Wildfire Analysis Pipeline

A complete machine learning pipeline for wildfire detection and confidence prediction using VIIRS fire data, ERA5 meteorological data, and topographic features.

🎯 Features

- **Multi-Source Data Integration**: Combines VIIRS fire detections, ERA5 wind data, and USGS 3DEP elevation data
- **Transformer Model**: State-of-the-art attention-based neural network for fire confidence prediction
- **Random Forest Baseline**: Traditional ML baseline for comparison
- **Modular Architecture**: Clean, reusable components for data extraction, training, and prediction
- **Interactive Visualization**: Folium-based maps with fire points, slope overlays, and wind vectors

📋 Requirements

Python Version

- Python 3.8 or higher

Required Packages

```bash
pip install pandas numpy torch xarray geopandas rasterio richdem scikit-learn folium netCDF4 cdsapi requests apscheduler
```

Data Files

Place these files in your project directory:

1. `custom_date_fires.geojson` - VIIRS fire detection data
2. `era5_wind_la.nc` - ERA5 wind data (u10, v10 components)
3. `USGS3DEP_30m_33.5_34.5_-119.0_-118.0.tif` - DEM elevation data

🚀 Quick Start

1. Installation

```bash
```

```
# Clone or download the repository
cd wildfire-analysis

# Install dependencies
pip install -r requirements.txt

# Create API keys file (if needed)
cat > VIIRS_API_keys.py << EOF
MAP_KEY = "your_firms_api_key"
OPEN_TOPOGRAPHY_MAP_KEY = "your_opentopo_api_key"
EOF
```

2. Run Tests

```bash
# Test all components
python test_script.py
```

Expected output:

```
✓ Package Imports      - PASSED
✓ Data Files           - PASSED
✓ Data Extraction      - PASSED
✓ Transformer Model    - PASSED
✓ Training Pipeline    - PASSED
✓ Full Pipeline        - PASSED

Total: 6/6 tests passed
🎉 All tests passed! Your pipeline is ready to use.
```

3. Run Full Pipeline

```bash
# Run complete pipeline with transformer model
python main_pipeline.py --step full --model transformer

# Or use Random Forest (faster)
python main_pipeline.py --step full --model random_forest
```

4. Run Individual Steps

bash

```bash
# Extract features only
python main_pipeline.py --step extract

# Train model only (uses cached features)
python main_pipeline.py --step train --model transformer

# Generate predictions only
python main_pipeline.py --step predict
```

📁 Project Structure

```
wildfire-analysis/
├── data_extraction.py         # Unified data extraction module
├── config_module.py           # Configuration management
├── transformer_model.py       # Transformer model implementation
├── main_pipeline.py           # Main pipeline orchestration
├── test_script.py             # Testing suite
├── visualization.py           # Visualization utilities
├── VIIRS_API_keys.py          # API keys (create this)
│
├── outputs/                   # Generated outputs
│   ├── wildfire_features.csv
│   ├── wildfire_predictions.csv
│   ├── transformer_model.pth
│   └── random_forest_model.pkl
│
└── data/                      # Input data files
    ├── custom_date_fires.geojson
    ├── era5_wind_la.nc
    └── USGS3DEP_30m_*.tif
```

🔧 Configuration

Edit `config_module.py` to customize:

python

```python
```

```
# Geographic bounds
bounds.lat_min = 33.5
bounds.lat_max = 34.5
bounds.lon_min = -119.0
bounds.lon_max = -118.0

# Model parameters
model.d_model = 128
model.nhead = 8
model.num_layers = 4
model.n_estimators = 200  # For Random Forest
```

📊 Output Files

1. `wildfire_features.csv`

Extracted features for each fire detection:

- `latitude`, `longitude`: Fire location
- `frp`: Fire Radiative Power (MW)
- `u10`, `v10`: Wind components at 10m (m/s)
- `elevation`: Terrain elevation (m)
- `slope`: Terrain slope (degrees)
- `confidence`: VIIRS confidence level (l/n/h)
- `confidence_num`: Numerical confidence (0/1/2)

2. `wildfire_predictions.csv`

Predictions added to features:

- `predicted_confidence`: Numerical prediction (0/1/2)
- `predicted_confidence_label`: Label prediction (l/n/h)

3. Model Checkpoints

- `transformer_model.pth`: Trained transformer weights + scaler
- `random_forest_model.pkl`: Trained RF model

🎓 Model Architecture

Transformer Model

```
Input (5 features)
  ↓
Linear Projection → d_model dimensions
  ↓
Positional Encoding
  ↓
Transformer Encoder (4 layers, 8 heads)
  ↓
Classification Head (FC → ReLU → Dropout → FC)
  ↓
Output (3 classes: low/nominal/high)
```

Hyperparameters:

- `d_model`: 128
- `nhead`: 8 attention heads
- `num_layers`: 4 transformer layers
- `dim_feedforward`: 512
- `dropout`: 0.1

Random Forest Baseline

- `n_estimators`: 200 trees
- `max_depth`: 15
- `min_samples_split`: 5
- `class_weight`: balanced

📈 Performance Metrics

Example results on LA fires dataset:

Model	Accuracy	Notes
Random Forest	86%	Fast baseline, good feature importance
Transformer	~85-90%	Captures complex patterns, slower training

Key Features by Importance:

1. FRP (Fire Radiative Power) - 40.8%
2. Elevation - 26.6%
3. Slope - 24.8%
4. Wind components - ~8%

🗺️ Visualization

Generate interactive maps:

```python
from visualization import create_interactive_map

# Create map with actual confidence
map1 = create_interactive_map(
    features_df,
    confidence_column='confidence_num',
    output_file='actual_confidence_map.html'
)

# Create map with predictions
map2 = create_interactive_map(
    features_df,
    confidence_column='predicted_confidence',
    output_file='predicted_confidence_map.html'
)
```

Features:

- Fire points colored by confidence level
- Slope overlay (terrain steepness)
- Wind vectors showing direction and magnitude
- Interactive popups with detailed info
- Layer toggle for actual vs predicted

🔬 Advanced Usage

Custom Training Loop

```python
```

```python
from data_extraction import WildfireDataExtractor
from transformer_model import WildfireTransformer, WildfireModelTrainer

# Extract data
extractor = WildfireDataExtractor(
    viirs_path='custom_date_fires.geojson',
    era5_path='era5_wind_la.nc',
    dem_path='USGS3DEP_30m_33.5_34.5_-119.0_-118.0.tif'
)
extractor.load_all_data()
extractor.clip_to_dem_bounds()
features_df = extractor.extract_features()

# Train transformer
model = WildfireTransformer(input_dim=5, d_model=128)
trainer = WildfireModelTrainer(model)
train_loader, val_loader = trainer.prepare_data(
    features_df,
    ['frp', 'u10', 'v10', 'elevation', 'slope']
)
trainer.train(train_loader, val_loader, num_epochs=50)

# Make predictions
predictions = trainer.predict(features_df[feature_cols].values)
```

Batch Prediction API

python

```python
def predict_fire_confidence(lat, lon, frp, u10, v10, elevation, slope):
    """Predict confidence for a single fire detection"""
    features = np.array([[frp, u10, v10, elevation, slope]])
    prediction = trainer.predict(features)
    return {0: 'low', 1: 'nominal', 2: 'high'}[prediction[0]]
```

🐛 Troubleshooting

Common Issues

1. "Module not found" errors

bash

```
pip install --upgrade pip
pip install -r requirements.txt
```

2. "File not found" errors

- Ensure all data files are in the project directory
- Check file names match exactly (case-sensitive)

3. Memory errors with large datasets

- Reduce batch size in `config_module.py`
- Process data in chunks
- Use smaller model dimensions

4. CUDA/GPU errors

- Model automatically uses CPU if CUDA unavailable
- For GPU: `pip install torch --index-url https://download.pytorch.org/whl/cu118`

5. richdem installation issues

```bash
# On Linux/Mac:
conda install -c conda-forge richdem

# Or build from source:
pip install richdem --no-binary richdem
```

📝 Citation

If you use this pipeline in your research, please cite:

```bibtex
@software{wildfire_analysis_pipeline,
  title = {Wildfire Analysis Pipeline},
  author = {Your Name},
  year = {2025},
  url = {https://github.com/yourusername/wildfire-analysis}
}
```

🤝 Contributing

Contributions welcome! Areas for improvement:

- Additional data sources (weather stations, vegetation indices)
- Model architectures (CNNs for spatial context, LSTMs for temporal)
- Real-time prediction API
- Web dashboard
- Multi-region support

📄 License

🔗 Resources

- VIIRS Fire Data
- ERA5 Climate Data
- USGS 3DEP
- Transformer Paper

📞 Support

For questions or issues:

- Open an issue on GitHub
- Check documentation at `docs/`
- Contact: your.email@example.com