Help with ONNX file export code in Python that I can add into the SageMaker Python SDK codebase for Training pipeline, and Promotion pipeline.

You can add a small ONNX-export step at the end of your Hugging Face training and promotion flows so that `output.tar.gz` contains both the usual HF files and `model.onnx` (or a dedicated ONNX artifact).[1] [2] [3]

Below assumes a text/sequence classification model trained with `transformers` + PyTorch (e.g., `AutoModelForSequenceClassification`) and that your training script already saves into `/opt/ml/model`.

1. Install dependencies in your training image

In your SageMaker training container (Dockerfile or `requirements.txt`):

```
pip install "transformers>=4.28" "optimum[onnxruntime]" onnx onnxruntime
```

`optimum`'s ONNX exporter simplifies Hugging Face → ONNX conversion and works well in SageMaker.[4] [3] [1]

2. ONNX export in the training script (end of training job)

At the *end* of your training script (after `Trainer.train()` and `trainer.save_model("/opt/ml/model")` or equivalent), add something like:

```
import os
from pathlib import Path

from transformers import AutoTokenizer, AutoModelForSequenceClassification
from optimum.exporters.onnx import export
from optimum.exporters.onnx import OnnxConfig
from transformers import AutoConfig

MODEL_DIR = Path("/opt/ml/model")  # SageMaker model dir
ONNX_PATH = MODEL_DIR / "model.onnx"

def export_to_onnx():
    # Load config, tokenizer, and model from the just-saved directory
```

```
    config = AutoConfig.from_pretrained(MODEL_DIR)
    tokenizer = AutoTokenizer.from_pretrained(MODEL_DIR, use_fast=True)
    model = AutoModelForSequenceClassification.from_pretrained(MODEL_DIR, config=config)

    # Build default ONNX config for sequence classification
    onnx_config = OnnxConfig.from_model_config(config, task="sequence-classification")

    # Export using Optimum
    export(
        model=model,
        config=onnx_config,
        output=ONNX_PATH,
        tokenizer=tokenizer,
        opset=14,  # or 13/17 depending on your target runtime
    )

    print(f"Exported ONNX model to: {ONNX_PATH}")

if __name__ == "__main__":
    # ... your existing training code ...

    # After training + saving HF artifacts
    export_to_onnx()
```

Key points:

- Use `/opt/ml/model` so SageMaker automatically tars `model.onnx` into `output.tar.gz`.[5] [6]
- `OnnxConfig.from_model_config(..., task="sequence-classification")` matches HF's recommended export path for text classification.[2] [7]
- Adjust model class (e.g., `AutoModelForTokenClassification`, `AutoModelForSeq2SeqLM`) and `task` if you're not doing pure sequence classification.[7] [2]

If you prefer the lower-level `transformers.onnx` API, a minimal alternative is:

```
from transformers.onnx import export, FeaturesManager
from pathlib import Path
from transformers import AutoTokenizer, AutoConfig, AutoModelForSequenceClassification

MODEL_DIR = Path("/opt/ml/model")
ONNX_PATH = MODEL_DIR / "model.onnx"

def export_to_onnx():
    config = AutoConfig.from_pretrained(MODEL_DIR)
    tokenizer = AutoTokenizer.from_pretrained(MODEL_DIR, use_fast=True)
    model = AutoModelForSequenceClassification.from_pretrained(MODEL_DIR, config=config)

    feature = "sequence-classification"
    onnx_config_cls = FeaturesManager.get_config_class_for_model_type(config.model_type)
    onnx_config = onnx_config_cls(config, task=feature)

    export(
        preprocessor=tokenizer,
        model=model,
        config=onnx_config,
```

```
        output=ONNX_PATH,
        opset=14,
    )
```

This follows the official `transformers` ONNX docs.[8] [2]

3. Promotion pipeline: ONNX-only or model-package step

In your “Promotion” or “Model Registry / Packaging” pipeline, you typically have a step that: downloads model artifacts from the training job S3 location → validates → re-uploads to the next S3 prefix or registers a model package.

You can do one of:

1. **Reuse the ONNX already in output.tar.gz**
    - Untar `output.tar.gz`, verify that `model.onnx` exists, maybe validate it with ONNX checker, and then:
        - Either re-tar and store as the promoted artifact, or
        - Extract just `model.onnx` and upload it separately for your Spring Boot inference service.[9] [10] [11]
2. **(Re)export ONNX during promotion (if you want isolation)**
    - Promotion Python step:

```
import tarfile
from pathlib import Path
import onnx
from onnx import checker

from transformers import AutoConfig, AutoTokenizer, AutoModelForSequenceClassification
from optimum.exporters.onnx import export, OnnxConfig

def promote_model(input_tar_s3_uri: str, output_s3_uri: str, local_dir: str = "/opt/ml/pr
    # 1. Download output.tar.gz from S3 to local_dir (using boto3 or SageMaker Processing
    # 2. Extract
    tar_path = Path(local_dir) / "output.tar.gz"
    extract_dir = Path(local_dir) / "extracted"
    extract_dir.mkdir(parents=True, exist_ok=True)

    with tarfile.open(tar_path) as tar:
        tar.extractall(extract_dir)

    model_dir = extract_dir  # where config.json, model.safetensors, etc. live

    # 3. Export ONNX (or skip if model.onnx already exists and you just validate)
    config = AutoConfig.from_pretrained(model_dir)
    tokenizer = AutoTokenizer.from_pretrained(model_dir, use_fast=True)
    model = AutoModelForSequenceClassification.from_pretrained(model_dir, config=config)

    onnx_path = model_dir / "model.onnx"
    onnx_config = OnnxConfig.from_model_config(config, task="sequence-classification")
```

```
    export(model=model, config=onnx_config, output=onnx_path, tokenizer=tokenizer, opset=

    # 4. Optional: validate ONNX
    m = onnx.load(str(onnx_path))
    checker.check_model(m)

    # 5. Re-tar or upload model.onnx + tokenizer files to output_s3_uri
    #    (using boto3 to put_object, or tarfile + boto3)
```

That gives you a promotion step that can fail fast if ONNX export breaks, independent of training.[12] [3] [5]

4. Notes for SageMaker integration

- Training job: nothing changes in your `Estimator`/`HuggingFace` estimator call; you only modify the `entry_point` script to include ONNX export at the end.[13] [5]
- If you use SageMaker **Processing** or **Pipelines** for promotion, the above promotion snippet can live in a Processing script that takes training S3 URI as input and writes ONNX artifact to a promotion S3 URI.[14] [5]
- For large models, consider:
  - Using dynamic axes in the ONNX config for variable sequence length.[2] [7]
  - Possibly exporting an optimized / quantized variant using Optimum's ONNXRuntime integration later.[3] [15]

If you share your exact HF class (e.g., `AutoModelForSequenceClassification` vs `AutoModelForTokenClassification`) and how you currently set up the SageMaker `Estimator`, I can write a drop-in training `entry_point.py` showing where to plug this export step.

⁂

1. https://huggingface.co/docs/optimum-onnx/onnx/usage_guides/export_a_model
2. https://huggingface.co/docs/transformers/v4.29.1/serialization
3. https://www.philschmid.de/convert-transformers-to-onnx
4. https://github.com/huggingface/optimum-onnx
5. https://amazon-sagemaker-examples-anve.readthedocs.io/en/latest/sagemaker-python-sdk/mxnet_onnx_export/mxnet_onnx_export.html
6. https://docs.aws.amazon.com/sagemaker/latest/dg/your-algorithms-training-algo-output.html
7. https://huggingface.co/docs/transformers/v4.28.1/serialization
8. https://huggingface.co/docs/transformers/v4.20.1/en/serialization
9. https://lindevs.com/check-if-onnx-model-is-valid-using-python
10. https://onnx.ai/onnx/api/checker.html
11. https://onnx.ai/onnx/intro/python.html
12. https://onnxruntime.ai/huggingface

13. http://www.nbertagnolli.com/jekyll/update/2023/06/05/Deploy_ONNX_Transformer_to_Sagemaker.html
14. https://sagemaker-examples.readthedocs.io/en/latest/sagemaker-pipelines/tabular/abalone_build_train_deploy/sagemaker-pipelines-preprocess-train-evaluate-batch-transform.html
15. https://www.youtube.com/watch?v=_AKFDOnrZz8
16. https://huggingface.co/docs/optimum-onnx/en/onnx/usage_guides/export_a_model
17. https://github.com/huggingface/optimum/blob/main/optimum/exporters/onnx/convert.py
18. https://stackoverflow.com/questions/78635731/cannot-export-huggingface-model-to-onnx-with-optimum-cli
19. https://github.com/aws/amazon-sagemaker-examples/issues/968
20. https://github.com/JohnSnowLabs/spark-nlp/blob/master/examples/python/transformers/onnx/HuggingFace_ONNX_in_Spark_NLP_BertForSequenceClassification.ipynb