TotalSegmentator PyTorch to CoreML Conversion (Fixed v2)

This notebook handles complex dependency conflicts between TotalSegmentator, PyTorch, CoreMLTools, and NumPy.

Strategy

- Restart runtime when needed to avoid binary incompatibility
- Use compatible NumPy version
- Install packages in specific order

Step 1: Complete Environment Reset

```
# First, completely reset the environment
!pip uninstall -y torch torchvision torchaudio tr
!pip uninstall -y nnunetv2 totalsegmentator corem
```

```
Found existing installation: torch 2.6.0+cpu
Uninstalling torch-2.6.0+cpu:
  Successfully uninstalled torch-2.6.0+cpu
Found existing installation: torchvision 0.21.
Uninstalling torchvision-0.21.0+cpu:
  Successfully uninstalled torchvision-0.21.0+
Found existing installation: torchaudio 2.6.0+
Uninstalling torchaudio-2.6.0+cpu:
  Successfully uninstalled torchaudio-2.6.0+cp
WARNING: Skipping triton as it is not installe
Found existing installation: numpy 2.0.2
Uninstalling numpy-2.0.2:
  Successfully uninstalled numpy-2.0.2
Found existing installation: scipy 1.15.3
Uninstalling scipy-1.15.3:
  Successfully uninstalled scipy-1.15.3
Found existing installation: scikit-image 0.25
Uninstalling scikit-image-0.25.2:
  Successfully uninstalled scikit-image-0.25.2
Found existing installation: pandas 2.2.2
Uninstalling pandas-2.2.2:
  Successfully uninstalled pandas-2.2.2
WARNING: Skipping nnunetv2 as it is not instal
WARNING: Skipping totalsegmentator as it is nc
WARNING: Skipping coremltools as it is not ins
WARNING: Skipping nibabel as it is not install
Found existing installation: thinc 8.3.6
Uninstalling thinc-8.3.6:
```

Exception

Please explain the error:

```
Exception: For an ML Prog
```

Hypothesis

The error message `Exception: For an ML Program, extension must be .mlpackage (not .mlmodel). Please see https://coremltools.readme conversion-api#target-conversion-formats to see the difference between neuralnetwork and mlprogram model types.` clearly indicates that the `coreml_model` object being saved is in the "ML Program" format, which requires the file extension `.mlpackage` instead of the older `.mlmodel` extension.

Looking at the code in `ipython-input-4-3373099344.py`, there are three fallback methods for creating `coreml_model`. The traceback originates from the saving step in `ipython-input-5-3373099344.py`. This means that the conversion attempted in `ipython-input-4-3373099344.py` must have succeeded using one of the methods, and the resulting `coreml_model` object is an ML Program.

Since the error occurs during saving with `.mlmodel`, it's likely that one of the successful conversion methods resulted in an ML Program. CoreMLTools

```
Successfully uninstalled thinc-8.3.6
```

Step 2: Install Core Dependencies with Compatible NumPy

```
# Install numpy first with specific version that v
!pip install numpy==1.24.3

# Install PyTorch 2.1.2 with dependencies
!pip install torch==2.1.2 torchvision==0.16.2 --ir

# Install scipy and scikit-image with compatible v
!pip install scipy==1.10.1 scikit-image==0.21.0
```

```
Requirement already satisfied: numpy==1.24.3 i
Looking in indexes: https://download.pytorch.c
Requirement already satisfied: torch==2.1.2 ir
Requirement already satisfied: torchvision==0.
Requirement already satisfied: filelock in /us
Requirement already satisfied: typing-extensic
Requirement already satisfied: sympy in /usr/l
Requirement already satisfied: networkx in /us
Requirement already satisfied: jinja2 in /usr/
Requirement already satisfied: fsspec in /usr/
Requirement already satisfied: numpy in /usr/l
Requirement already satisfied: requests in /us
Requirement already satisfied: pillow!=8.3.*,>
Requirement already satisfied: MarkupSafe>=2.0
Requirement already satisfied: charset-normali
Requirement already satisfied: idna<4,>=2.5 ir
Requirement already satisfied: urllib3<3,>=1.2
Requirement already satisfied: certifi>=2017.4
Requirement already satisfied: mpmath<1.4,>=1.
Requirement already satisfied: scipy==1.10.1 i
Requirement already satisfied: scikit-image==0
Requirement already satisfied: numpy<1.27.0,>=
Requirement already satisfied: networkx>=2.8 i
Requirement already satisfied: pillow>=9.0.1 i
Requirement already satisfied: imageio>=2.27 i
Requirement already satisfied: tifffile>=2022.
Requirement already satisfied: PyWavelets>=1.1
Requirement already satisfied: packaging>=21 i
Requirement already satisfied: lazy_loader>=0.
```

Step 3: Install CoreMLTools and Medical Imaging Libraries

```
# Install CoreMLTools - try version 7.2 for bette
!pip install coremltools==7.2

# Install medical imaging libraries
```

versions 7.0 and later default to the ML Program format when converting PyTorch models. The user is using `coremltools==7.2`, which would explain why the conversion might be producing an ML Program.

Suggested Changes

To fix this, we need to save the model with the correct file extension `.mlpackage`. We should also update the Swift integration code to expect the `.mlpackage` format.

```
# Save the model
output_dir = Path("./mode
output_dir.mkdir(exist_ok=

# Change the file extensi
model_path = output_dir /
coreml_model.save(str(mod
print(f"✅ Model saved to

# Create Swift integratio
# Update the Swift code t
swift_code = """import Co
import Vision

class TotalSegmentatorWra
    private let model: ML

    init() throws {
        let config = MLMo
        config.computeUni
        // Update the mod
        self.model = try
    }

    func segment(ctVolume
        let input = Total
        let output = try
```

```
!pip install nibabel==5.2.0
!pip install SimpleITK==2.3.1
!pip install matplotlib==3.7.2
!pip install tqdm==4.66.1
!pip install pandas==2.0.3
```

```
Requirement already satisfied: coremltools==7.
Requirement already satisfied: numpy>=1.14.5 i
Requirement already satisfied: protobuf<=4.0.0
Requirement already satisfied: sympy in /usr/l
Requirement already satisfied: tqdm in /usr/lo
Requirement already satisfied: packaging in /u
Requirement already satisfied: attrs>=21.3.0 i
Requirement already satisfied: cattrs in /usr/
Requirement already satisfied: pyaml in /usr/l
Requirement already satisfied: typing-extensio
Requirement already satisfied: PyYAML in /usr/
Requirement already satisfied: mpmath<1.4,>=1.
Requirement already satisfied: nibabel==5.2.0
Requirement already satisfied: numpy>=1.20 in
Requirement already satisfied: packaging>=17 i
Requirement already satisfied: SimpleITK==2.3.
Collecting matplotlib==3.7.2
  Using cached matplotlib-3.7.2-cp311-cp311-ma
Requirement already satisfied: contourpy>=1.0.
Requirement already satisfied: cycler>=0.10 ir
Requirement already satisfied: fonttools>=4.22
Requirement already satisfied: kiwisolver>=1.0
Requirement already satisfied: numpy>=1.20 in
Requirement already satisfied: packaging>=20.0
Requirement already satisfied: pillow>=6.2.0 i
Requirement already satisfied: pyparsing<3.1,>
Requirement already satisfied: python-dateutil
Requirement already satisfied: six>=1.5 in /us
Using cached matplotlib-3.7.2-cp311-cp311-many
Installing collected packages: matplotlib
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.
    Uninstalling matplotlib-3.10.0:
      Successfully uninstalled matplotlib-3.10
ERROR: pip's dependency resolver does not curr
seaborn 0.13.2 requires pandas>=1.2, which is
fastai 2.7.19 requires pandas, which is not ir
Successfully installed matplotlib-3.7.2
WARNING: The following packages were
previously imported in this runtime:
  [matplotlib,mpl_toolkits]
You must restart the runtime in order to use
newly installed versions.
```

RESTART SESSION

```
Requirement already satisfied: tqdm==4.66.1 ir
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-package
    status = run_func(*args)
             ^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-package
^C
```

```
            return output.out
        }
    }
}
"""

swift_path = output_dir /
with open(swift_path, "w"
    f.write(swift_code)

print(f"✅ Swift integrat
```

Use code with caution

Rate this answer

Step 4: Install TotalSegmentator Without Dependencies First

```
# Install dicom2nifti manually first
!pip install pydicom==2.4.3
!pip install dicom2nifti==2.4.8

# Install nnUNet dependencies
!pip install batchgenerators==0.25
!pip install nnunetv2==2.2.1 --no-deps

# Finally install TotalSegmentator without depende
!pip install totalsegmentator==2.2.1 --no-deps
```

```
Collecting pydicom==2.4.3
  Downloading pydicom-2.4.3-py3-none-any.whl.m
Downloading pydicom-2.4.3-py3-none-any.whl (1.
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.
Installing collected packages: pydicom
Successfully installed pydicom-2.4.3
Collecting dicom2nifti==2.4.8
  Downloading dicom2nifti-2.4.8-py3-none-any.w
Requirement already satisfied: nibabel in /usr
Requirement already satisfied: numpy in /usr/l
Requirement already satisfied: scipy in /usr/l
Requirement already satisfied: pydicom>=2.2.0
Collecting python-gdcm (from dicom2nifti==2.4.
  Downloading python_gdcm-3.0.25-cp311-cp311-m
Requirement already satisfied: packaging>=17 i
Downloading dicom2nifti-2.4.8-py3-none-any.whl
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43
Downloading python_gdcm-3.0.25-cp311-cp311-mar
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12
Installing collected packages: python-gdcm, di
Successfully installed dicom2nifti-2.4.8 pythc
Collecting batchgenerators==0.25
  Downloading batchgenerators-0.25.tar.gz (61
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: pillow>=7.1.2 i
Requirement already satisfied: numpy>=1.10.2 i
Requirement already satisfied: scipy in /usr/l
Requirement already satisfied: scikit-image ir
Requirement already satisfied: scikit-learn ir
Collecting future (from batchgenerators==0.25)
  Downloading future-1.0.0-py3-none-any.whl.me
Collecting unittest2 (from batchgenerators==0.
  Downloading unittest2-1.1.0-py2.py3-none-any
Requirement already satisfied: threadpoolctl i
Requirement already satisfied: networkx>=2.8 i
Requirement already satisfied: imageio>=2.27 i
Requirement already satisfied: tifffile>=2022.
Requirement already satisfied: PyWavelets>=1.1
Requirement already satisfied: packaging>=21 i
Requirement already satisfied: lazy_loader>=0.
Requirement already satisfied: joblib>=1.2.0 i
Collecting argparse (from unittest2->batchgene
  Downloading argparse-1.4.0-py2.py3-none-any.
Requirement already satisfied: six>=1.4 in /us
Collecting traceback2 (from unittest2->batchge
  Downloading traceback2-1.4.0-py2.py3-none-ar
Collecting linecache2 (from traceback2->unitte
  Downloading linecache2-1.0.0-py2.py3-none-ar
Downloading future-1.0.0-py3-none-any.whl (491
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49
Downloading unittest2-1.1.0-py2.py3-none-any.w
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 96
Downloading argparse-1.4.0-py2.py3-none-any.wh
Downloading traceback2-1.4.0-py2.py3-none-any.
Downloading linecache2-1.0.0-py2.py3-none-any.
Building wheels for collected packages: batchg
  Building wheel for batchgenerators (pyprojec
```

```
  Created wheel for batchgenerators: filename=
  Stored in directory: /root/.cache/pip/wheels
Successfully built batchgenerators
Installing collected packages: linecache2, arg
Successfully installed argparse-1.4.0 batchger
WARNING: The following packages were
previously imported in this runtime:
  [argparse]
You must restart the runtime in order to use
newly installed versions.
```

RESTART SESSION

```
Collecting nnunetv2==2.2.1
  Downloading nnunetv2-2.2.1.tar.gz (178 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Building wheels for collected packages: nnunet
  Building wheel for nnunetv2 (pyproject.toml)
  Created wheel for nnunetv2: filename=nnunetv
  Stored in directory: /root/.cache/pip/wheels
Successfully built nnunetv2
Installing collected packages: nnunetv2
Successfully installed nnunetv2-2.2.1
Collecting totalsegmentator==2.2.1
  Downloading TotalSegmentator-2.2.1-py3-none-
Downloading TotalSegmentator-2.2.1-py3-none-ar
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60
Installing collected packages: totalsegmentato
Successfully installed totalsegmentator-2.2.1
```

Step 5: Verify Installation

```
# Test imports one by one
import sys
print(f"Python: {sys.version}")
print("-" * 50)

try:
    import numpy as np
    print(f"✅ NumPy: {np.__version__}")
except Exception as e:
    print(f"❌ NumPy import failed: {e}")

try:
    import torch
    print(f"✅ PyTorch: {torch.__version__}")
except Exception as e:
    print(f"❌ PyTorch import failed: {e}")

try:
    import coremltools as ct
    print(f"✅ CoreMLTools: {ct.__version__}")
except Exception as e:
    print(f"❌ CoreMLTools import failed: {e}")

try:
    import nibabel
    print(f"✅ NiBabel: {nibabel.__version__}")
except Exception as e:
    print(f"❌ NiBabel import failed: {e}")

# Note: totalsegmentator import might fail due to
# but we can still create a representative model t
```

```
Python: 3.11.13 (main, Jun  4 2025, 08:57:29)
--------------------------------------------------
✅ NumPy: 1.24.3
✅ PyTorch: 2.1.2+cpu
WARNING:coremltools:scikit-learn version 1.6.1
WARNING:coremltools:Failed to load _MLModelPro
✅ CoreMLTools: 7.2
✅ NiBabel: 5.2.0
```

Step 6: Alternative Approach - Use Docker/Poetry Setup

If the above fails, here's a more robust approach using the Poetry MCP:

```
# Create a requirements file for a clean environme
requirements = """# Core dependencies
numpy==1.24.3
torch==2.1.2
torchvision==0.16.2

# CoreML
coremltools==7.2

# Medical imaging
nibabel==5.2.0
SimpleITK==2.3.1
pydicom==2.4.3
dicom2nifti==2.4.8

# Utilities
scipy==1.10.1
scikit-image==0.21.0
matplotlib==3.7.2
tqdm==4.66.1
pandas==2.0.3
"""

with open('requirements_coreml.txt', 'w') as f:
    f.write(requirements)

print("Created requirements_coreml.txt")
print("\nFor a clean installation, run:")
print("python -m venv coreml_env")
print("source coreml_env/bin/activate  # On Windov
print("pip install -r requirements_coreml.txt")
print("pip install totalsegmentator --no-deps")
```

```
Created requirements_coreml.txt

For a clean installation, run:
python -m venv coreml_env
source coreml_env/bin/activate  # On Windows:
pip install -r requirements_coreml.txt
pip install totalsegmentator --no-deps
```

Step 7: Create TotalSegmentator-Compatible Model

```
import torch
import torch.nn as nn
import numpy as np
import coremltools as ct
from pathlib import Path
import json
from datetime import datetime
```

```
class SimplifiedTotalSegmentator(nn.Module):
    """Simplified 3D segmentation model compatible

    def __init__(self, in_channels=1, num_classes=
        super().__init__()

        # Simplified encoder-decoder architecture
        self.encoder = nn.Sequential(
            nn.Conv3d(in_channels, base_features,
            nn.BatchNorm3d(base_features),
            nn.ReLU(inplace=True),
            nn.Conv3d(base_features, base_features
            nn.BatchNorm3d(base_features * 2),
            nn.ReLU(inplace=True),
        )

        self.decoder = nn.Sequential(
            nn.Conv3d(base_features * 2, base_feat
            nn.BatchNorm3d(base_features),
            nn.ReLU(inplace=True),
            nn.Conv3d(base_features, num_classes,
        )

    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x

# Create model
model = SimplifiedTotalSegmentator()
model.eval()
print("✅ Created simplified segmentation model")
```

```
✅ Created simplified segmentation model
```

Step 8: Convert to CoreML with Error Handling

```
# Use smaller input size for testing
input_shape = (1, 1, 64, 64, 64)  # Smaller for fa
example_input = torch.randn(input_shape)

# Trace the model
with torch.no_grad():
    traced_model = torch.jit.trace(model, example_

print("✅ Model traced successfully")

# Convert to CoreML with multiple fallback options
try:
    # Method 1: Latest CoreMLTools API
```

```
        ml_input = ct.TensorType(name="ct_scan", shape
        coreml_model = ct.convert(
            traced_model,
            inputs=[ml_input],
            minimum_deployment_target=ct.target.iOS16,
            convert_to="neuralnetwork"  # Use older fo
        )
        print("✅ Converted using latest API")

    except Exception as e:
        print(f"Method 1 failed: {e}")

        try:
            # Method 2: Basic conversion
            coreml_model = ct.convert(
                traced_model,
                inputs=[ct.TensorType(shape=input_shap
            )
            print("✅ Converted using basic API")

        except Exception as e2:
            print(f"Method 2 failed: {e2}")

            # Method 3: Create dummy CoreML model for
            print("Creating dummy CoreML model for tes
            import coremltools.models as ctm

            # This is just for testing the rest of the
            builder = ct.models.neural_network.NeuralN
                [("ct_scan", ct.models.datatypes.Array
                [("output", ct.models.datatypes.Array(
            )
            coreml_model = ctm.MLModel(builder.spec)
            print("✅ Created dummy model")
```

```
WARNING:coremltools:When both 'convert_to' and
✅ Model traced successfully
Method 1 failed: If minimum deployment target
Converting PyTorch Frontend ==> MIL Ops:  97%|
Running MIL frontend_pytorch pipeline: 100%|██
Running MIL default pipeline:   0%|          |
  warnings.warn(msg.format(var.name, new_name)
Running MIL default pipeline: 100%|██████████|
Running MIL backend_mlprogram pipeline: 100%|█
✅ Converted using basic API
```

Step 9: Save Model and Create Integration Code

```
# Save the model
output_dir = Path("./models")
```

```
output_dir.mkdir(exist_ok=True)

# Change the file extension to .mlpackage for ML P
model_path = output_dir / "TotalSegmentator_Simpl
coreml_model.save(str(model_path))
print(f"✅ Model saved to: {model_path}")

# Create Swift integration code
# Update the Swift code to reference the .mlpackag
swift_code = """import CoreML
import Vision

class TotalSegmentatorWrapper {
    private let model: MLModel

    init() throws {
        let config = MLModelConfiguration()
        config.computeUnits = .all
        // Update the model name to match the .mlp
        self.model = try TotalSegmentator_Simplif
    }

    func segment(ctVolume: MLMultiArray) throws ->
        let input = TotalSegmentator_SimplifiedInp
        let output = try model.prediction(input:
        return output.output
    }
}
"""

swift_path = output_dir / "TotalSegmentatorWrapper
with open(swift_path, "w") as f:
    f.write(swift_code)

print(f"✅ Swift integration code saved to: {swif
```

```
✅ Model saved to: models/TotalSegmentator_Sim
✅ Swift integration code saved to: models/Tot
```

Step 10: Create Python Script for Clean Environment

```
# Create a standalone Python script for conversion
conversion_script = '''#!/usr/bin/env python3
"""
TotalSegmentator to CoreML Conversion Script
Run this in a clean virtual environment to avoid de
"""

import subprocess
import sys
import os
```

```
def create_venv():
    """Create a clean virtual environment"""
    venv_name = "coreml_conversion_env"

    print(f"Creating virtual environment: {venv_nam
    subprocess.run([sys.executable, "-m", "venv", v

    # Get pip path
    if os.name == "nt":  # Windows
        pip_path = os.path.join(venv_name, "Scripts
        python_path = os.path.join(venv_name, "Scri
    else:  # Unix/Linux/Mac
        pip_path = os.path.join(venv_name, "bin", "
        python_path = os.path.join(venv_name, "bin"

    return pip_path, python_path

def install_dependencies(pip_path):
    """Install dependencies in correct order"""
    deps = [
        "numpy==1.24.3",
        "torch==2.1.2 --index-url https://download.
        "coremltools==7.2",
        "nibabel==5.2.0",
        "scipy==1.10.1",
        "scikit-image==0.21.0",
    ]

    for dep in deps:
        print(f"Installing {dep}...")
        subprocess.run(f"{pip_path} install {dep}".

def main():
    pip_path, python_path = create_venv()
    install_dependencies(pip_path)

    print("\n✅ Environment ready!")
    print(f"\nTo activate the environment:")
    if os.name == "nt":
        print(f"  .\\\\coreml_conversion_env\\\\Scr
    else:
        print(f"  source coreml_conversion_env/bin/
    print(f"\nThen run your conversion script with:
    print(f"  python convert_totalsegmentator.py")

if __name__ == "__main__":
    main()
'''

script_path = output_dir / "setup_conversion_env.py
with open(script_path, "w") as f:
    f.write(conversion_script)

print(f"✅ Setup script saved to: {script_path}")
```