01_inspect_data

December 29, 2025

1 01 - Inspect AI4Arctic RTT Data

Goal: Understand the data structure before building the baseline.

- What variables are available?
- What are the SAR channels?
- What are the target labels?
- What values need to be ignored (255)?

[1]:
```
import glob
import numpy as np
import xarray as xr
import matplotlib.pyplot as plt

# Set data path
DATA_ROOT = "../data/ai4arctic_hugging face"
TRAIN_DIR = f"{DATA_ROOT}/train"
TEST_DIR = f"{DATA_ROOT}/test"
```

1.1 1. Count Files

[2]:
```
train_files = sorted(glob.glob(f"{TRAIN_DIR}/*.nc"))
test_files = sorted(glob.glob(f"{TEST_DIR}/*.nc"))

print(f"Train files: {len(train_files)}")
print(f"Test files:  {len(test_files)}")
print(f"\nFirst train file: {train_files[0].split('/')[-1]}")
```

```
Train files: 512
Test files:  40

First train file: 20180108T184332_dmi_prep.nc
```

1.2 2. Inspect One Scene

[3]:
```
# Load first scene
ds = xr.open_dataset(train_files[0])
print("Variables in dataset:")
print("=" * 70)
```

```
for name, var in ds.data_vars.items():
    arr = var.values
    print(f"{name:30s} shape={str(var.shape):20s} dtype={str(var.dtype):10s}")
```

```
Variables in dataset:
================================================================================
SIC                            shape=(5353, 5180)          dtype=uint8
SOD                            shape=(5353, 5180)          dtype=uint8
FLOE                           shape=(5353, 5180)          dtype=uint8
sar_grid2d_latitude            shape=(23, 21)              dtype=float64
sar_grid2d_longitude           shape=(23, 21)              dtype=float64
nersc_sar_primary              shape=(5353, 5180)          dtype=float32
nersc_sar_secondary            shape=(5353, 5180)          dtype=float32
sar_incidenceangle             shape=(5353, 5180)          dtype=float32
distance_map                   shape=(5353, 5180)          dtype=float32
btemp_6_9h                     shape=(214, 207)            dtype=float32
btemp_6_9v                     shape=(214, 207)            dtype=float32
btemp_7_3h                     shape=(214, 207)            dtype=float32
btemp_7_3v                     shape=(214, 207)            dtype=float32
btemp_10_7h                    shape=(214, 207)            dtype=float32
btemp_10_7v                    shape=(214, 207)            dtype=float32
btemp_18_7h                    shape=(214, 207)            dtype=float32
btemp_18_7v                    shape=(214, 207)            dtype=float32
btemp_23_8h                    shape=(214, 207)            dtype=float32
btemp_23_8v                    shape=(214, 207)            dtype=float32
btemp_36_5h                    shape=(214, 207)            dtype=float32
btemp_36_5v                    shape=(214, 207)            dtype=float32
btemp_89_0h                    shape=(214, 207)            dtype=float32
btemp_89_0v                    shape=(214, 207)            dtype=float32
u10m_rotated                   shape=(214, 207)            dtype=float32
v10m_rotated                   shape=(214, 207)            dtype=float32
t2m                            shape=(214, 207)            dtype=float32
skt                            shape=(214, 207)            dtype=float32
tcwv                           shape=(214, 207)            dtype=float32
tclw                           shape=(214, 207)            dtype=float32
```

[4]:
```
# Detailed stats for each variable
print("\nVariable Statistics:")
print("=" * 70)
for name, var in ds.data_vars.items():
    arr = var.values
    valid = arr[~np.isnan(arr)] if arr.dtype == np.float32 else arr[arr != 255]
    print(f"{name}:")
    print(f"  min={np.nanmin(arr):.3f}, max={np.nanmax(arr):.3f}, mean={np.
 ↪nanmean(arr):.3f}")
    print(f"  unique values: {len(np.unique(arr))}")
    print()
```

```
Variable Statistics:
======================================================================
SIC:
  min=0.000, max=255.000, mean=81.061
  unique values: 7

SOD:
  min=0.000, max=255.000, mean=176.495
  unique values: 6

FLOE:
  min=0.000, max=255.000, mean=116.372
  unique values: 5

sar_grid2d_latitude:
  min=67.297, max=71.980, mean=69.707
  unique values: 483

sar_grid2d_longitude:
  min=-28.850, max=-15.640, mean=-21.835
  unique values: 483

nersc_sar_primary:
  min=-10.875, max=4.907, mean=0.281
  unique values: 7413662

nersc_sar_secondary:
  min=-8.898, max=7.376, mean=0.165
  unique values: 4946366

sar_incidenceangle:
  min=-1.747, max=1.504, mean=0.027
  unique values: 16146382

distance_map:
  min=-1.576, max=1.025, mean=-0.084
  unique values: 40

btemp_6_9h:
  min=-1.095, max=1.504, mean=0.287
  unique values: 14276

btemp_6_9v:
  min=-1.145, max=1.416, mean=0.232
  unique values: 10436

btemp_7_3h:
```

```
  min=-1.096, max=1.498, mean=0.284
  unique values: 14311

btemp_7_3v:
  min=-1.141, max=1.420, mean=0.228
  unique values: 10482

btemp_10_7h:
  min=-1.081, max=1.542, mean=0.284
  unique values: 14023

btemp_10_7v:
  min=-1.131, max=1.472, mean=0.218
  unique values: 10034

btemp_18_7h:
  min=-1.159, max=1.629, mean=0.269
  unique values: 13775

btemp_18_7v:
  min=-1.180, max=1.587, mean=0.148
  unique values: 8958

btemp_23_8h:
  min=-1.331, max=1.624, mean=0.188
  unique values: 12861

btemp_23_8v:
  min=-1.317, max=1.568, mean=0.010
  unique values: 7736

btemp_36_5h:
  min=-1.232, max=1.554, mean=0.247
  unique values: 11994

btemp_36_5v:
  min=-2.352, max=1.592, mean=-0.050
  unique values: 6168

btemp_89_0h:
  min=-2.034, max=2.014, mean=-0.095
  unique values: 9610

btemp_89_0v:
  min=-4.284, max=1.310, mean=-0.557
  unique values: 9250

u10m_rotated:
```

```
  min=-0.576, max=3.690, mean=1.573
  unique values: 44263

v10m_rotated:
  min=-3.451, max=0.573, mean=-1.011
  unique values: 44273

t2m:
  min=-2.371, max=0.140, mean=-0.919
  unique values: 42925

skt:
  min=-3.182, max=0.685, mean=-0.704
  unique values: 42603

tcwv:
  min=-1.220, max=-0.045, mean=-0.685
  unique values: 44174

tclw:
  min=-0.580, max=2.100, mean=-0.356
  unique values: 39215
```

1.3 3. Identify SAR Channels (Inputs)

[5]:
```python
# Find SAR variables
sar_vars = [k for k in ds.data_vars if 'sar' in k.lower()]
print(f"SAR variables: {sar_vars}")

# Show their shapes
for var in sar_vars:
    print(f"  {var}: {ds[var].shape}")
```

```
SAR variables: ['sar_grid2d_latitude', 'sar_grid2d_longitude',
'nersc_sar_primary', 'nersc_sar_secondary', 'sar_incidenceangle']
  sar_grid2d_latitude: (23, 21)
  sar_grid2d_longitude: (23, 21)
  nersc_sar_primary: (5353, 5180)
  nersc_sar_secondary: (5353, 5180)
  sar_incidenceangle: (5353, 5180)
```

1.4 4. Identify Target Labels

[6]:
```python
# Find target variables
all_vars = list(ds.data_vars.keys())

sod_vars = [k for k in all_vars if 'sod' in k.lower()]
```

```
sic_vars = [k for k in all_vars if 'sic' in k.lower()]
floe_vars = [k for k in all_vars if 'floe' in k.lower()]

print(f"SOD (Stage of Development): {sod_vars}")
print(f"SIC (Sea Ice Concentration): {sic_vars}")
print(f"FLOE (Floe Size): {floe_vars}")
```

```
SOD (Stage of Development): ['SOD']
SIC (Sea Ice Concentration): ['SIC']
FLOE (Floe Size): ['FLOE']
```

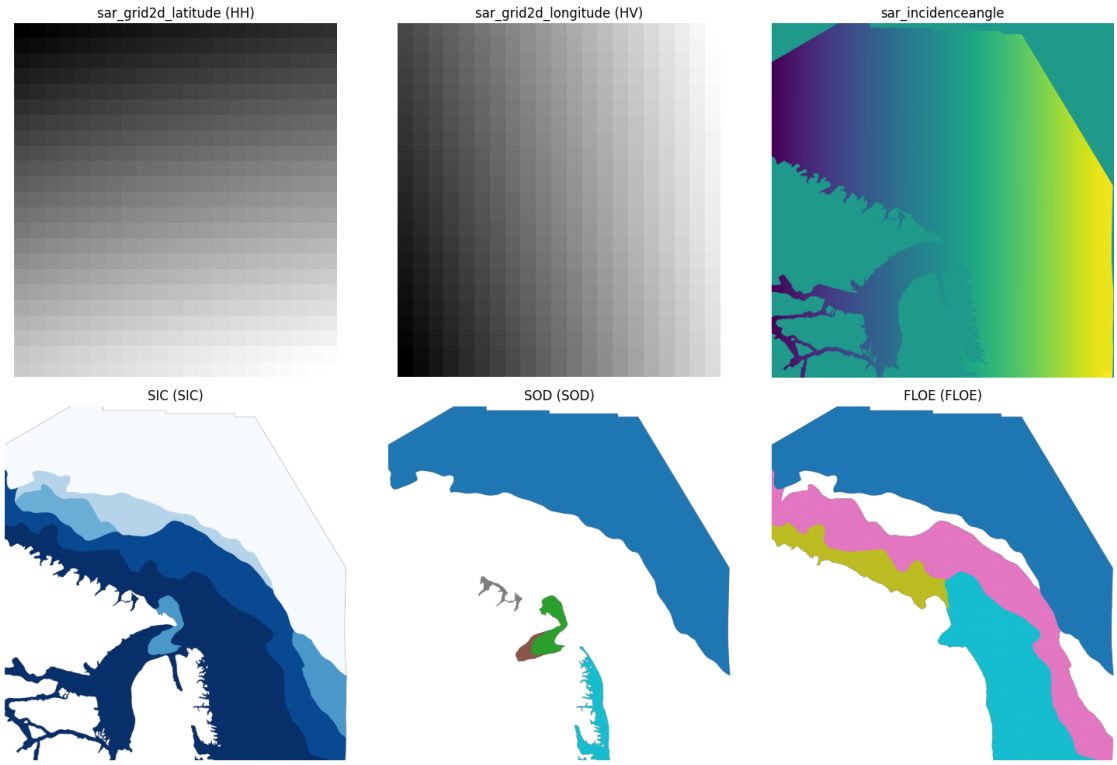
[7]:
```
# Check unique values in SOD (our baseline target)
if sod_vars:
    sod = ds[sod_vars[0]].values
    unique_sod = np.unique(sod)
    print(f"SOD unique values: {unique_sod}")
    print(f"SOD num classes (excluding 255): {len(unique_sod[unique_sod !=
 ↪255])}")
    print(f"\nPixels with value 255 (ignore): {np.mean(sod == 255)*100:.1f}%")
```

```
SOD unique values: [  0   1   2   3   4 255]
SOD num classes (excluding 255): 5

Pixels with value 255 (ignore): 69.2%
```

1.5 5. Visualize One Scene

[8]:
```
fig, axes = plt.subplots(2, 3, figsize=(15, 10))

# SAR channels
if len(sar_vars) >= 2:
    axes[0, 0].imshow(ds[sar_vars[0]].values, cmap='gray')
    axes[0, 0].set_title(f'{sar_vars[0]} (HH)')
    axes[0, 0].axis('off')

    axes[0, 1].imshow(ds[sar_vars[1]].values, cmap='gray')
    axes[0, 1].set_title(f'{sar_vars[1]} (HV)')
    axes[0, 1].axis('off')

# Incidence angle if available
inc_vars = [k for k in all_vars if 'incidence' in k.lower() or 'angle' in k.
 ↪lower()]
if inc_vars:
    axes[0, 2].imshow(ds[inc_vars[0]].values, cmap='viridis')
    axes[0, 2].set_title(f'{inc_vars[0]}')
    axes[0, 2].axis('off')

# Targets
```

```
if sic_vars:
    sic_data = ds[sic_vars[0]].values.copy().astype(float)
    sic_data[sic_data == 255] = np.nan
    axes[1, 0].imshow(sic_data, cmap='Blues')
    axes[1, 0].set_title(f'{sic_vars[0]} (SIC)')
    axes[1, 0].axis('off')

if sod_vars:
    sod_data = ds[sod_vars[0]].values.copy().astype(float)
    sod_data[sod_data == 255] = np.nan
    axes[1, 1].imshow(sod_data, cmap='tab10')
    axes[1, 1].set_title(f'{sod_vars[0]} (SOD)')
    axes[1, 1].axis('off')

if floe_vars:
    floe_data = ds[floe_vars[0]].values.copy().astype(float)
    floe_data[floe_data == 255] = np.nan
    axes[1, 2].imshow(floe_data, cmap='tab10')
    axes[1, 2].set_title(f'{floe_vars[0]} (FLOE)')
    axes[1, 2].axis('off')

plt.tight_layout()
plt.show()
```



1.6 6. Summary - Variables for Baseline

[9]:
```
print("=" * 70)
print("SUMMARY: Variables for 3-Channel SAR Baseline")
print("=" * 70)
print(f"\nINPUTS (3 channels):")
print(f"  1. {sar_vars[0] if sar_vars else 'TBD'} - SAR HH polarization")
print(f"  2. {sar_vars[1] if len(sar_vars) > 1 else 'TBD'} - SAR HV␣
 ↪polarization")
print(f"  3. {inc_vars[0] if inc_vars else 'TBD'} - Incidence angle")
print(f"\nTARGET (single task):")
print(f"  {sod_vars[0] if sod_vars else 'TBD'} - Stage of Development")
print(f"  Classes: {len(unique_sod[unique_sod != 255])} (ignore_index=255)")
print(f"\nScene shape: {ds[sar_vars[0]].shape if sar_vars else 'TBD'}")
```

```
======================================================================
SUMMARY: Variables for 3-Channel SAR Baseline
======================================================================

INPUTS (3 channels):
  1. sar_grid2d_latitude - SAR HH polarization
  2. sar_grid2d_longitude - SAR HV polarization
  3. sar_incidenceangle - Incidence angle

TARGET (single task):
  SOD - Stage of Development
  Classes: 5 (ignore_index=255)

Scene shape: (23, 21)
```

[10]:
```
# Close dataset
ds.close()
```

[ ]:
```

```