动态shape修改为静态

export_onnx.py

根据第一部分对export_onnx.py的学习，可以知道contronet、controlunet和decoder都是静态shape的输入输出，只有clip模型是有动态shape。查看export_onnx.py的clip转换代码的模型转换部分：

```
def export_clip_model():
    ...

    onnx_path = "./onnx/CLIP.onnx"

    tokens = torch.zeros(1, 77, dtype=torch.int32)
    input_names = ["input_ids"]
    output_names = ["last_hidden_state"]
    dynamic_axes = {"input_ids": {1: "S"}, "last_hidden_state": {1: "S"}}

    torch.onnx.export(
        clip_model,
        (tokens),
        onnx_path,
        verbose=True,
        opset_version=18,
        do_constant_folding=True,
        input_names=input_names,
        output_names=output_names,
        dynamic_axes=dynamic_axes,
    )
    ...
```

可以看到，torch.onnx.export转换函数中主要是dynamic_axes参数在设置动态shape，对应的输入为`dynamic_axes = {"input_ids": {1: "S"}, "last_hidden_state": {1: "S"}}`。这个输入表示将名称为“input_ids”和“last_hidden_state”的节点的第一维输入设置为动态，并命名为S。删去该参数即可。

此外tokens原本定义的77是占位值，当dynamic_axes参数去掉之后，tokens第一维变为固定大小77了。

```
def export_clip_model():
    ...

    onnx_path = "./onnx/CLIP.onnx"

    tokens = torch.zeros(1, 77, dtype=torch.int32)
    input_names = ["input_ids"]
    output_names = ["last_hidden_state"]

    torch.onnx.export(
        clip_model,
        (tokens),
        onnx_path,
        verbose=True,
```

```
        opset_version=18,
        do_constant_folding=True,
        input_names=input_names,
        output_names=output_names,
    )
    ...
```

验证代码：

```python
# 使用ONNX Runtime检查输入输出形状
import onnxruntime as ort

sess = ort.InferenceSession("./onnx/CLIP.onnx")
for input in sess.get_inputs():
    print(f"Input: {input.name}, Shape: {input.shape}")

for output in sess.get_outputs():
    print(f"Output: {output.name}, Shape: {output.shape}")
```

```
======================== DECODER onnx model verify done!
root@03cbfec78aae:~/TensorRT-StableDiffusion# python3 test_model_static.py
Input: input_ids, Shape: [1, 77]
Output: last_hidden_state, Shape: [1, 77, 768]
```

onnx2trt.py

以clip转换代码为例：

```python
def export_clip_model():
    onnx_path = "./onnx/CLIP.onnx"
    plan_path = "./engine/CLIP.plan"

    onnx2trt(onnx_path, plan_path, [(1, 77)], [(1, 77)], [(1, 77)])

    # onnx2trt(onnx_path, plan_path, [(1, 1)], [(1, 77)], [(1, 128)], use_fp16=True)
    print("======================== CLIP onnx2trt done!")
```

这里调用了onnx2trt函数，就在onnx2trt.py内，查看源码看输入情况：

```python
def onnx2trt(onnxFile, plan_name, min_shapes, opt_shapes, max_shapes, max_workspace_size = None, use_fp16=False, builder_opt_evel=None):
    ...
```

这里看到输入[(1, 77)], [(1, 77)], [(1, 77)]对应min_shapes，opt_shapes，max_shapes。当这三个参数的值是一模一样的时候，相当于静态输入。此时clip已经是静态输入。

controlnet

输入为：

```
x_noisy = torch.randn(1, 4, 32, 48, dtype=torch.float32)
timestep = torch.tensor([1], dtype=torch.int32)
context = torch.randn(1, 77, 768, dtype=torch.float32)
hint = torch.randn(1, 3, 256, 384, dtype=torch.float32)
```

根据get_shapes函数计算B=1，S=77.

```
# controlnet
def export_control_net_model():
    def get_shapes(B, S):
        return [(B, 4, 32, 48), (B, 3, 256, 384), tuple([B]), (B, S, 768)]

    onnx_path = "./onnx/CONTROL_NET.onnx"
    plan_path = "./engine/CONTROL_NET.plan"

    onnx2trt(onnx_path, plan_path,
             get_shapes(1, 77),
             get_shapes(1, 77),
             get_shapes(1, 77))
```

controlunet

输入为：

```
x_noisy = torch.randn(1, 4, 32, 48, dtype=torch.float32)
timestep = torch.tensor([1], dtype=torch.int32)
context = torch.randn(1, 77, 768, dtype=torch.float32)

control = [
    torch.randn(1, 320, 32, 48, dtype=torch.float32),
    torch.randn(1, 320, 32, 48, dtype=torch.float32),
    torch.randn(1, 320, 32, 48, dtype=torch.float32),
    torch.randn(1, 320, 16, 24, dtype=torch.float32),
    torch.randn(1, 640, 16, 24, dtype=torch.float32),
    torch.randn(1, 640, 16, 24, dtype=torch.float32),
    torch.randn(1, 640, 8, 12, dtype=torch.float32),
    torch.randn(1, 1280, 8, 12, dtype=torch.float32),
    torch.randn(1, 1280, 8, 12, dtype=torch.float32),
    torch.randn(1, 1280, 4, 6, dtype=torch.float32),
    torch.randn(1, 1280, 4, 6, dtype=torch.float32),
    torch.randn(1, 1280, 4, 6, dtype=torch.float32),
    torch.randn(1, 1280, 4, 6, dtype=torch.float32),
]
```

计算B=1，S=77。

```
def export_controlled_unet_model():
    def get_shapes(B, S):
        return [(B, 4, 32, 48), tuple([B]), (B, S, 768),
                (B, 320, 32, 48),
                (B, 320, 32, 48),
                (B, 320, 32, 48),
                (B, 320, 16, 24),
                (B, 640, 16, 24),
                (B, 640, 16, 24),
```

```
                (B, 640, 8, 12),
                (B, 1280, 8, 12),
                (B, 1280, 8, 12),
                (B, 1280, 4, 6),
                (B, 1280, 4, 6),
                (B, 1280, 4, 6),
                (B, 1280, 4, 6)]

    onnx_path = "./onnx/CONTROL_UNET.onnx"

    plan_path = "./engine/CONTROL_UNET.plan"

    onnx2trt(onnx_path, plan_path,
             get_shapes(1, 77),
             get_shapes(1, 77),
             get_shapes(1, 77))
```

decoder

输入为:

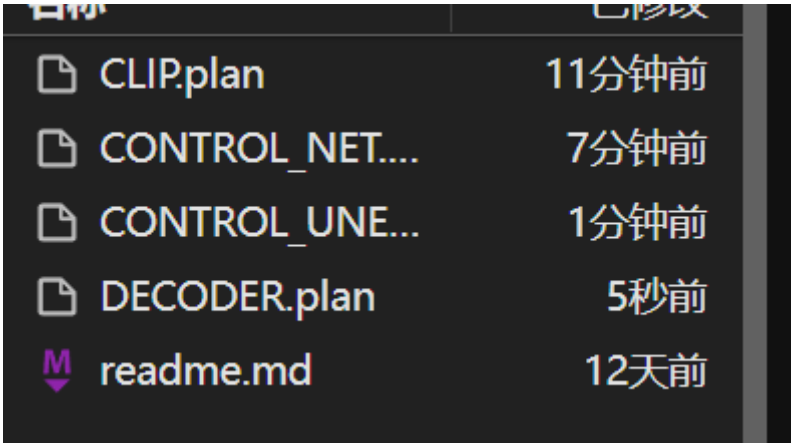
```
latent = torch.randn(1, 4, 32, 48, dtype=torch.float32)
```

源码已经是静态输入

```
def export_decoder_model():
    onnx_path = "./onnx/DECODER.onnx"
    plan_path = "./engine/DECODER.plan"

    onnx2trt(onnx_path, plan_path,
            [(1, 4, 32, 48)], [(1, 4, 32, 48)], [(1, 4, 32, 48)])
```

结果



转换时间controlunet>controlnet>decoder>clip（云平台4090）

测试

tetexec方法

```
./trtexec --loadEngine=./engine/CLIP.plan --dumpProfile
```

```
Input binding for input_ids with dimensions 1x77 is created.
Output binding for last_hidden_state with dimensions 1x77x768 is created.
```

```
./trtexec --loadEngine=./engine/CONTROL_NET.plan --dumpProfile
```

```
Using random values for input x_noisy
Input binding for x_noisy with dimensions 1x4x32x48 is created.
Using random values for input hint
Input binding for hint with dimensions 1x3x256x384 is created.
Using random values for input timestep
Input binding for timestep with dimensions 1 is created.
Using random values for input context
Input binding for context with dimensions 1x77x768 is created.
Output binding for latent with dimensions 1x320x32x48 is created.
Output binding for 816 with dimensions 1x320x32x48 is created.
Output binding for 1341 with dimensions 1x320x32x48 is created.
Output binding for 1343 with dimensions 1x320x16x24 is created.
Output binding for 1869 with dimensions 1x640x16x24 is created.
Output binding for 2394 with dimensions 1x640x16x24 is created.
Output binding for 2396 with dimensions 1x640x8x12 is created.
Output binding for 2922 with dimensions 1x1280x8x12 is created.
Output binding for 3447 with dimensions 1x1280x8x12 is created.
Output binding for 3449 with dimensions 1x1280x4x6 is created.
Output binding for 3494 with dimensions 1x1280x4x6 is created.
Output binding for 3539 with dimensions 1x1280x4x6 is created.
Output binding for 4108 with dimensions 1x1280x4x6 is created.
Starting inference
```

```
./trtexec --loadEngine=./engine/CONTROL_UNET.plan --dumpProfile
```

输出太长了看不到

```
./trtexec --loadEngine=./engine/DECODER.plan --dumpProfile
```

```
[I] Input binding for latent with dimensions 1x4x32x48 is created.
[I] Output binding for image with dimensions 1x3x256x384 is created.
```