# Requirements.txt for Master’s Final Project: Unsupervised vs Supervised Learning for Industrial Anomaly Detection

# Core Python Libraries  
python>=3.7  
pip>=21.0  
  
# Deep Learning Frameworks  
torch>=1.9.0  
torchvision>=0.10.0  
tensorflow>=2.6.0  
keras>=2.6.0  
  
# Numerical and Scientific Computing  
numpy>=1.19.2  
scipy>=1.7.0  
pandas>=1.3.0  
scikit-learn>=1.0.0  
  
# Image Processing and Computer Vision  
Pillow>=8.3.0  
opencv-python>=4.5.0  
imageio>=2.15.0  
scikit-image>=0.18.0  
  
# Visualization and Plotting  
matplotlib>=3.4.0  
seaborn>=0.11.0  
plotly>=5.0.0  
tqdm>=4.62.0  
  
# Data Handling and Utilities  
h5py>=3.1.0  
pyyaml>=5.4.0  
joblib>=1.0.0  
tables>=3.6.0  
  
# GPU Support (if using CUDA)  
# Uncomment the appropriate line based on your CUDA version  
# torch-audio>=0.9.0 # For CUDA 11.1  
# torch-audio>=0.9.0+cu111 # For CUDA 11.1  
  
# Advanced ML and Evaluation Metrics  
xgboost>=1.5.0  
lightgbm>=3.3.0  
optuna>=2.10.0 # For hyperparameter optimization  
  
# Model Explainability and Visualization  
captum>=0.4.0 # For model interpretability in PyTorch  
shap>=0.40.0 # For SHAP values  
grad-cam>=1.4.0 # For Grad-CAM visualizations  
  
# Specialized Libraries for t-SNE and Feature Visualization  
multicore-tsne>=0.1 # For faster t-SNE computation  
umap-learn>=0.5.0 # For UMAP dimensionality reduction  
  
# Data Augmentation  
albumentations>=1.1.0 # Advanced image augmentation  
imgaug>=0.4.0 # Another image augmentation library  
  
# Development and Testing  
pytest>=6.2.0  
black>=21.0.0 # Code formatting  
flake8>=3.9.0 # Code linting  
jupyter>=1.0.0 # For Jupyter notebooks  
ipywidgets>=7.6.0 # Interactive widgets in notebooks  
  
# Documentation  
sphinx>=4.0.0 # Documentation generation  
sphinx-rtd-theme>=0.5.0 # Theme for documentation  
  
# Utilities  
argparse # For command-line argument parsing (built-in)  
pathlib # For path manipulation (built-in)  
logging # For logging (built-in)  
datetime # For timestamp handling (built-in)  
json # For JSON handling (built-in)  
pickle # For object serialization (built-in)

## Additional System Requirements

### Hardware Requirements

* GPU: NVIDIA GPU with at least 8GB VRAM (Tesla T4 or equivalent recommended)
* RAM: 16GB minimum (32GB recommended)
* Storage: SSD with at least 50GB free space
* CPU: Multi-core processor (4+ cores recommended)

### Software Requirements

* Operating System: Linux (Ubuntu 18.04+ recommended) or Windows 10/11 with WSL2
* CUDA Toolkit: 11.1 or higher (if using NVIDIA GPU)
* cuDNN: 8.0 or higher (if using NVIDIA GPU)

### Dataset Requirements

* MVTec AD Dataset (available at https://www.mvtec.com/company/research/datasets/mvtec-ad)
* Approximately 10GB of storage space for the dataset

## Setup Instructions

1. Create a virtual environment:

python -m venv anomaly\_detection\_env  
source anomaly\_detection\_env/bin/activate # On Linux/Mac  
# or  
anomaly\_detection\_env\Scripts\activate # On Windows

1. Install the requirements:

pip install -r requirements.txt

1. Verify PyTorch CUDA support:

import torch  
print(torch.cuda.is\_available())

1. Verify TensorFlow GPU support:

import tensorflow as tf  
print("GPU Available: ", tf.test.is\_gpu\_available())

## Project Structure Notes

The project involves comparing supervised (CNN-based) and unsupervised (SimCLR-based) approaches for industrial anomaly detection. The requirements above cover:

1. **Deep Learning Frameworks**: Both PyTorch (for SimCLR) and TensorFlow/Keras (for CNN)
2. **Data Processing**: Libraries for handling the MVTec AD dataset
3. **Model Evaluation**: Metrics and visualization tools for comprehensive analysis
4. **Advanced Techniques**: Libraries for data augmentation, explainability, and feature visualization
5. **Development Tools**: Code formatting, testing, and documentation tools

This setup ensures you can implement both approaches and conduct a thorough comparison to demonstrate that unsupervised models, while potentially more computationally intensive, can achieve superior performance in anomaly detection tasks.