```
from unittest.mock import patch

import pytest
import torch

from swarm_models.huggingface_pipeline import HuggingfacePipeline



@pytest.fixture
def mock_pipeline():
    with patch("swarms.models.huggingface_pipeline.pipeline") as mock:
        yield mock



@pytest.fixture
def pipeline(mock_pipeline):
    return HuggingfacePipeline(
        "text-generation", "meta-llama/Llama-2-13b-chat-hf"
    )



def test_init(pipeline, mock_pipeline):
    assert pipeline.task_type == "text-generation"
    assert pipeline.model_name == "meta-llama/Llama-2-13b-chat-hf"
    assert (
        pipeline.use_fp8 is True
```

```
        if torch.cuda.is_available()
        else False
    )
    mock_pipeline.assert_called_once_with(
        "text-generation",
        "meta-llama/Llama-2-13b-chat-hf",
        use_fp8=pipeline.use_fp8,
    )


def test_run(pipeline, mock_pipeline):
    mock_pipeline.return_value = "Generated text"
    result = pipeline.run("Hello, world!")
    assert result == "Generated text"
    mock_pipeline.assert_called_once_with("Hello, world!")


def test_run_with_exception(pipeline, mock_pipeline):
    mock_pipeline.side_effect = Exception("Test exception")
    with pytest.raises(Exception):
        pipeline.run("Hello, world!")


def test_run_with_different_task(pipeline, mock_pipeline):
    mock_pipeline.return_value = "Generated text"
    result = pipeline.run("text-classification", "Hello, world!")
```

```
assert result == "Generated text"

mock_pipeline.assert_called_once_with(
    "text-classification", "Hello, world!"
)
```