计算平台事业部
COMPUTING PLATFORM

Mixed Precision Training

计算平台事业部PAI团队

Overview

计算平台事业部
COMPUTING PLATFORM

- What is mixed-precision & Why mixed-precision
- How mixed-precision
- Mixed-precision tools on PAI-tensorflow
- Experimental results

What is mixed-precision

- mixed-precision
  - FP32 and FP16
  - *More precision format in the future*

- TensorCore
  - Matrix-multiply and accumulate units
    - FP16 storage/inputs
    - FP32/Fp16 accumulator
  - Such as:
    - Conv
    - MatMul

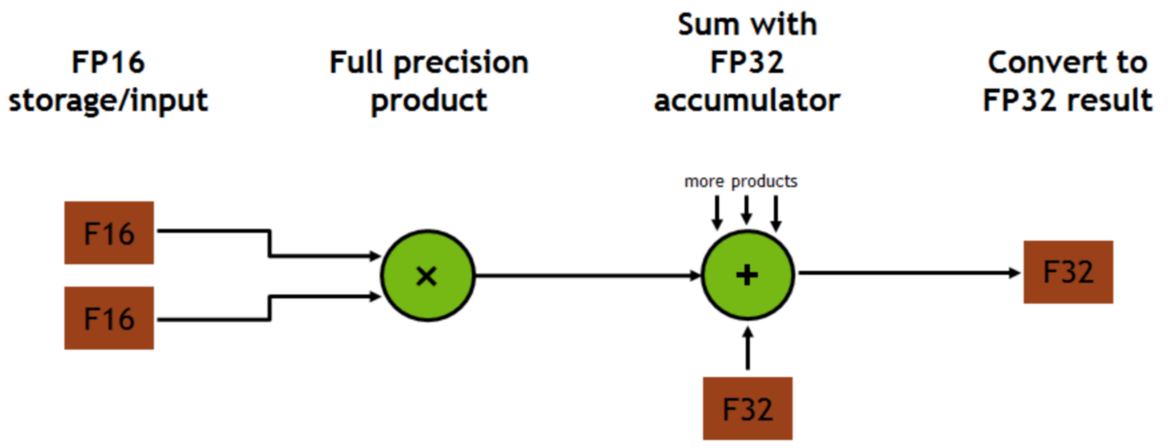


Figure 2: Volta GV100 Tensor Core operation.

Why mixed-precision

计算平台事业部
COMPUTING PLATFORM

- Two key points which matter in training/inference:
  - Computation
    - Tensorcore 8X higher throughput in MP than FP32 (15Tflops v.s. 120Tflops)
  - Memory access
    - Inputs is FP16
    - Memory access is reduced by **2X**

Overview

计算平台事业部
COMPUTING PLATFORM

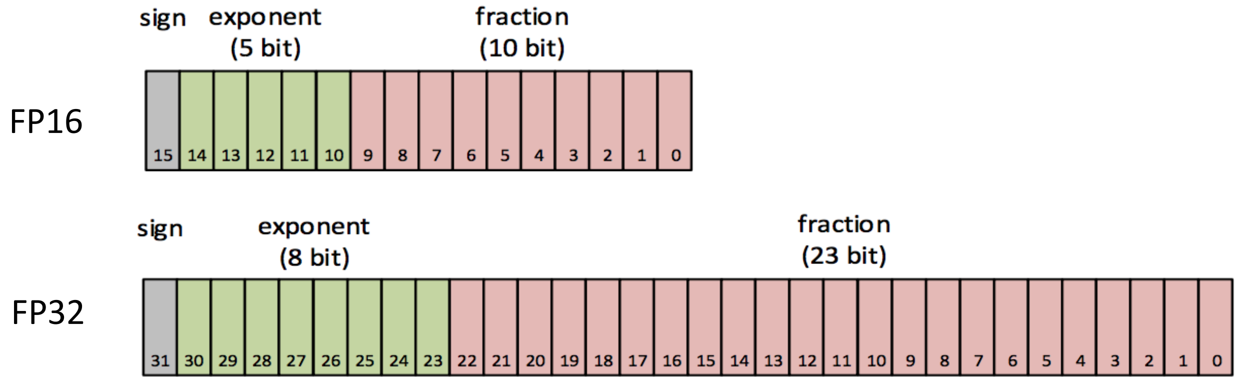
- What is mixed-precision & Why mixed-precision
- How mixed-precision
- Mixed-precision tools on PAI-tensorflow
- Experimental results

How mixed-precision

- Key Strategies in mixed-precision training
  - Issues using FP16 for training and the solutions
    - Less bits in fraction: → Precision gap in sum
    - Less bits in exponent: → Gradients underflow
  - Arithmetic precision design
    - Considering both efficiency and performance

Issues using FP16 for training

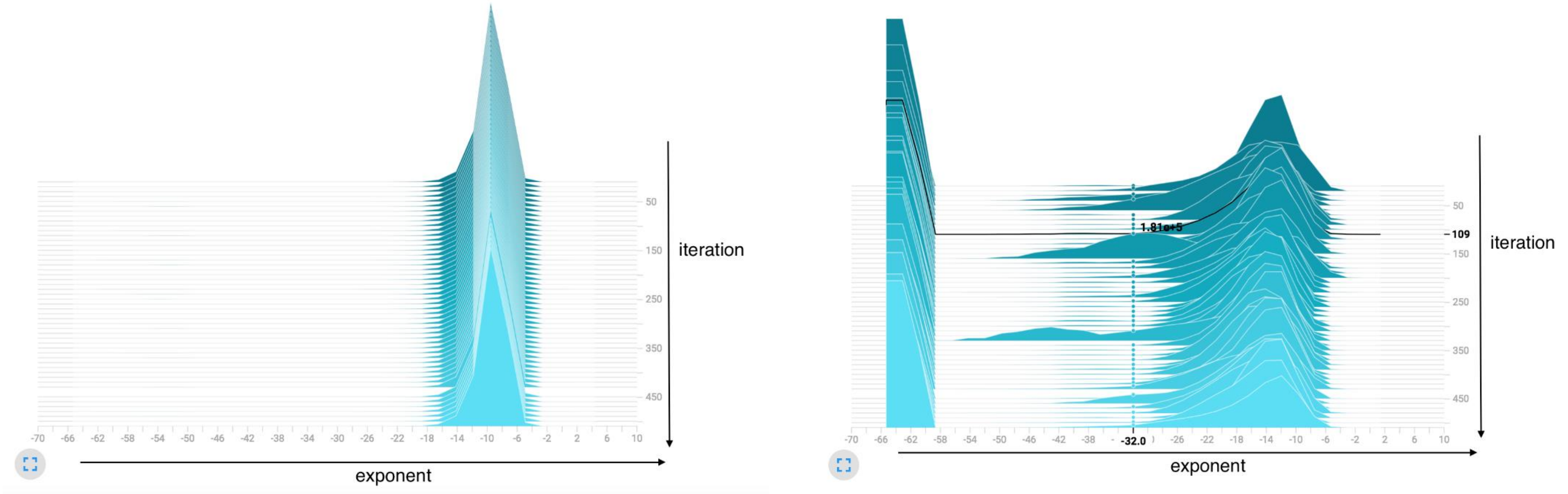
- Less bits in fraction: Precision gap in sum
  - A+B, if A/B>2^{10}, B will degrade to **zero**.
    - For FP32, the ratio can be up to 2^{23}
  - Common in weight update:
    - *W←W+lr*dW*



- Less bits in exponent: Gradients underflow
  - Gradients smaller that 2^{-24} will become zero

Precision gap in sum

- variables v.s. gradients
  - weight update: $W \leftarrow W + lr * dW$ (lr normally in [10^{-1}, 10^{-4}])



Variables: 2^{-16} to 2^{-4}

gradients: 2^{-30} to 2^{-5}

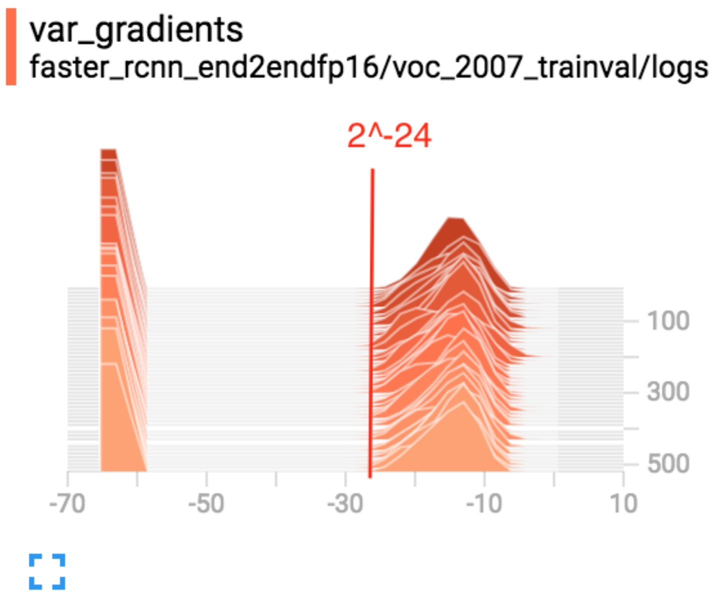
Fig. Variables and gradients histogram in Faster RCNN

- Solution: Variables stored in FP32, and optimizer computation in FP32

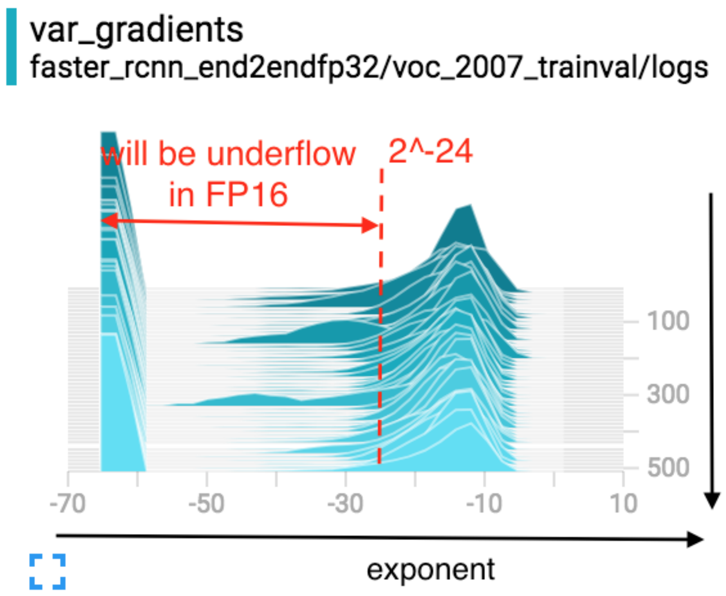
Gradients underflow in FP16

计算平台事业部 COMPUTING PLATFORM

- Gradients of variables



FP16



FP32

Fig. Histogram for gradients of variables in Faster RCNN, respectively training in mixed-precision and FP32

Gradients underflow in FP16

- Gradients of activations

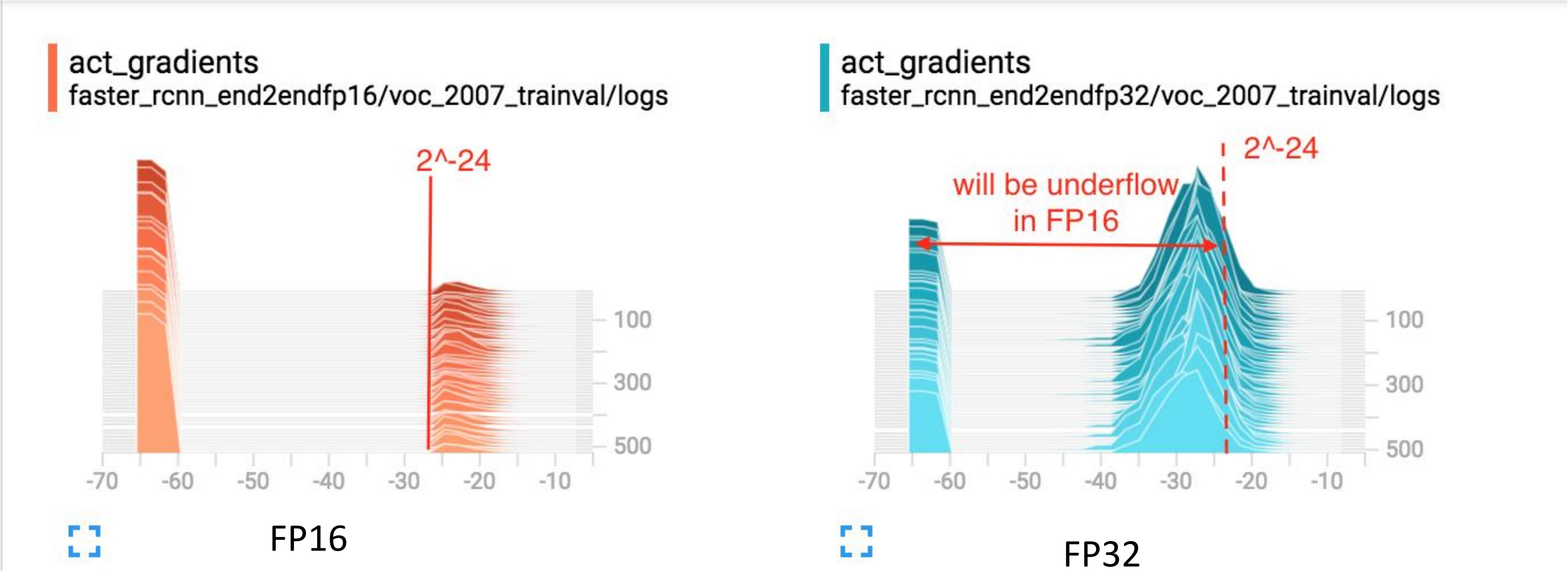
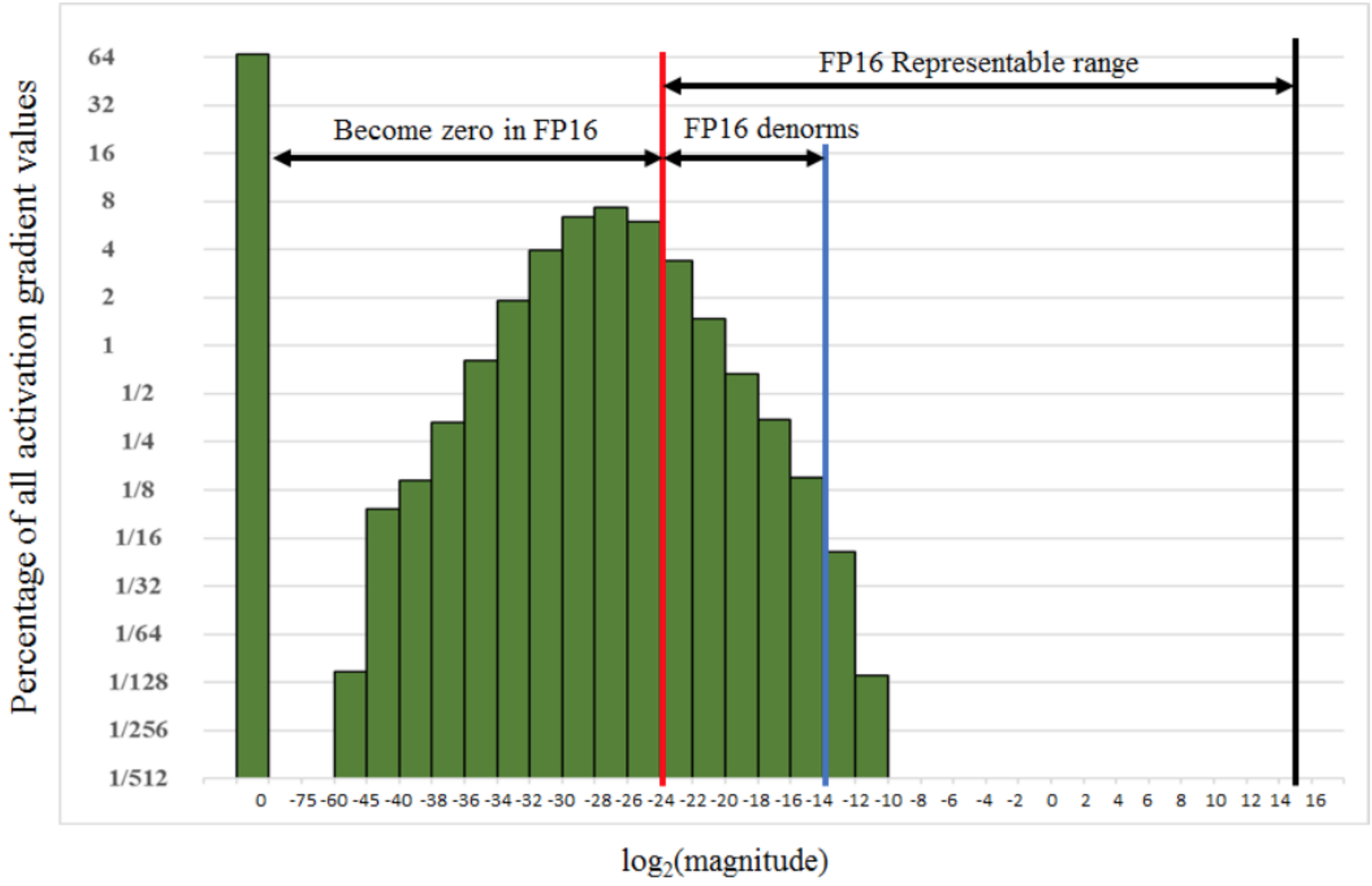


Fig. Histogram for gradients of variables in Faster RCNN, respectively training in mixed-precision and FP32

Gradients underflow in FP16



Solution: gradients shift using loss scaling

Gradients underflow in FP16

- Constant loss scaling
  - Scale the loss by a factor S
  - Backprop to compute the dW
  - Unscale dW by 1/S

- Automatic loss scaling
  - Start with a large scaling factor S
  - For each training iteration:
    - Scale the loss by S
    - Backprop to compute the dW
    - Unscale dW by 1/S
      - If dW contains Inf/NaN, the decrease S by a step factor S/step
      - Otherwise, update dW to W
        - If there is no Inf/NaN for N updates, the increase S by a step factor S*step

How mixed-precision

计算平台事业部
COMPUTING PLATFORM

- Key Strategies in mixed-precision training
  - Issues using FP16 for training and the solutions
    - Less bits in fraction: → Precision gap in sum
      - Solution: Variables stored in FP32, and optimizer computation in FP32
    - Less bit in exponent: → Gradients underflow
      - Solution: loss scaling
  - Arithmetic precision design
    - Considering both efficiency and performance

How mixed-precision

- Key Strategies in mixed-precision training
  - Issues using FP16 for training and the solutions
    - Less bits in fraction: → Precision gap in sum
      - Solution: Variables stored in FP32, and optimizer computation in FP32
    - Less bit in exponent: → Gradients underflow
      - Solution: loss scaling
  - Arithmetic precision design
    - Considering both efficiency and performance

Arithmetic precision design

- Arithmetic can be categorized into:
  1. Compute-bound
     - Convolution, Matmul

Take advantage of Tensorcore:

- Inputs: FP16
- Accumulator: FP32
- Outputs: FP32

  2. Memory-bound

     ① Reductions
     - Batch-norm/layer-norm/group-norm
     - Softmax / Average pooling

- Inputs/Ouputs in FP16
- Computation in FP32

     ② Element-wise operation
     - Add/mul, etc

- Inputs/Ouputs in FP16
- Computation in FP16
  - Computation in FP32

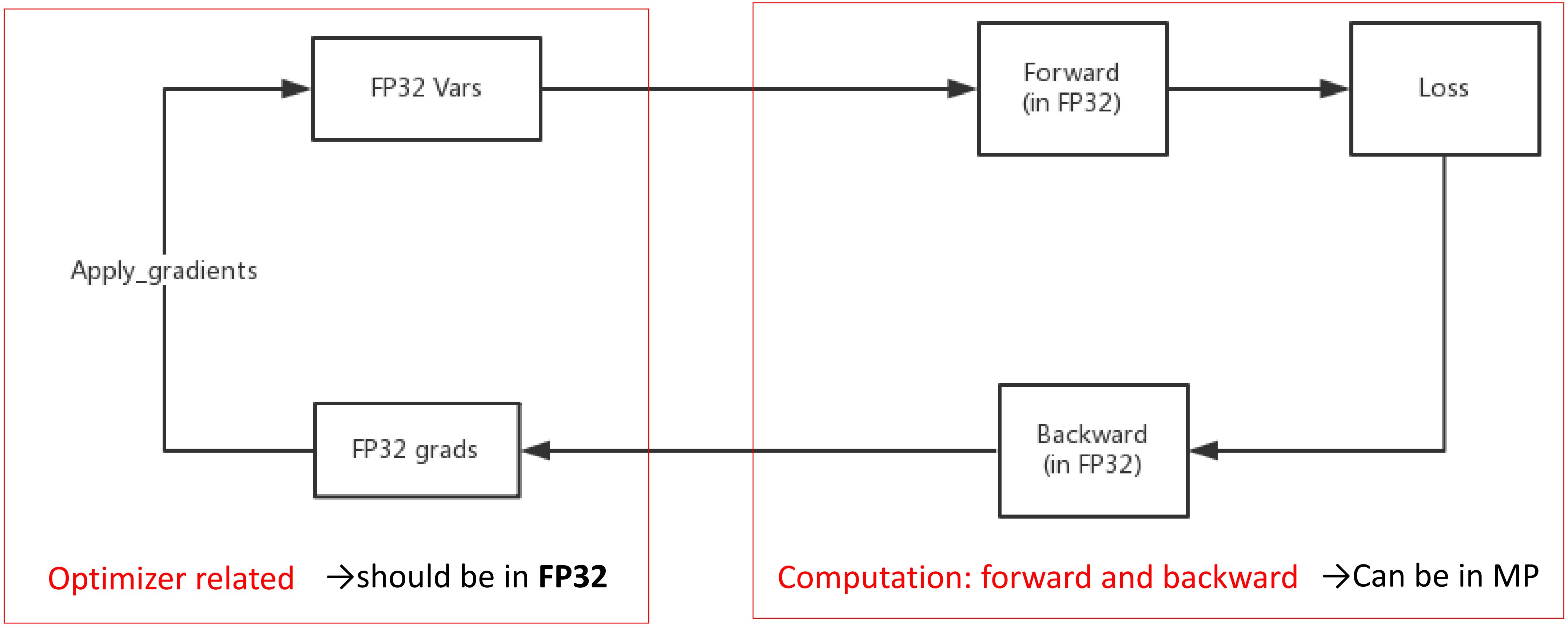
Arithmetic precision design

计算平台事业部
COMPUTING PLATFORM

- Compute-bound operations:
  - Inputs in FP16
  - Computation using Tensorcore
  - Outputs in FP32
- Memory-bound operations:
  - Inputs/outputs in FP16
  - Computation in FP32

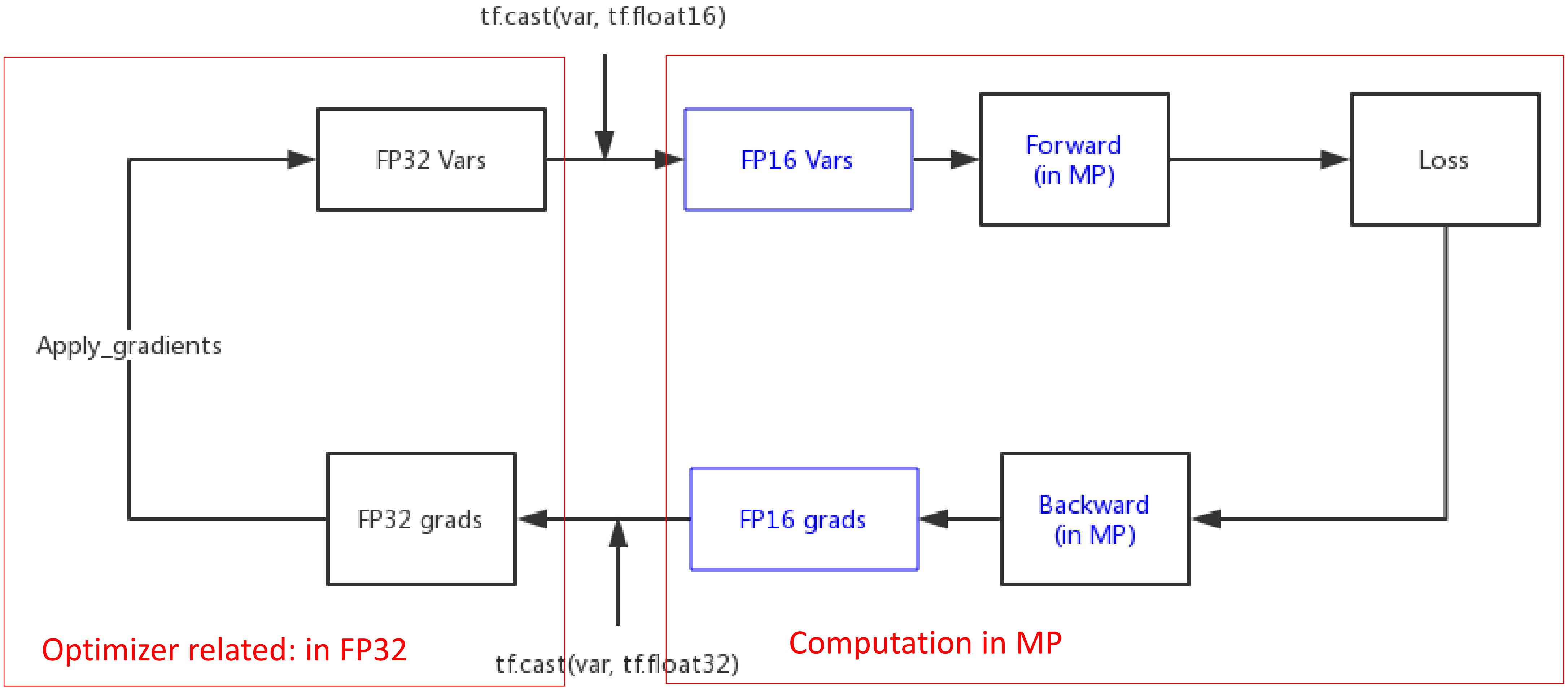
How mixed-precision training

FP32 training baseline



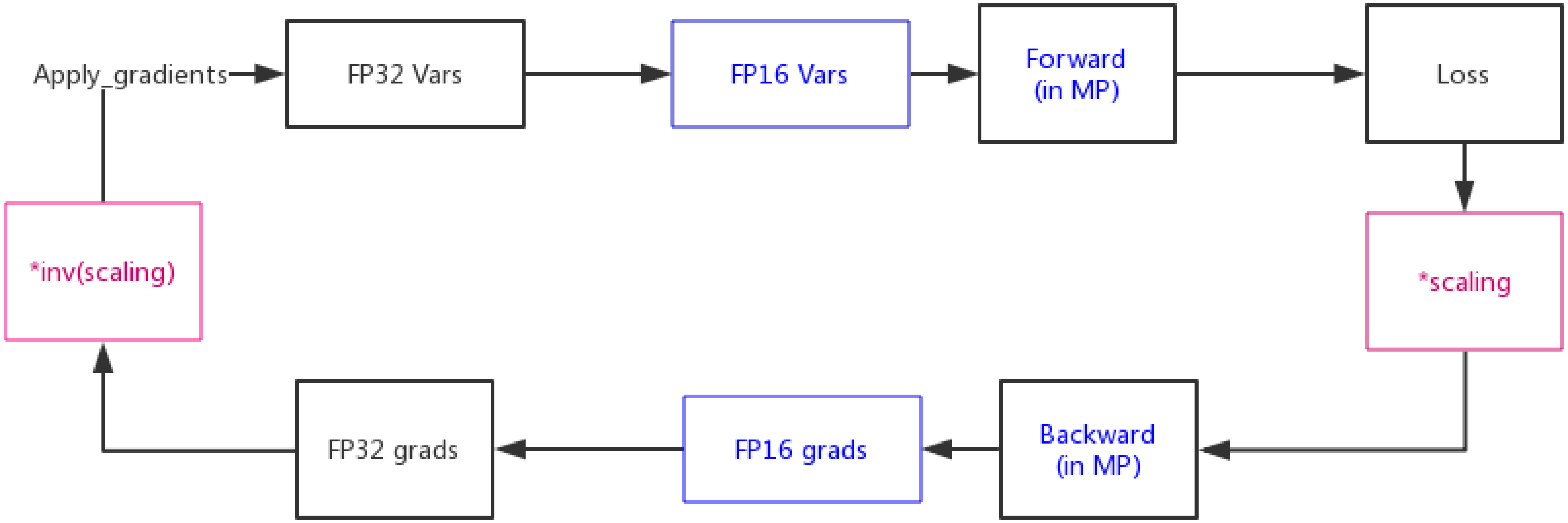
MP training (var in FP32):

- Convert the computation part to MP
- Remain the optimizer part in FP32



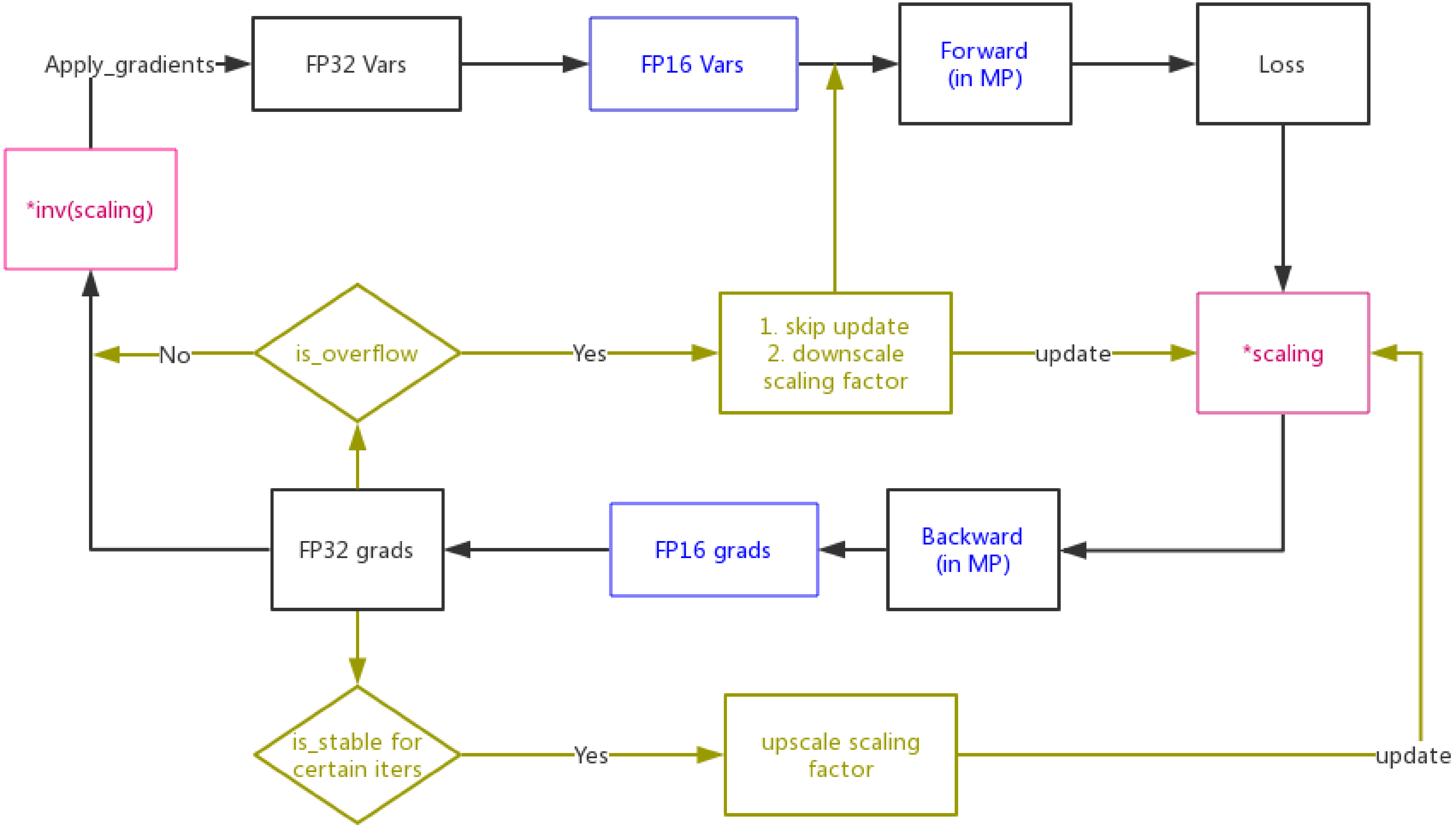
MP training (var in FP32):

- Loss Scaling strategy (*constant scaling*)



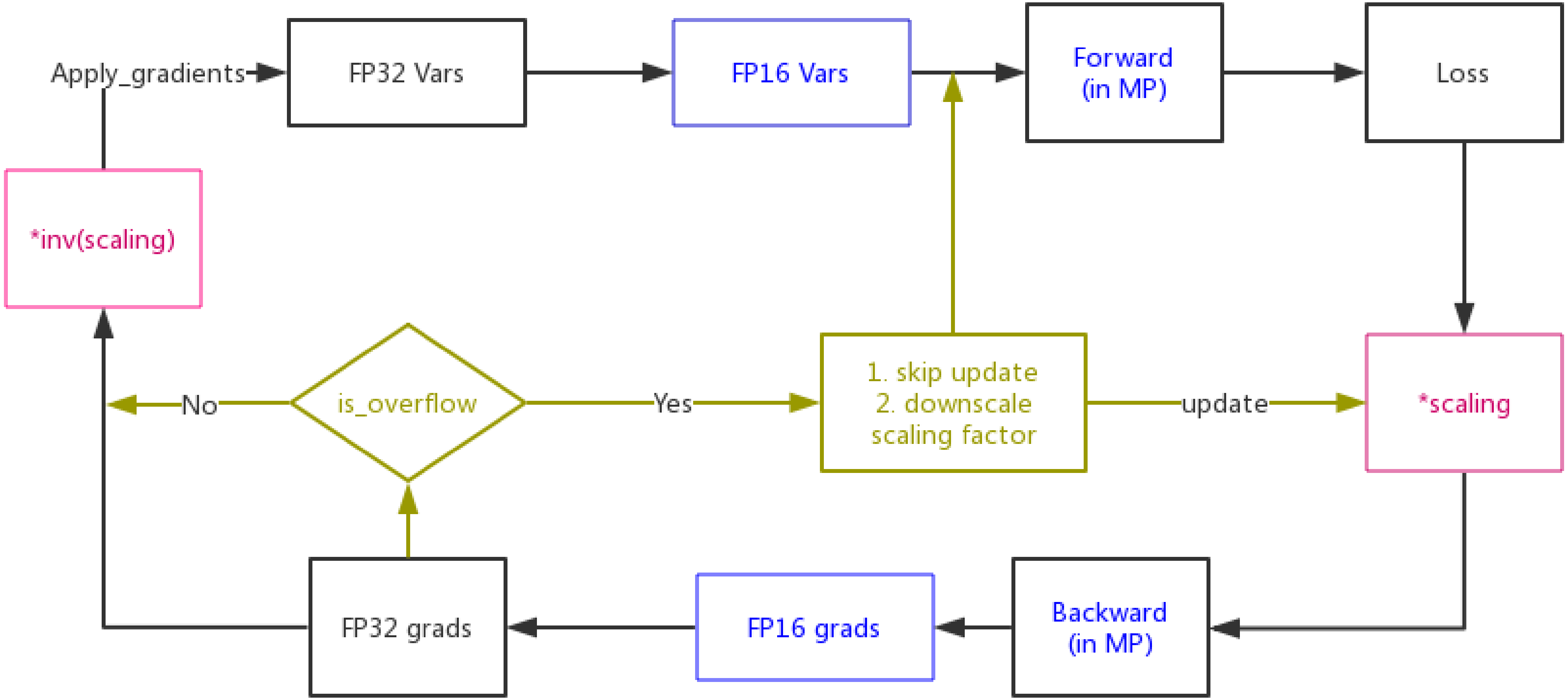
MP training (var in FP32):

- Auto Loss Scaling strategy



MP training (var in FP32):

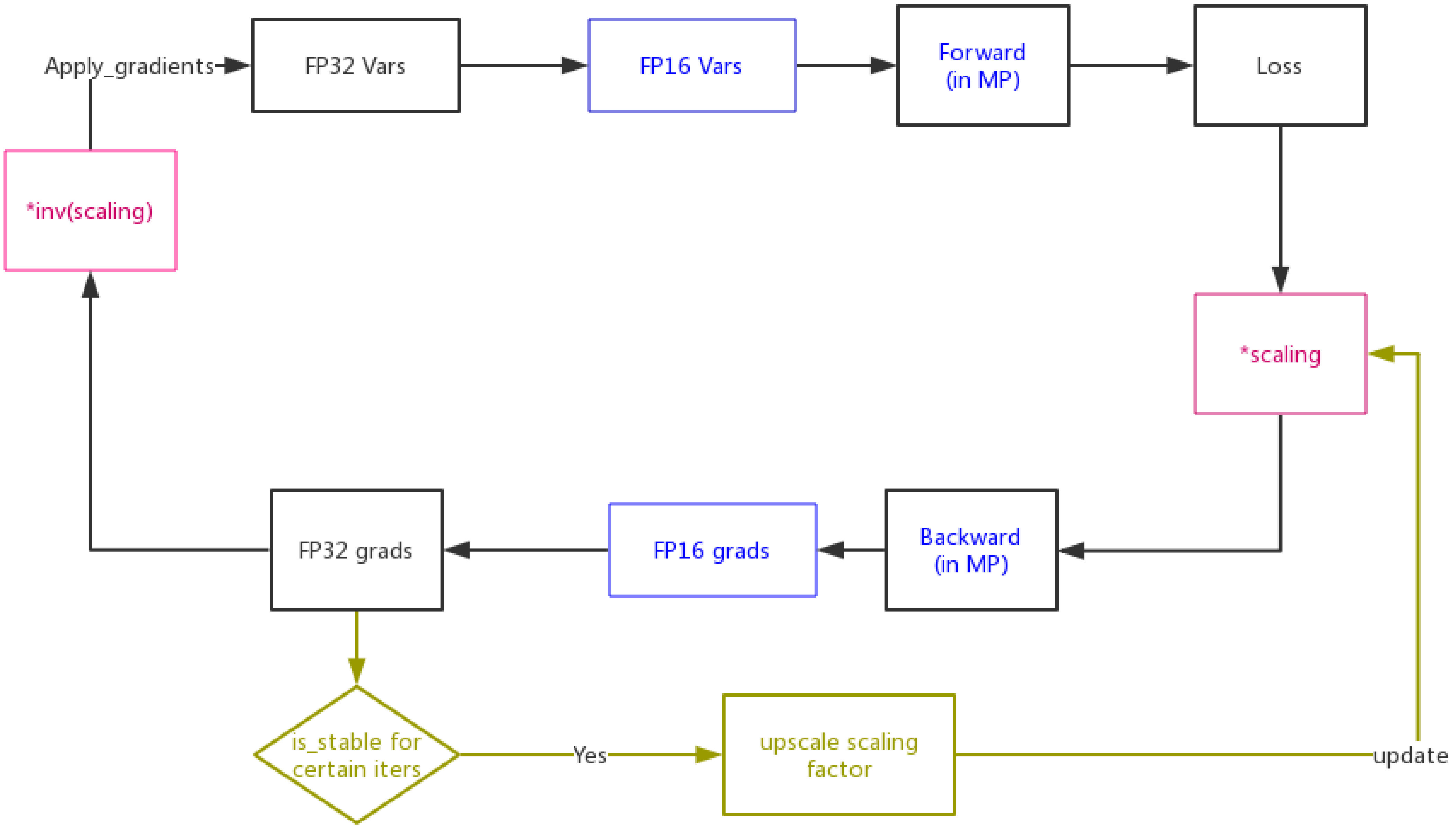
- Auto Loss Scaling strategy



MP training (var in FP32):

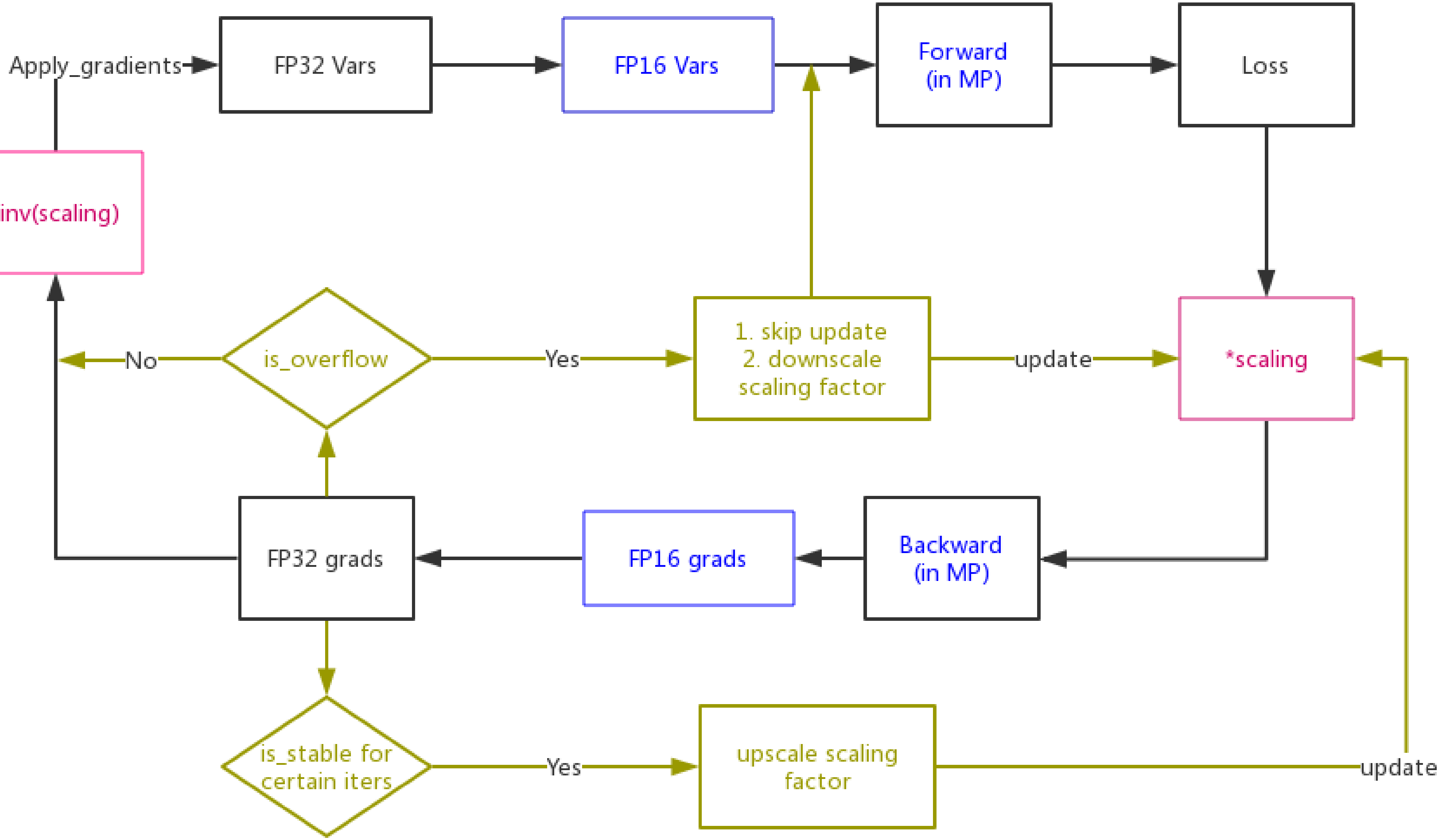
- Auto Loss Scaling strategy

计算平台事业部 COMPUTING PLATFORM



MP training (var in FP32):

- Auto Loss Scaling strategy



Overview

计算平台事业部
COMPUTING PLATFORM

- What is mixed-precision & Why mixed-precision
- How mixed-precision
- Mixed-precision tools on PAI-tensorflow
- Experimental results

MP training tools on PAI-TF

计算平台事业部
COMPUTING PLATFORM

- Graph Optimization + Loss Scaling Training Strategy
  - Graph Optimization：AutoMixedPrecision Graph Optimization Pass

FP32 graph_def $\xrightarrow{\text{Automatically conversion}}$ MP graph_def

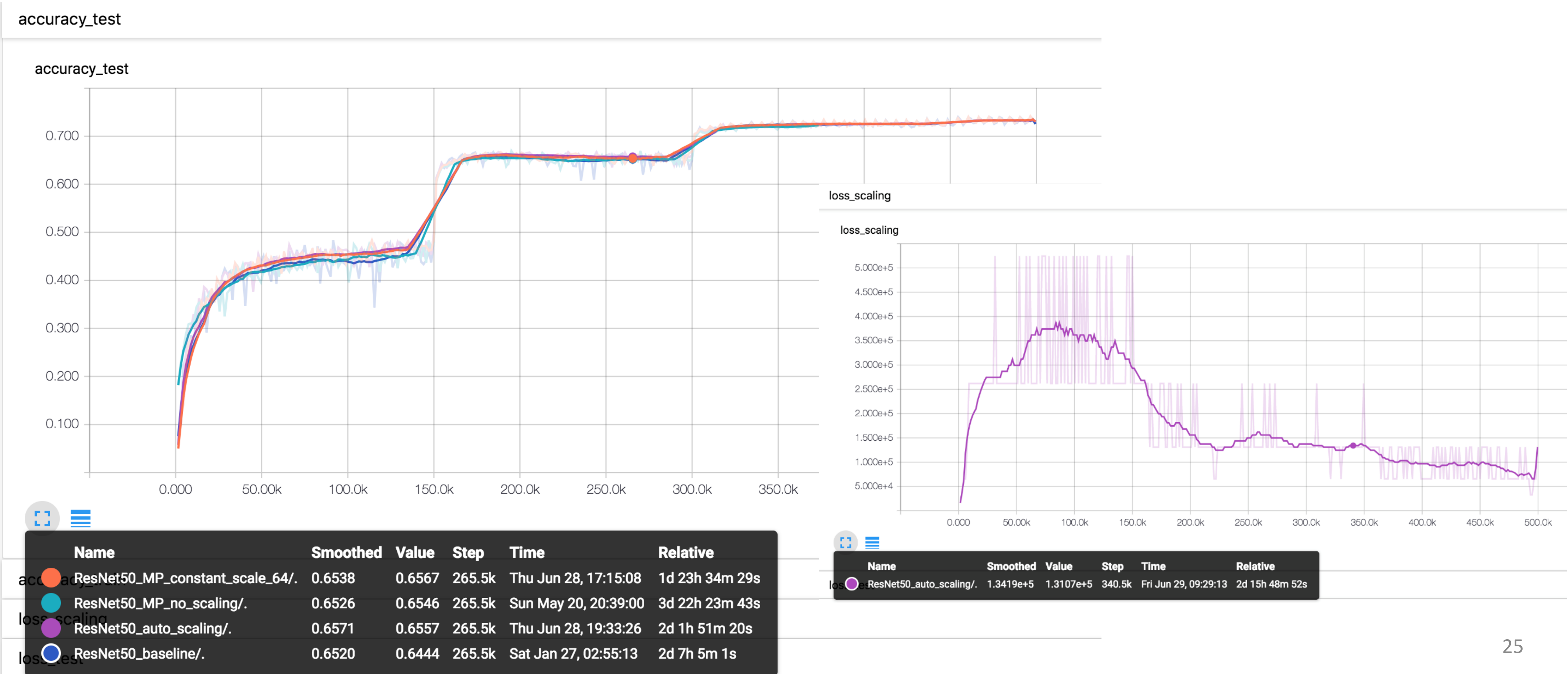
  - MP Training Strategy: MP optimizer wrapper
    - Wrap the standard optimizers to automatically adopt the constant/automatic loss scaling strategy
      - *opt = tf.contrib.mixed_precision.MixedPrecisionOptimizer(opt)*
    - Both constant/automatic loss scaling supported

Mixed-precision optimizer

Standard optimizer

Experimental results

- ResNet50 on ImageNet



Experimental results

计算平台事业部
COMPUTING PLATFORM

- Faster RCNN (VGG backbone) on PASCAL VOC 07

	mAP	mAP Diff
FP32(baseline)	69.48	-
FP16(no loss scaling)	67.83	-1.65
FP16(constant scaling=8)	69.71	+0.23
FP16(constant scaling=64)	69.96	+0.48
FP16(constant scaling=256)	69.78	+0.30
FP16(constant scaling=4096)	69.74	+0.26
FP16(auto scaling)	69.68	+0.20

Experimental results

- SSD (VGG backbone) on PASCAL VOC 07+12

训练精度	loss scaling策略	mAP
FP32	NO	77.46 (baseline)
Mixed precision	NO	diverge
	constant (=64)	77.63
	auto scaling	77.42

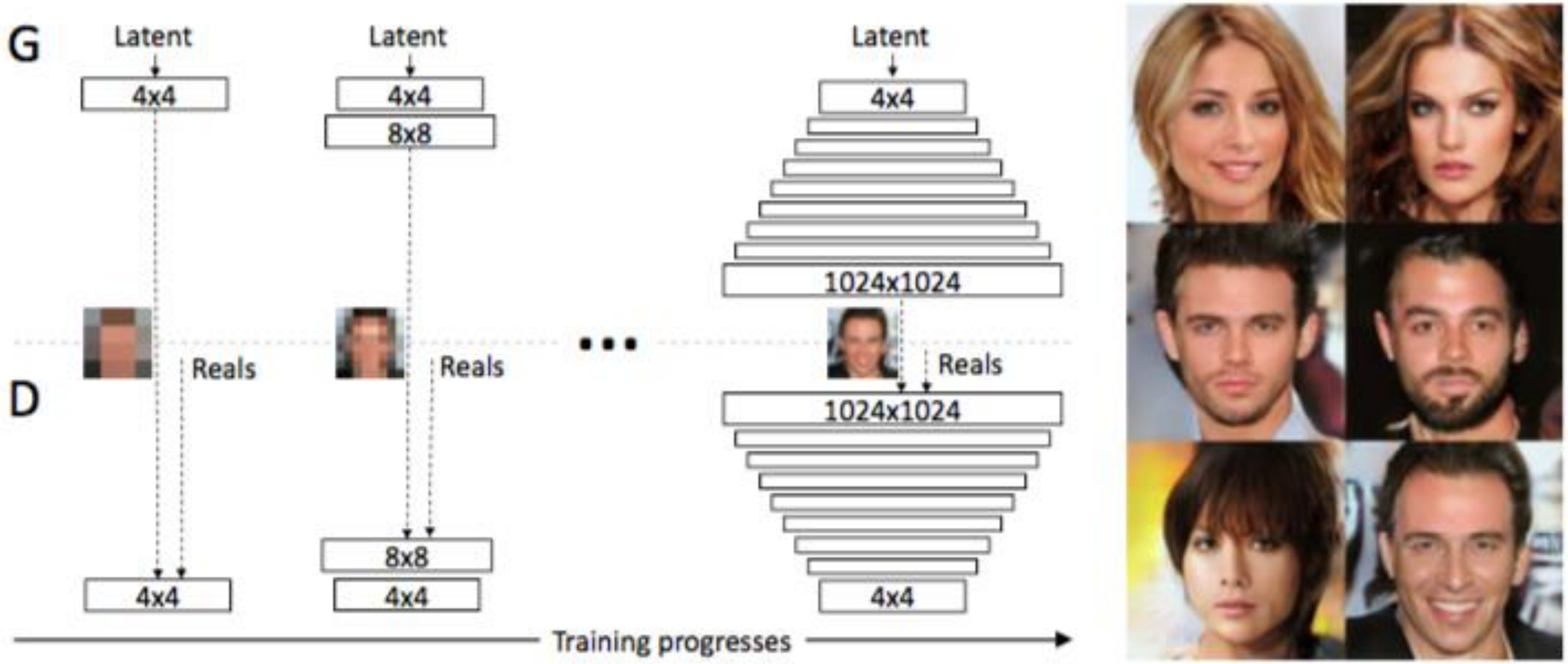
Experimental results

- Small NMT on **WMT German-English**
  - Encoder: 2 layers
  - Decoder: 2 layers with attention

训练精度	Scaling策略	BLEU
FP32	NO	23.68
Mixed-precision	Auto scaling	23.79

PGAN

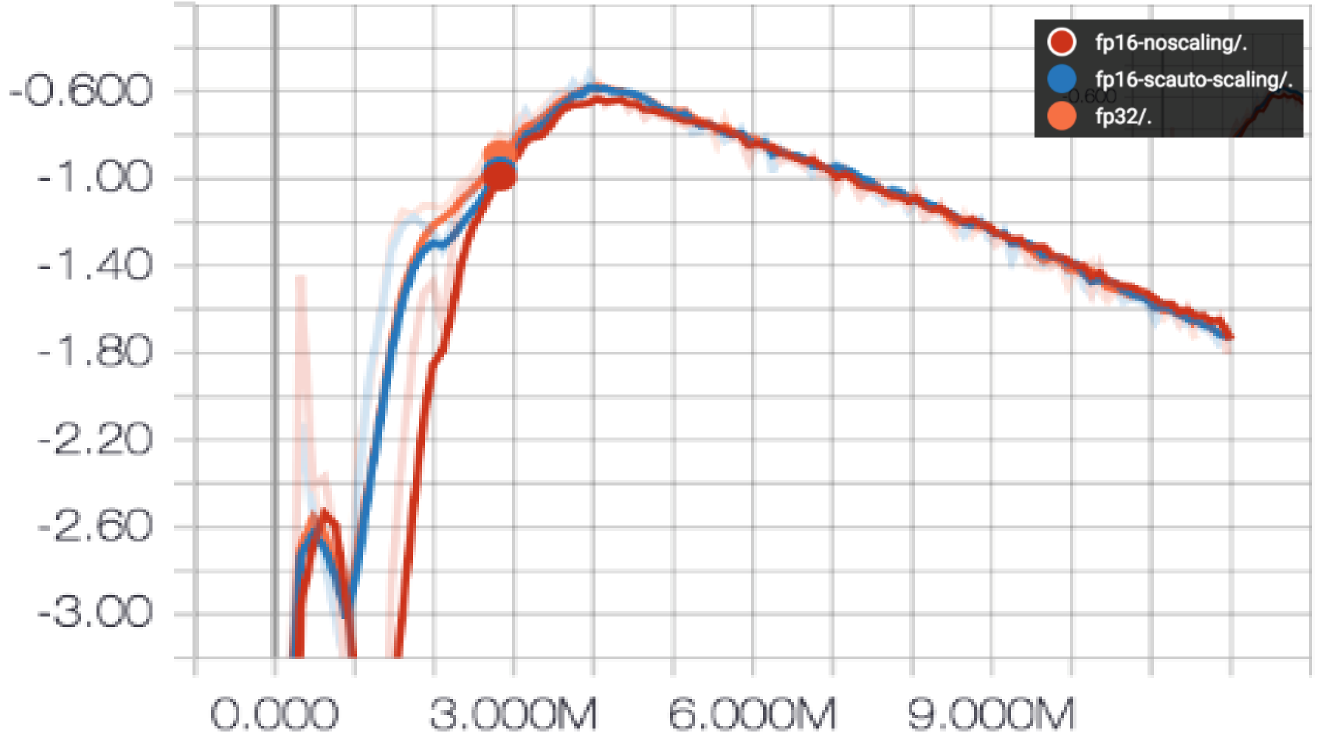
- PGAN (Progressive growth of GAN)



Karras, Tero, et al. "Progressive Growing of GANs for Improved Quality." *Stability, and Variation.*

PGAN

- G loss



Karras, Tero, et al. "Progressive Growing of GANs for Improved Quality." *Stability, and Variation.*

PGAN

计算平台事业部
COMPUTING PLATFORM

- Generation results (cifar10 dataset)

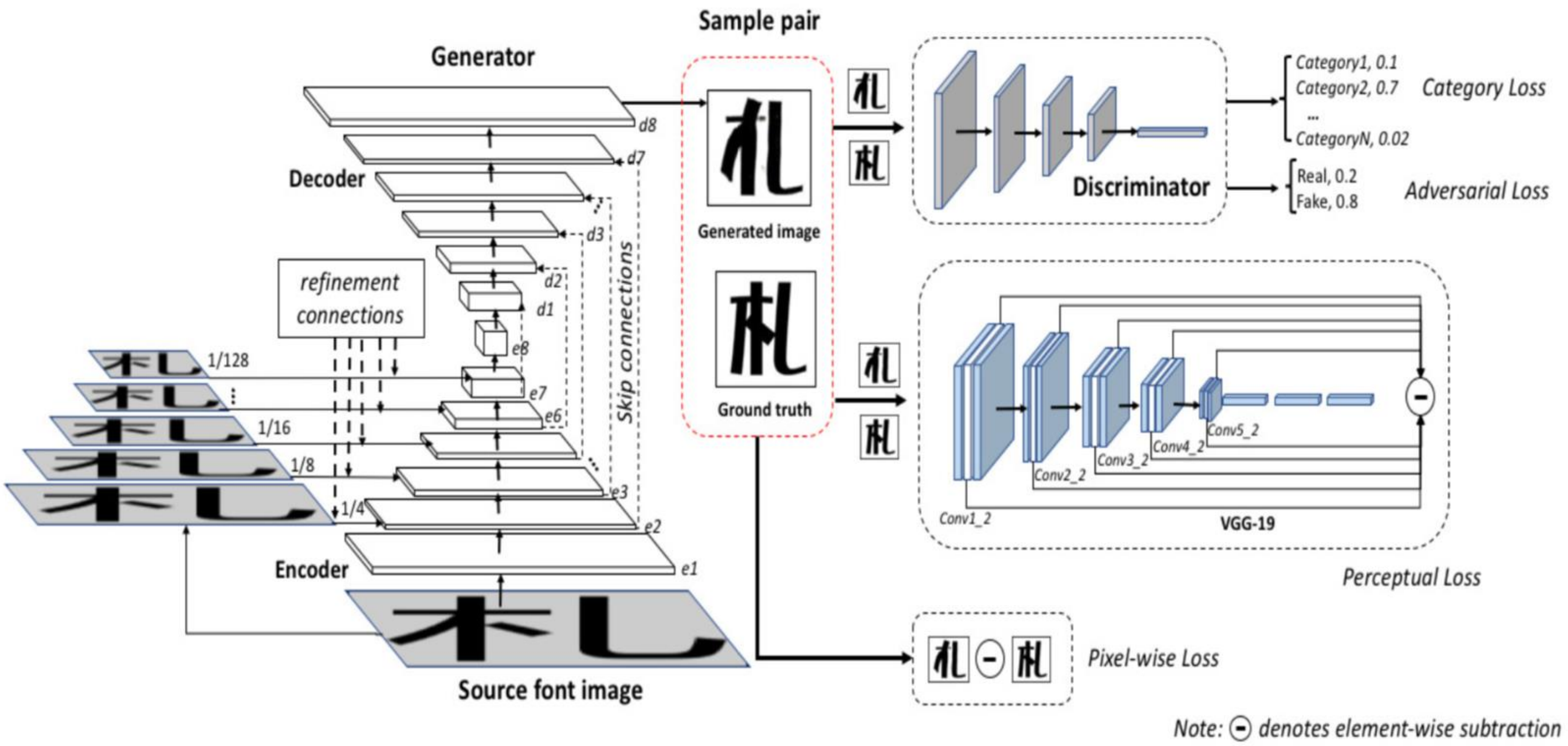
fp32

mp-auto-scaling

mp-no-scaling

Exp.	fp32	mp-auto-scaling	mp-no-scaling
sliced_wasserstein	9.3764	9.1662	7.9601

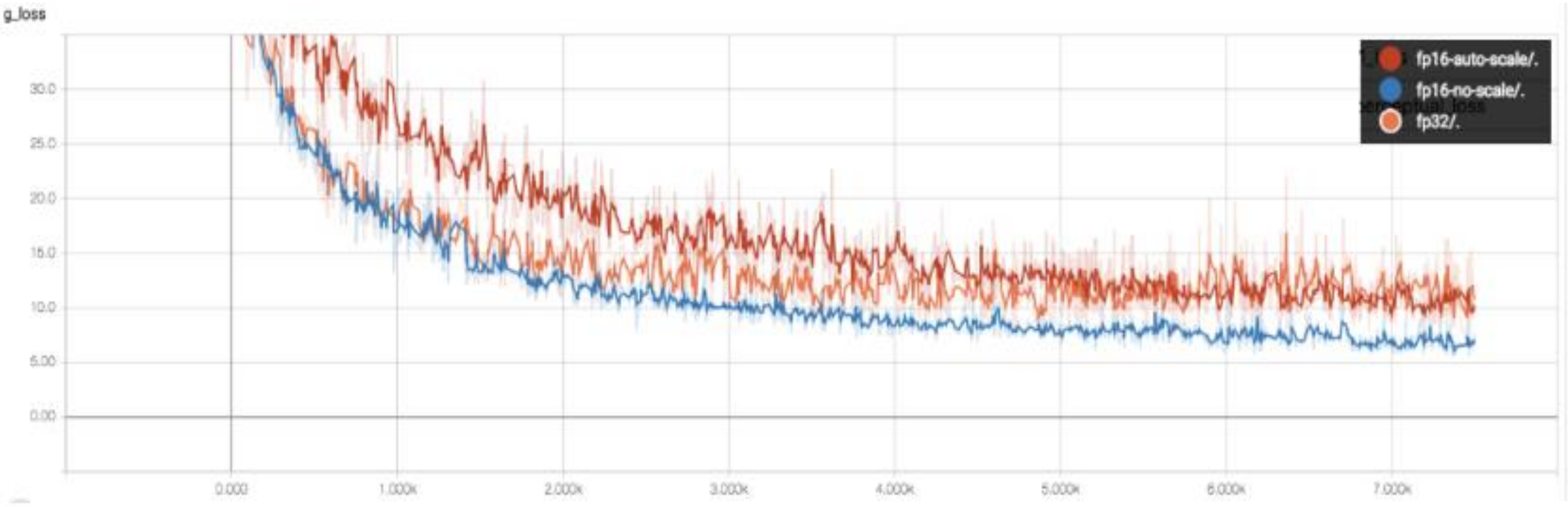
Font Generation



Pyramid Embedded Generative Adversarial Network for Automated Font Generation

Font Generation

- G loss



Pyramid Embedded Generative Adversarial Network for Automated Font Generation

Font Generation

- Generation results (金陵刻经体)

fp32 mp-no-scaling mp-auto-scaling

Wide & Deep Learning

- Predict the probability that the individual has an annual income of over 50,000 dollars

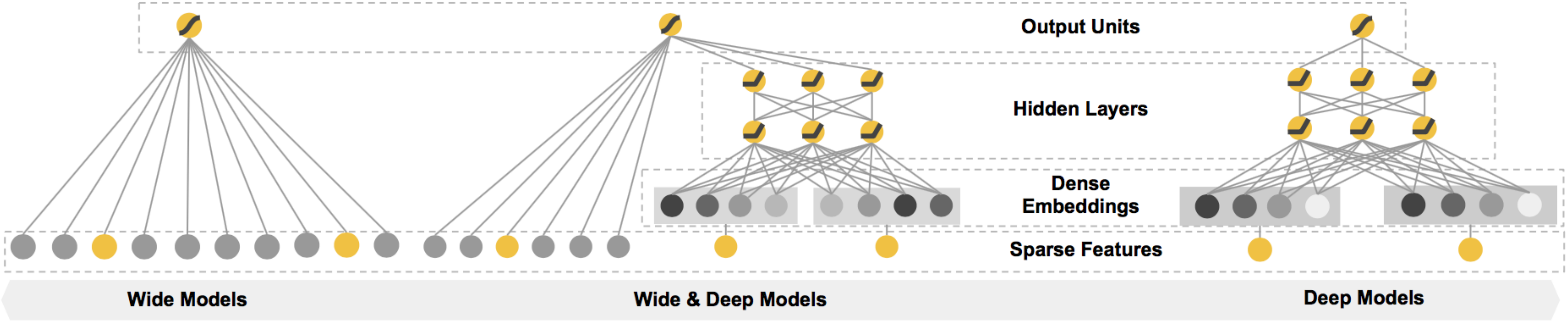


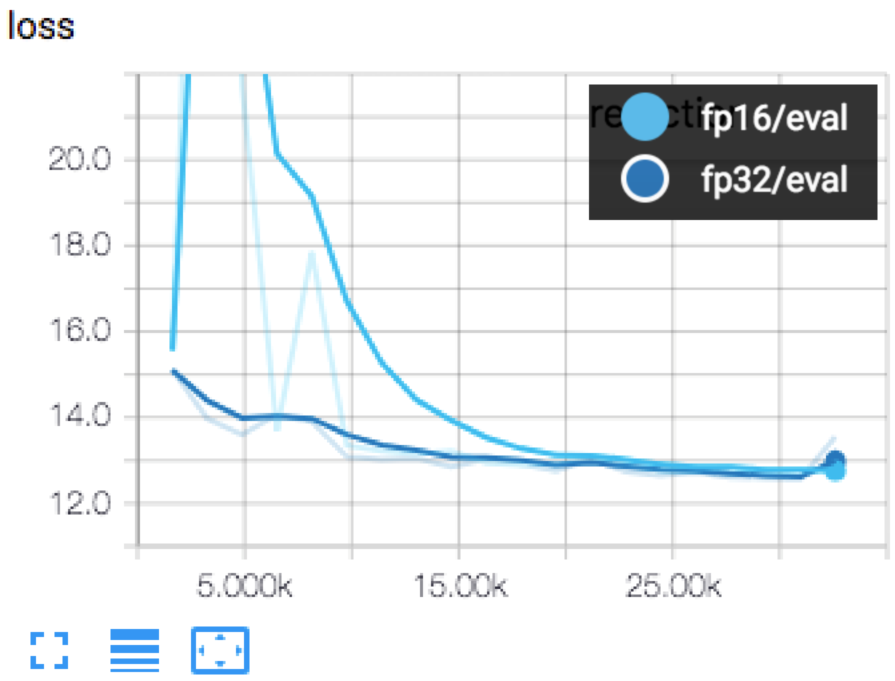
Figure 1: The spectrum of Wide & Deep models.

Wide & Deep Learning for Recommender Systems

Wide & Deep Learning

计算平台事业部
COMPUTING PLATFORM

- Loss



Exp	fp32	mp-no-scaling
Accuracy	84.31%	84.27%

Wide & Deep Learning for Recommender Systems

More try: small inputs (for normalization layers)

- Underflow in FP16 gradients
  - Design the model to be more adaptive to FP16 representation
  - Move the gradient itself into the FP16 representable range
    - Especially the activation gradients
- Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

- **BN** will stand for Batch Norm.
- f represents a layer upwards of the BN one.
- y is the linear transformation which scales x by γ and adds β.
- $\hat{x}$ is the normalized inputs.
- μ is the batch mean.
- σ^2 is the batch variance.

Small input

- Derivatives of BN layer

$$\frac{\partial f}{\partial x_i} = \frac{m\frac{\partial f}{\partial \hat{x}_i} - \sum_{j=1}^{m}\frac{\partial f}{\partial \hat{x}_j} - \hat{x}_i \sum_{j=1}^{m}\frac{\partial f}{\partial \hat{x}_j}\cdot \hat{x}_j}{m\sqrt{\sigma^2+\epsilon}}$$

↓

$$\frac{\partial f}{\partial x_i} \propto \mathcal{O}\left(1/\sigma\right)$$

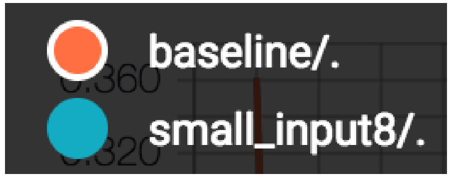
↓

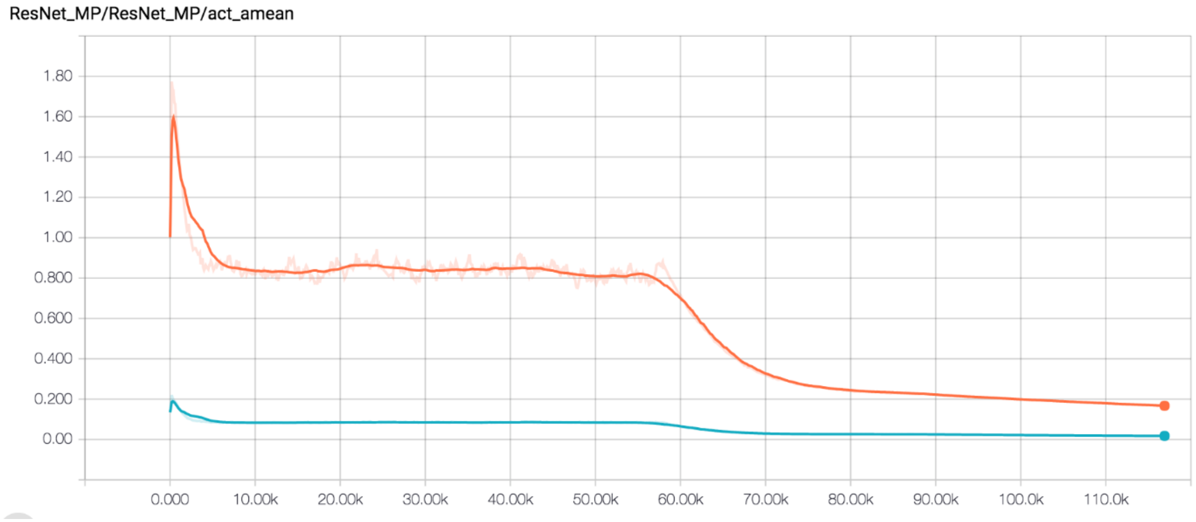
Smaller Inputs and Bigger derivatives

- Reduce the magnitude of the inputs
  - Reduce magnitude of the forward activations, so as to reduce the overflow in forward propagation when using FP16
  - Improve the magnitude of the derivatives
- Tips for Network with BN:
  - Normalize the layer to have std to be 1/S rather than 1.0

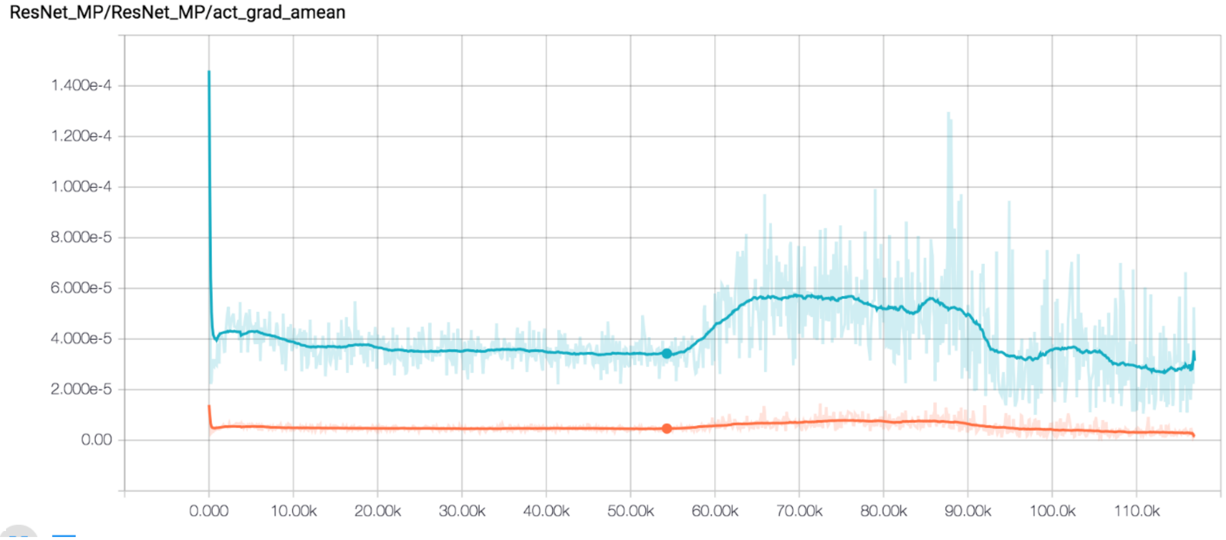
Small inputs

- ResNet32+CIFAR10
  - Activations and the gradients





activations



activation gradients

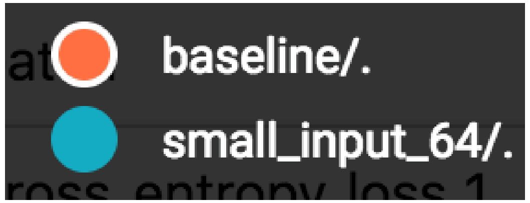
Small inputs

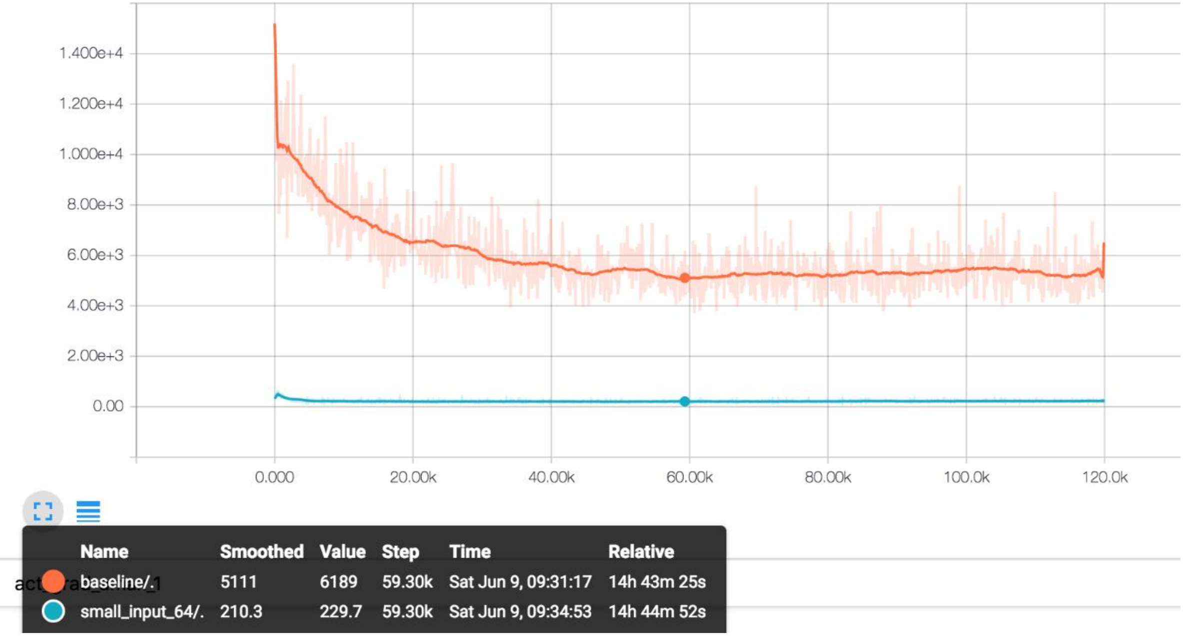
- ResNet32+CIFAR10
  - All without loss scaling

训练精度	Input scale	Accuracy
FP32	1.0	93.58
Mixed-precision	1.0	93.41
Mixed-precision	1/8.0	93.43

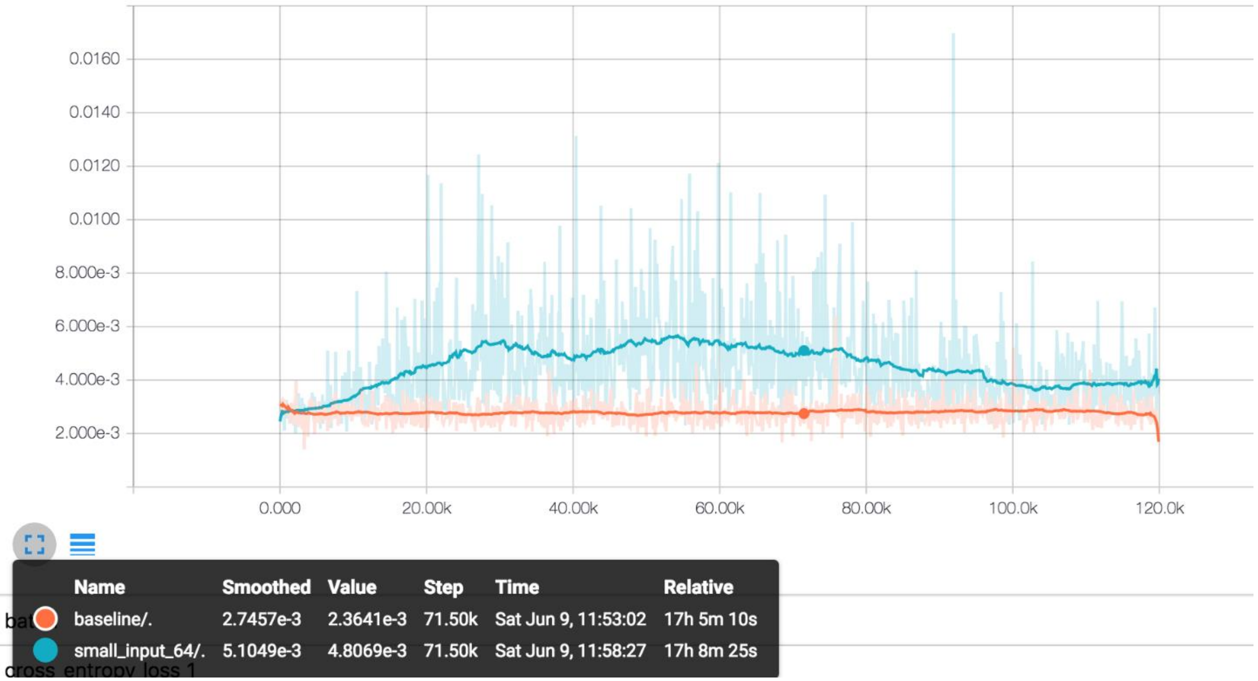
Small inputs

- SSD on PASCAL VOC 07+12
  - Activations and the gradients





activations



gradients

Small inputs

- SSD on PASCAL VOC 07+12
  - Activations and the gradients

训练精度	loss scaling策略	mAP (standard input)	mAP (small input, scaled 1/64)
FP32	NO	77.46 (baseline)	77.25
Mixed precision	NO	**diverge**	77.75
	constant (=64)	77.63	77.56
	auto scaling	77.42	77.17

Conclusion

- Mixed-precision tools have been supported on PAI-tensorflow
- More effort is still conducted to explore more in mixed-precision
  - More precision supported
  - More training strategy

Thank you