gradient from fp16 to fp32
7. Divide scaled gradient (fp32) by scale factor (/128.0)
8. Clip gradient
9. Apply gradient fp32 on fp32 weights (Master weights which is a copy of weights in fp32)
10. Convert fp32 weights to fp16
11. Transfer fp16 weights to the model.

MIXED PRECISION TRAINING



NEW GRAPH



Quantization
(TF32, FP32,
FP16, BFP16)

MLM, NSP

FP32, FP16,
BFP16

Part 2

Masked Language Model (MLM)

Sentence: "AI RoundTable is fantastic."



[CLS] + "AI RoundTable is fantastic." + [SEP], [PAD], ..., [PAD]



101, 9932, 2461, 10880, 2003, 10392, 1012, 102, 0, ..., 0

1, 1, 1, 1, 1, 1, 1, 1, 0, ..., 0

0, 0, 0, 0, 0, 0, 0, 0, 0, ..., 0

input_ids

attention_mask

token_type_ids

MLM, NSP

Masked Language Model (MLM)

Sentence: "AI RoundTable is fantastic."



[CLS] + "AI RoundTable is fantastic." + [SEP], [PAD], ..., [PAD]



101,	9932,	2461,	10880,	2003,	10392,	1012,	102,	0,	...	0
1,	1,	1,	1,	1,	1,	1,	1,	0,	...	0
0,	0,	0,	0,	0,	0,	0,	0,	0,	...	0

~~input_ids~~ labels
attention_mask
token_type_ids



101, 9932, 2461, 10880, **2003**, 10392, 1012, 102, 0, ..., 0

Masked 15%



101, 9932, 2461, 10880, 103, 10392, 1012, 102, 0, ..., 0

input_ids

MLM, NSP

Next Sentence Prediction (NSP)

Sentence 1: "AI RoundTable is fantastic."

Sentence 2: "I recommended AI RoundTable."

Sentence 3: "The weather is sunny."

Sentence_1, Sentence_2

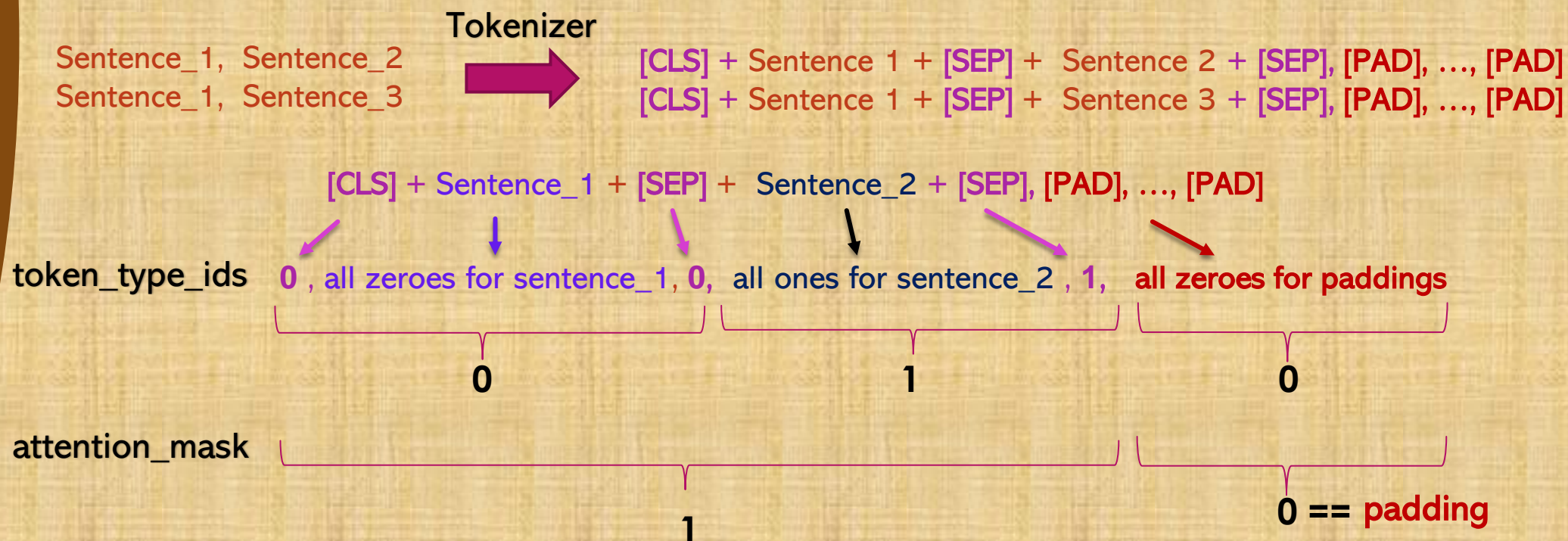
Sentence_1, Sentence_3



next_sentence_label = 0

next_sentence_label = 1

MLM, NSP



MLM, NSP

Code Time

Thanks

Thanks

Any Question ?

