MIXED PRECISION TRAINING FOR VIDEO SYNTHESIS

Ting-Chun Wang

VIDEO SYNTHESIS

- vid2vid [Wang et al. 2018]

- Few-shot vid2vid [Wang et al. 2019]

VIDEO SYNTHESIS

- vid2vid [Wang et al. 2018]

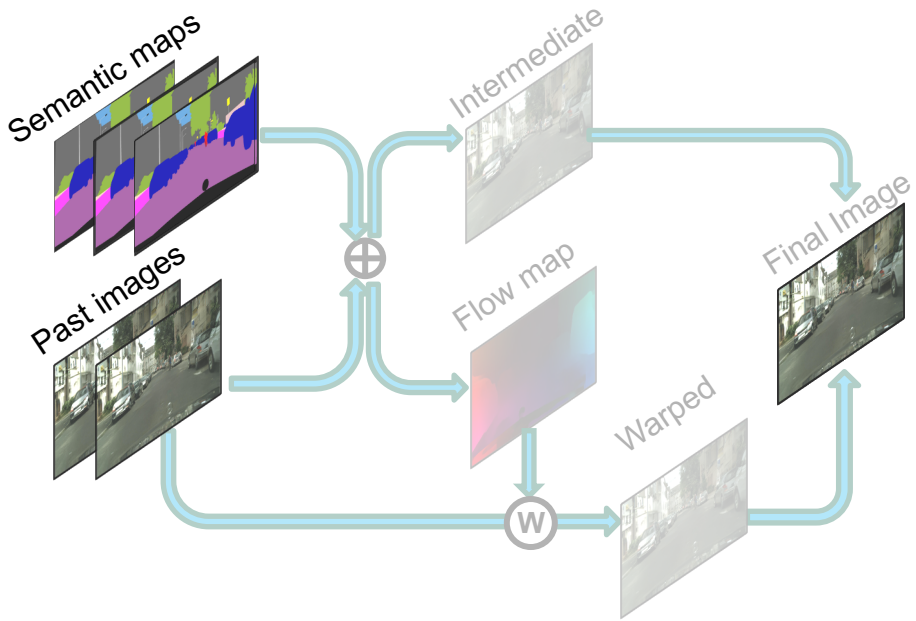
- Few-shot vid2vid [Wang et al. 2019]

APPLICATION

T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, B. Catanzaro, “Video-to-Video Synthesis,” in NeurIPS 2018.
https://github.com/NVIDIA/vid2vid

VID2VID: MODEL OVERVIEW

Generating frame at time *t+1*



TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- PyTorch JIT
- Fused Adam Optimizer
- Thread affinity binding
- Gradient checkpointing

TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- PyTorch JIT
  - Fuse multiple pointwise operations (elementwise addition, multiplication, sigmoid) into a single CUDA kernel

before

```
def gelu(x):
    return x * 0.5 * (1.0 + torch.erf(x / 1.41421))
```

after

```
@torch.jit.script
def fused_gelu(x):
    return x * 0.5 * (1.0 + torch.erf(x / 1.41421))
```

Function name	Number of launched CUDA kernels	Execution time (for input vector with 1M fp32 elements)
`gelu(x)`	5	79 us
`fused_gelu(x)`	1	18 us

TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- PyTorch JIT
  - Fuse multiple pointwise operations (elementwise addition, multiplication, sigmoid) into a single CUDA kernel

before

```
class GANLoss(nn.Module):
    def __init__(self, gan_mode, target_real_label=1.0, target_fake_label=0.0):
@@ -92,11 +104,13 @@ class GANLoss(nn.Module):
            assert reduce_dim is True
            if dis_update:
                if t_real:
-                   minval = torch.min(input_x - 1, input_x * 0)
-                   loss = -torch.mean(minval)

                else:
-                   minval = torch.min(-input_x - 1, input_x * 0)
-                   loss = -torch.mean(minval)

            else:
                assert t_real, "Hinge losses must be aiming for real."
                loss = -torch.mean(input_x)
```

after

```
+ @torch.jit.script
+ def fuse_math_min_mean1(input):
+     minval = torch.min(input - 1, input * 0)
+     loss = -torch.mean(minval)
+     return loss
+
+ @torch.jit.script
+ def fuse_math_min_mean2(input):
+     minval = torch.min(-input - 1, input * 0)
+     loss = -torch.mean(minval)
+     return loss
+

class GANLoss(nn.Module):
    def __init__(self, gan_mode, target_real_label=1.0, target_fake_label=0.0):
@@ -92,11 +104,13 @@ class GANLoss(nn.Module):
            assert reduce_dim is True
            if dis_update:
                if t_real:
+                   loss = fuse_math_min_mean1(input_x)
                else:
+                   loss = fuse_math_min_mean2(input_x)
```

TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- PyTorch JIT
  - Fuse multiple pointwise operations (elementwise addition, multiplication, sigmoid) into a single CUDA kernel

before

```
class SpectralNorm(object):
    """
@@ -87,20 +92,26 @@ class SpectralNorm(object):
                    # right singular vectors.
                    # This power iteration produces approximations o
f `u` and
                    # `v`.
-                   v = normalize(
-                       torch.mv(
-                           weight_mat.t(),
-                           u),
-                       dim=0,
-                       eps=self.eps,
-                       out=v)
-                   u = normalize(
-                       torch.mv(
-                           weight_mat,
-                           v),
-                       dim=0,
-                       eps=self.eps,
-                       out=u)
```

after

```
+ @torch.jit.script
+ def fuse_norm(input: torch.Tensor, eps: float, out: torch.Tensor):
+     out = normalize(input, dim=0, eps=eps, out=out)
+     return out
+
  class SpectralNorm(object):
      """
  @@ -87,20 +92,26 @@ class SpectralNorm(object):
                     # right singular vectors.
                     # This power iteration produces approximations o
  f `u` and
                     # `v`.
+                    v = fuse_norm(
+                        torch.mv(weight_mat.t(), u),
+                        self.eps, v)
+                    u = fuse_norm(
+                        torch.mv(weight_mat, v),
+                        self.eps, u)
```

TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- PyTorch JIT
  - Fuse multiple pointwise operations (elementwise addition, multiplication, sigmoid) into a single CUDA kernel
  - For pytorch == 1.5, initialize jit with

```
torch._C._jit_set_profiling_executor(False)
torch._C._jit_set_profiling_mode(False)
```

TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- PyTorch JIT
- Fused Adam Optimizer
  - Adam updates params ***sequentially*** in the network
  - Instead, Fused Adam updates all params ***in parallel*** → much faster

TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- PyTorch JIT
- Fused Adam Optimizer

before

```
opt = Adam(params,
           lr=cfg_opt.lr,
           betas=(cfg_opt.adam_beta1, cfg_opt.adam_beta2))
```

after

```
from apex.optimizers import FusedAdam

opt = FusedAdam(params,
                lr=cfg_opt.lr,
                betas=(cfg_opt.adam_beta1, cfg_opt.adam_beta2))
```

The larger the network, the more speed gain

~7% speed gain for vid2vid

TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- PyTorch JIT
- Fused Adam Optimizer
- Thread affinity binding
  - schedule application threads to the same CPU socket
  - better performance when doing CUDA host-to-device and device-to-host transfers

```
+ from imaginaire.utils.gpu_affinity import set_affinity

  def main():
      args = parse_args()
+     set_affinity(args.local_rank)
```

TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- Gradient checkpointing
  - Instead of storing activations for all layers, recompute part of them during the backward process
  - Trade (training) time for memory
  - Can fit larger batch size → may actually run faster per epoch
  - The larger the batch size/computation, the more speed gain for AMP

Chen et al. "Training deep nets with sublinear memory cost." *arXiv preprint arXiv:1604.06174* (2016).

TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- Gradient checkpointing

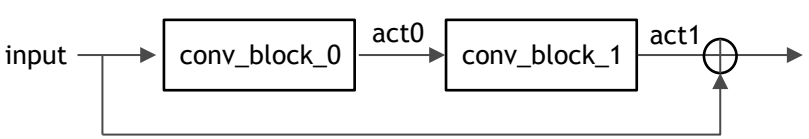
before

```
class _BaseResBlock(nn.Module):
    def __init__(self, in_channels, out_channels, kernel_size,
@@ -91,10 +92,15 @@ class _BaseResBlock(nn.Module):
-    def forward(self, x, *cond_inputs, **kw_cond_inputs):
         dx = self.conv_block_0(x, *cond_inputs, **kw_cond_inputs)
         dx = self.conv_block_1(dx, *cond_inputs, **kw_cond_inputs)
```

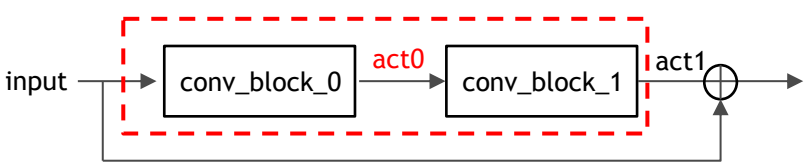
after

```
+ from torch.utils.checkpoint import checkpoint
  class _BaseResBlock(nn.Module):
      def __init__(self, in_channels, out_channels, kernel_size,
  @@ -91,10 +92,15 @@ class _BaseResBlock(nn.Module):
+     def checkpt(self, x, *cond_inputs, **kw_cond_inputs):
+
          dx = self.conv_block_0(x, *cond_inputs, **kw_cond_inputs)
          dx = self.conv_block_1(dx, *cond_inputs, **kw_cond_inputs)
+         return out
+
+     def forward(self, x, *cond_inputs, **kw_cond_inputs):
+         dx = checkpoint(self.checkpt, *cond_inputs, **kw_cond_inputs)
+
```

Residual block



Checkpointing



act0 is not saved but recomputed during backward

TIPS TO HELP VID2VID RUN FASTER

For both AMP O0/O1

- Gradient checkpointing (for vid2vid)
  - ~20% memory reduction
  - ~10% time increase
  - Can fit batch size 2 instead of 1 on 16GB nodes → 1.35X speed up

TIPS TO HELP VID2VID RUN FASTER

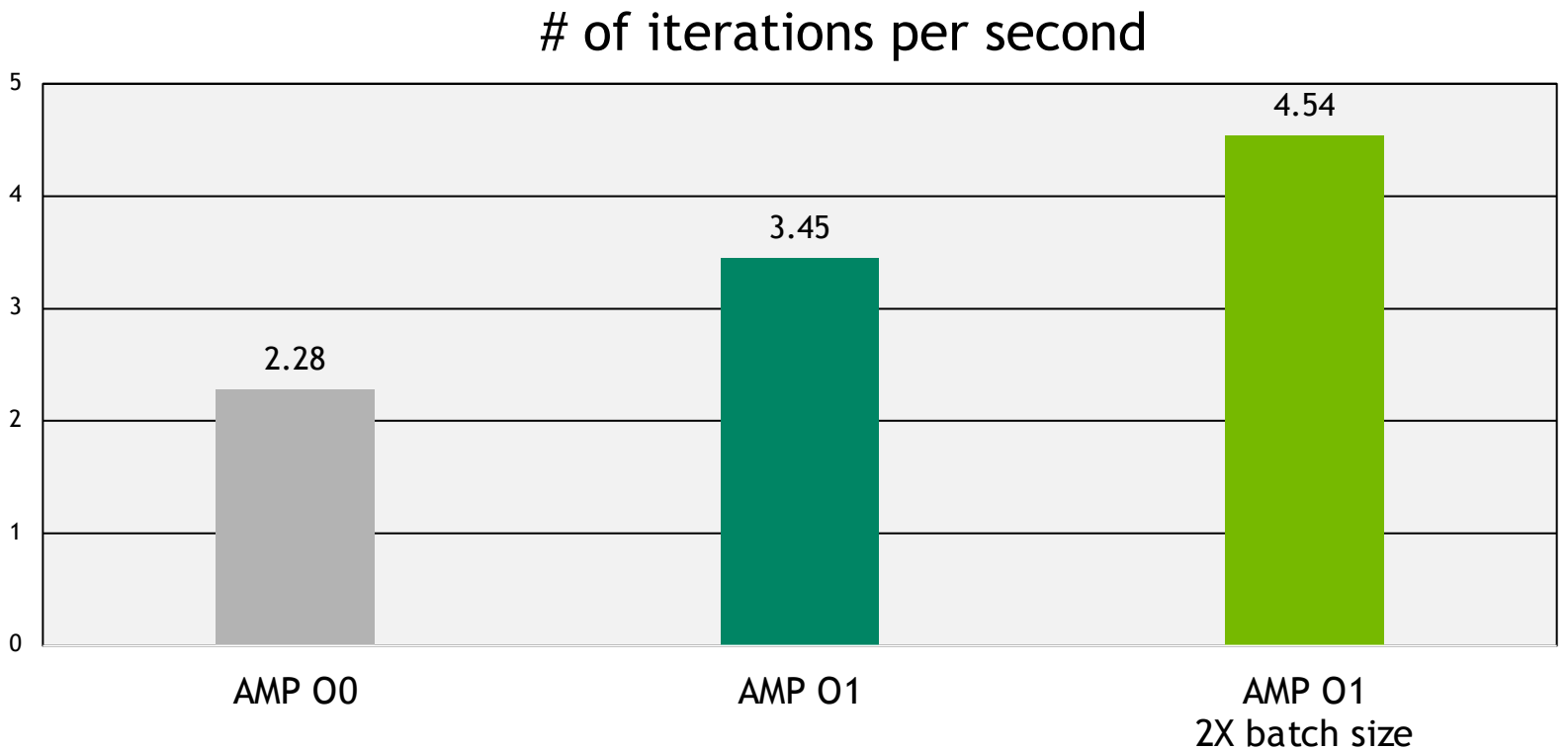
For both AMP O0/O1

- PyTorch JIT
- Fused Adam Optimizer
- Thread affinity binding
- Gradient checkpointing

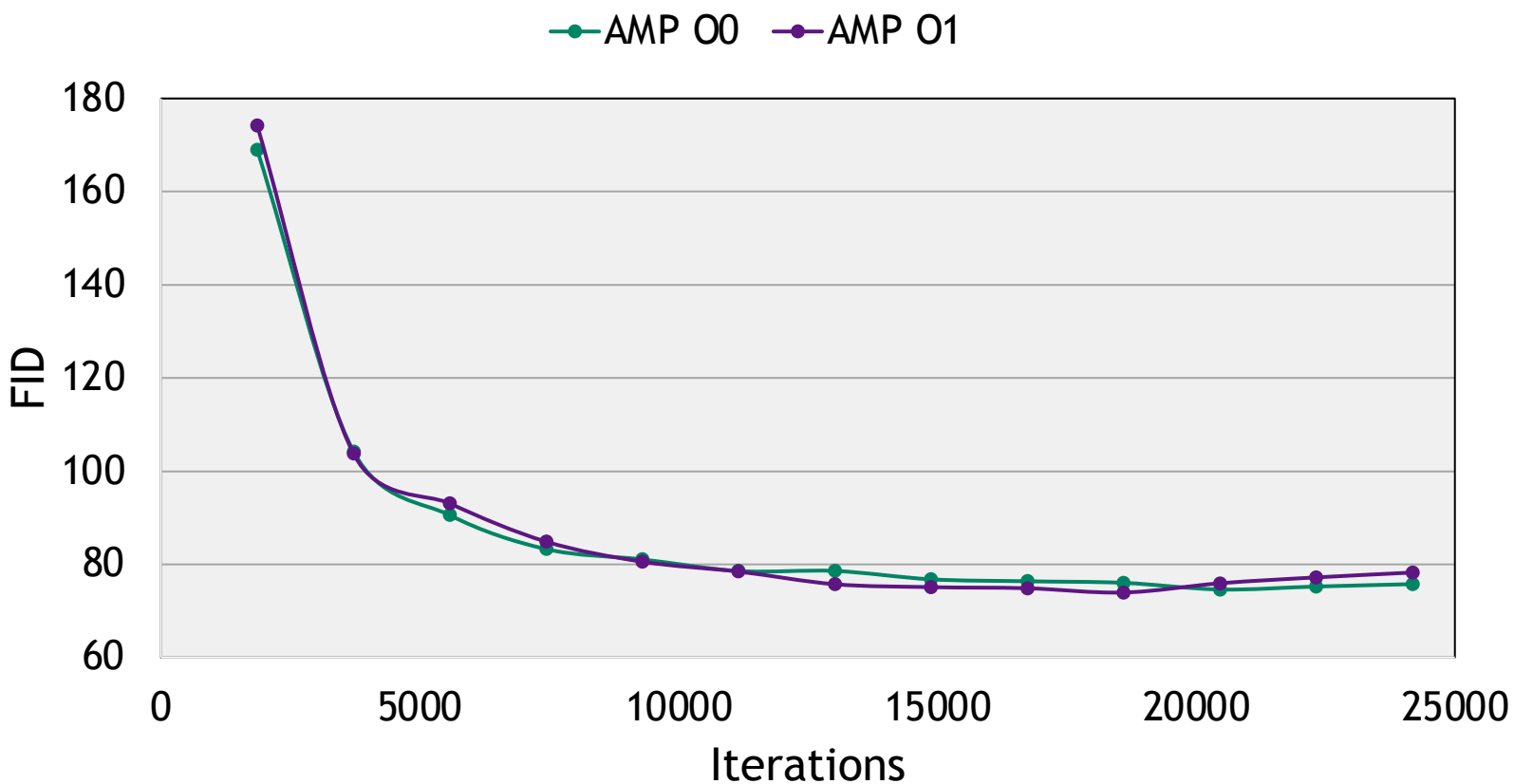
Specs:
Machine: DGX1 V100 32GB
Batch size: 8 images per batch

AMP OBTAINED SPEEDUPS

1.5x faster (2x faster with doubled batch size)



EFFECT ON ACCURACY



CONFIGURATION VS SPEEDUP

ampO1 (including data loading)	Output image resolution	
	512x256	1024x512
BS=8	0.84	1.51
BS=16	1.15	**1.99**
BS=32	1.53	
BS=64	**2.01**	

VIDEO SYNTHESIS

- vid2vid [Wang et al. 2018]

- Few-shot vid2vid [Wang et al. 2019]

APPLICATION

T.-C. Wang, M.-Y. Liu, A. Tao, G. Liu, J. Kautz, B. Catanzaro,
“Few-shot Adaptive Video-to-Video Synthesis,” in NeurIPS 2019.
https://github.com/NVlabs/few-shot-vid2vid

FEW-SHOT VID2VID: OVERVIEW

Original vid2vid

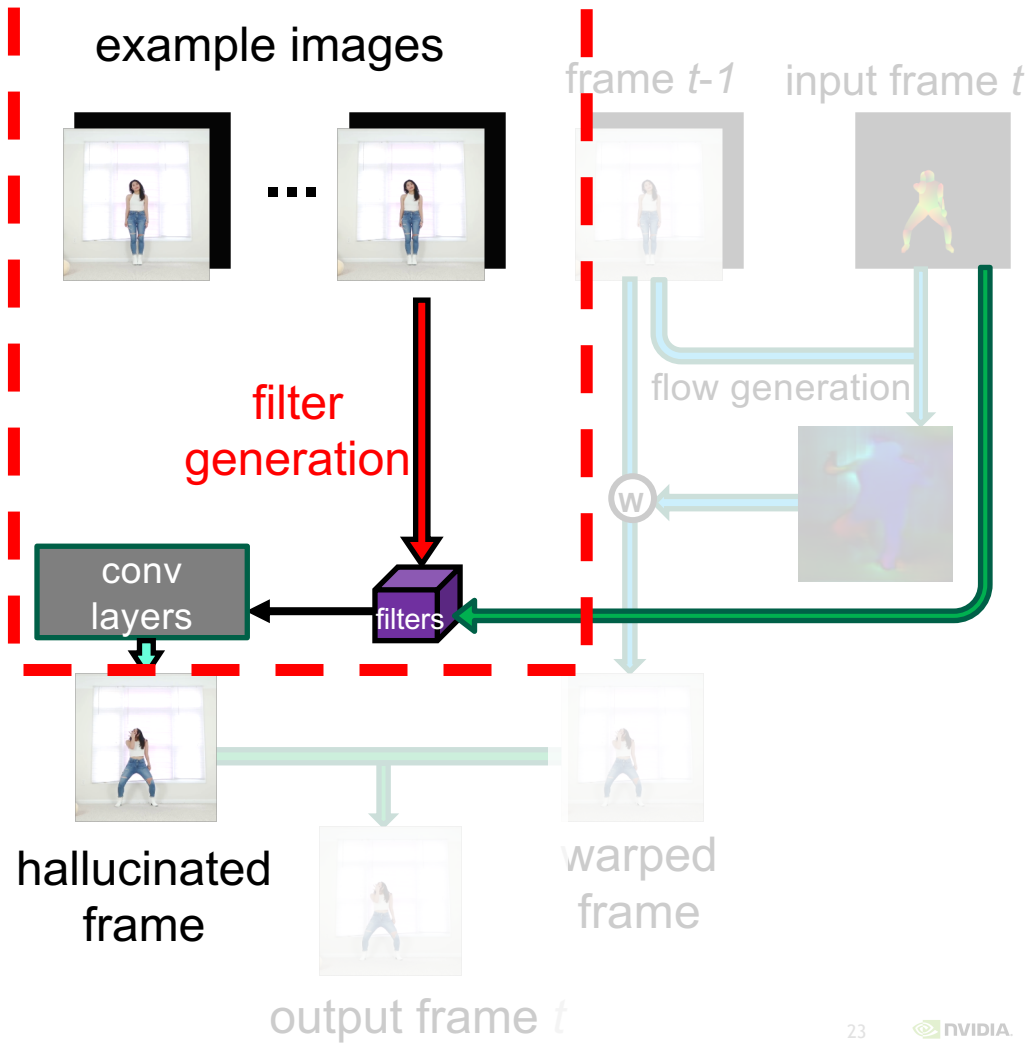
Output frame =
Hallucinated frame + Warped frame

Few-shot vid2vid

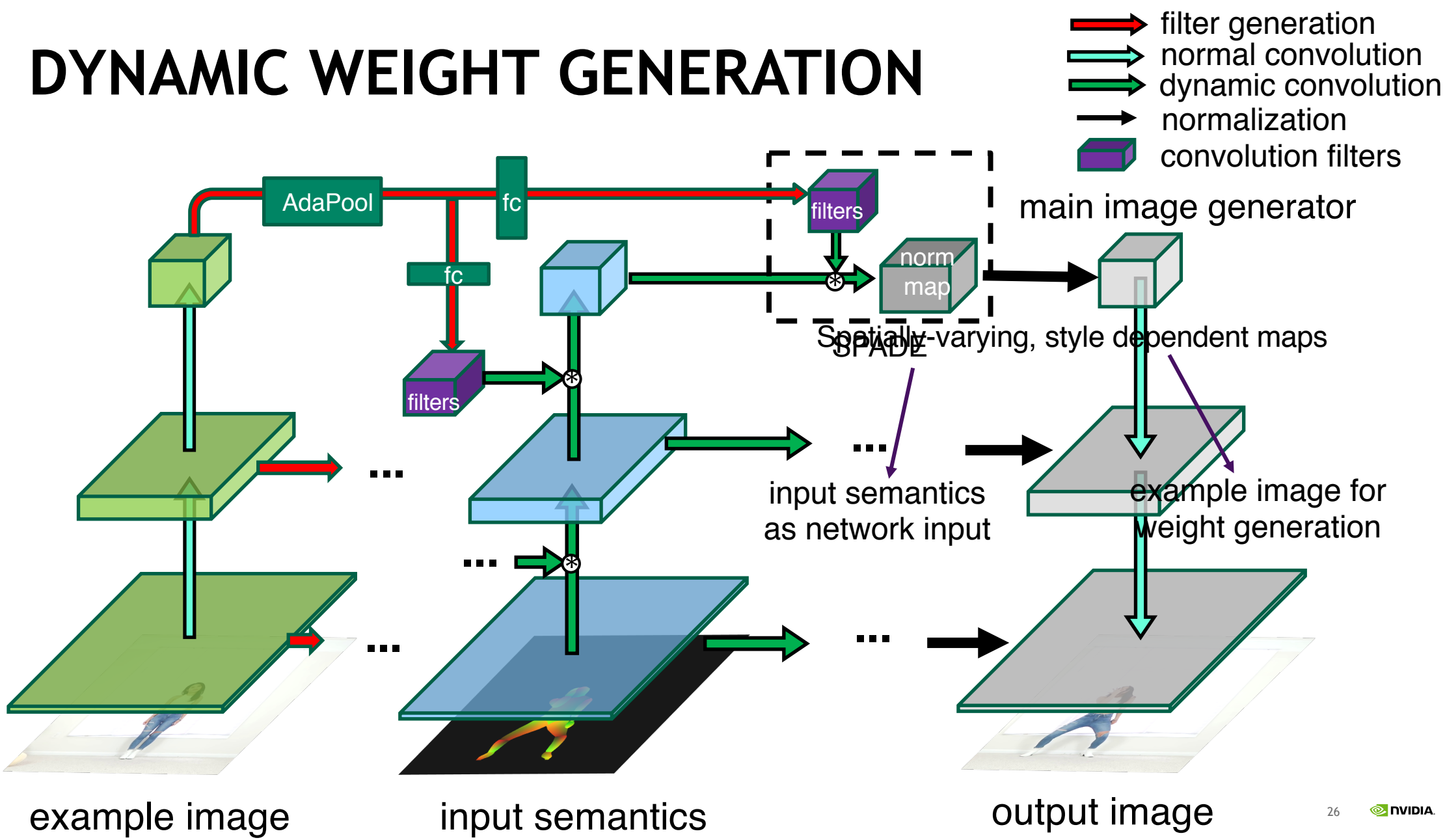
Hallucinated frames

generated based on example images

Using a filter generation scheme



DYNAMIC WEIGHT GENERATION



CHANGES REQUIRED

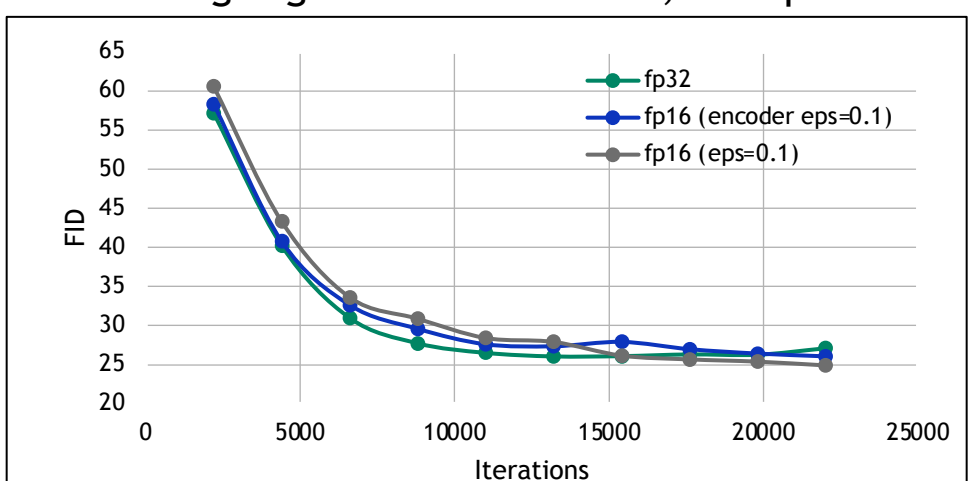
Few-shot vid2vid

- Change epsilon for ***instance norm*** in the weight generation layer

```
CLASS  torch.nn.InstanceNorm2d(num_features: int, eps: float = 1e-05, momentum: float = 0.1,
       affine: bool = False, track_running_stats: bool = False)
```

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

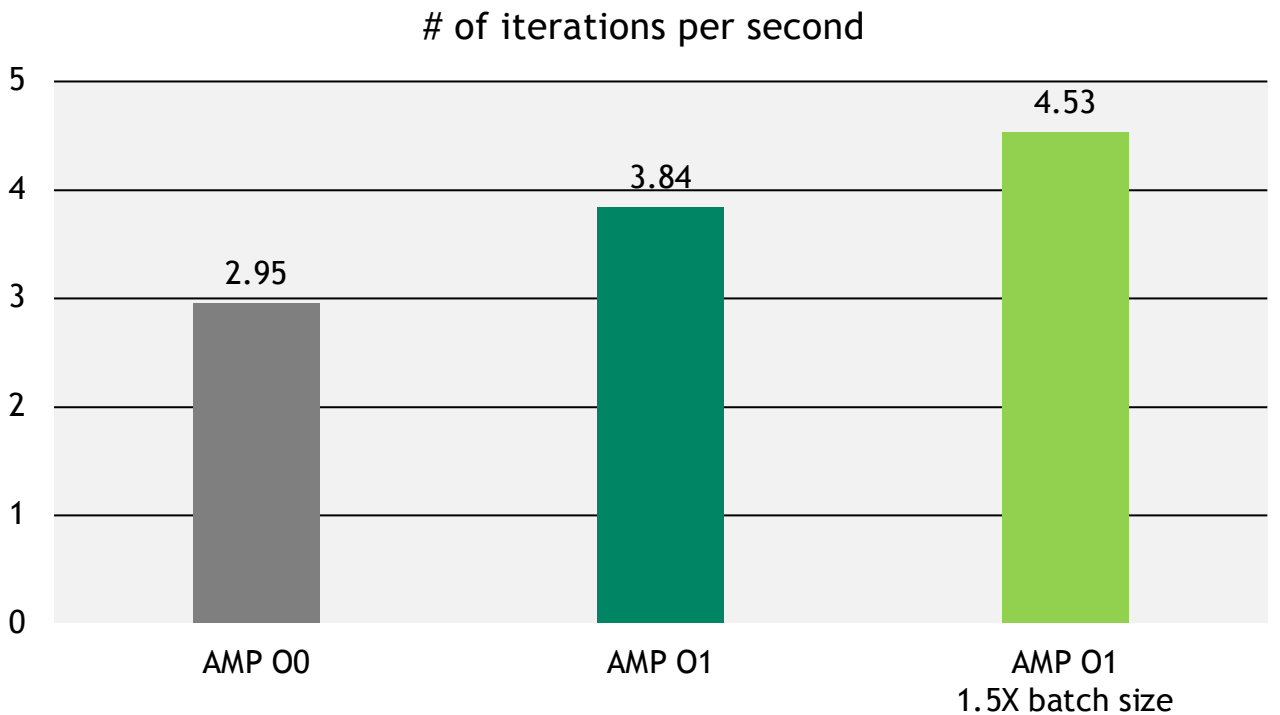
  - For the label encoder in weight generation network, set eps = 0.1



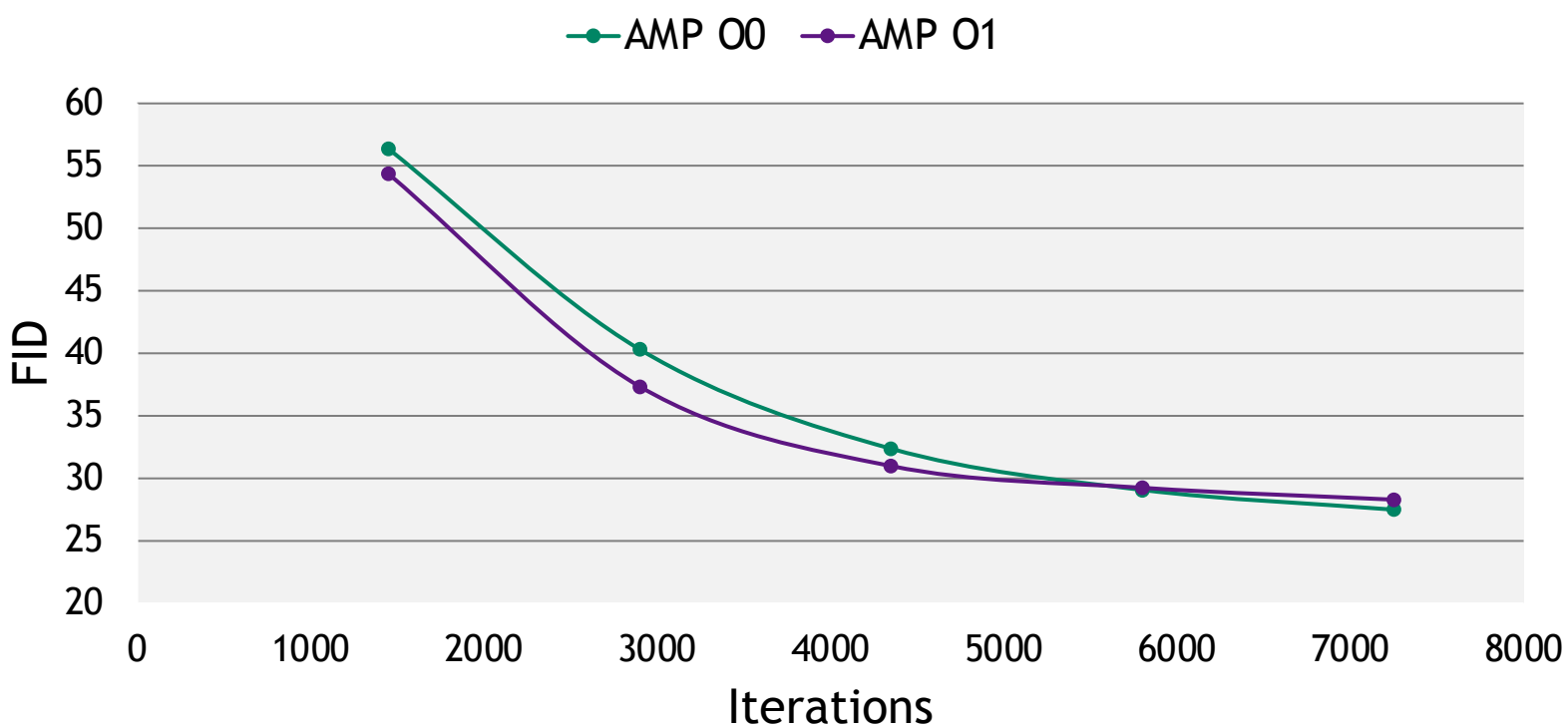
Specs:
Machine: DGX1 V100 32GB
Batch size: 16 images per batch

OBTAINED SPEEDUPS

1.3x faster (1.5x faster with 1.5X batch size)



EFFECT ON ACCURACY




NVIDIA