view

July 20, 2021

1 Solar Power Generation Data: View

1.1 Solar power generation and sensor data for two power plants.

Description This data has been gathered at two solar power plants in India over a 34 day period. It has two pairs of files - each pair has one power generation dataset and one sensor readings dataset. The power generation datasets are gathered at the inverter level - each inverter has multiple lines of solar panels attached to it. The sensor data is gathered at a plant level - single array of sensors optimally placed at the plant.

There are a few areas of concern at the solar power plant -

- Can we predict the power generation for next couple of days? - this allows for better grid management
- Can we identify generation profiles?
- Can we identify the need for panel cleaning/maintenance?
- Can we identify faulty or suboptimally performing equipment?

Link to source

[2]:
```
import sys

sys.path.append('../')
```

[3]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import math

from read import (
    dfGen01,
    dfGen02,
    dfWheather01,
    dfWheather02,
    sourcesGen01,
    sourcesGen02,
    groupsWeather01,
    groupsWeather02,
    sourcesDailyYield
)
```

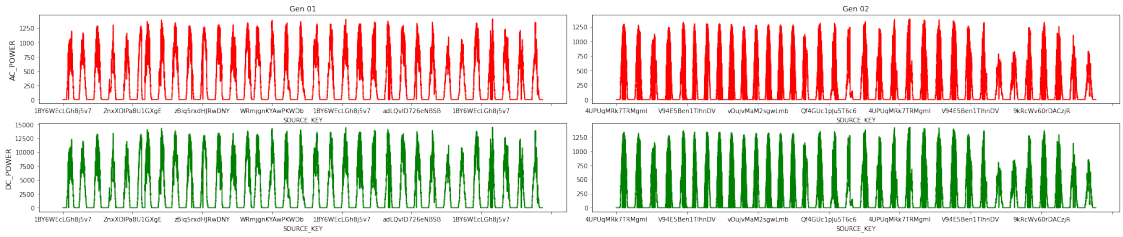
1.2 Set figure size and aspect

[3]:
```
plt.rcParams["figure.figsize"] = (21,3)
```

2 1. GEN

2.1 1.1. Initial view

[4]:
```
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(25, 5))
fig.tight_layout()

axes[0][0].set_title("Gen 01")
dfGen01['AC_POWER'].plot(color="red", ax=axes[0][0])
dfGen01['DC_POWER'].plot(color="green", ax=axes[1][0])

axes[0][1].set_title("Gen 02")
dfGen02['AC_POWER'].plot(color="red", ax=axes[0][1])
dfGen02['DC_POWER'].plot(color="green", ax=axes[1][1])

axes[0][0].set_ylabel("AC_POWER", rotation=90, size='large')
axes[1][0].set_ylabel("DC_POWER", rotation=90, size='large')

plt.show()
```



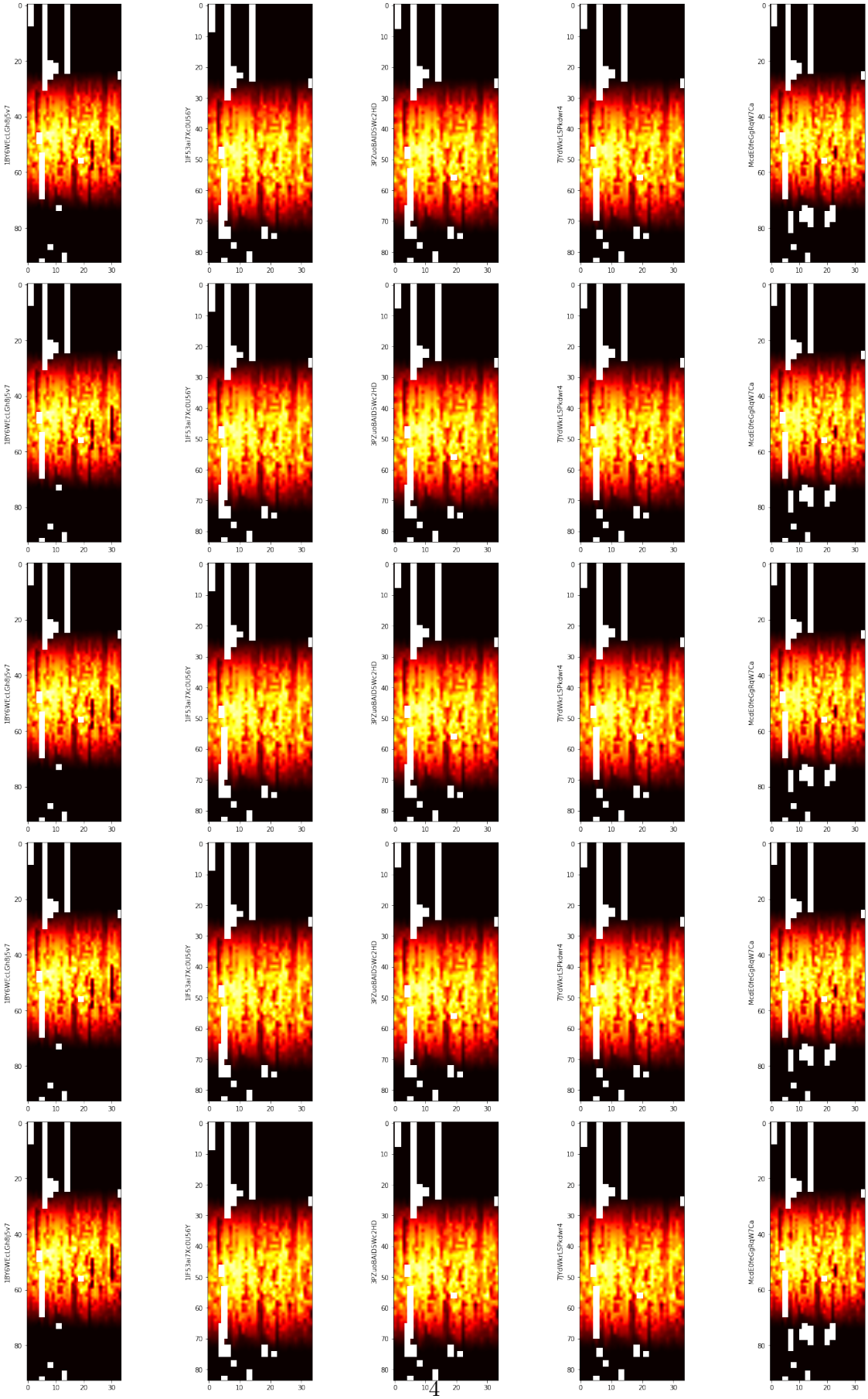
2.2 1.2. View heatmap by sources in time-range

2.2.1 GEN01

AC

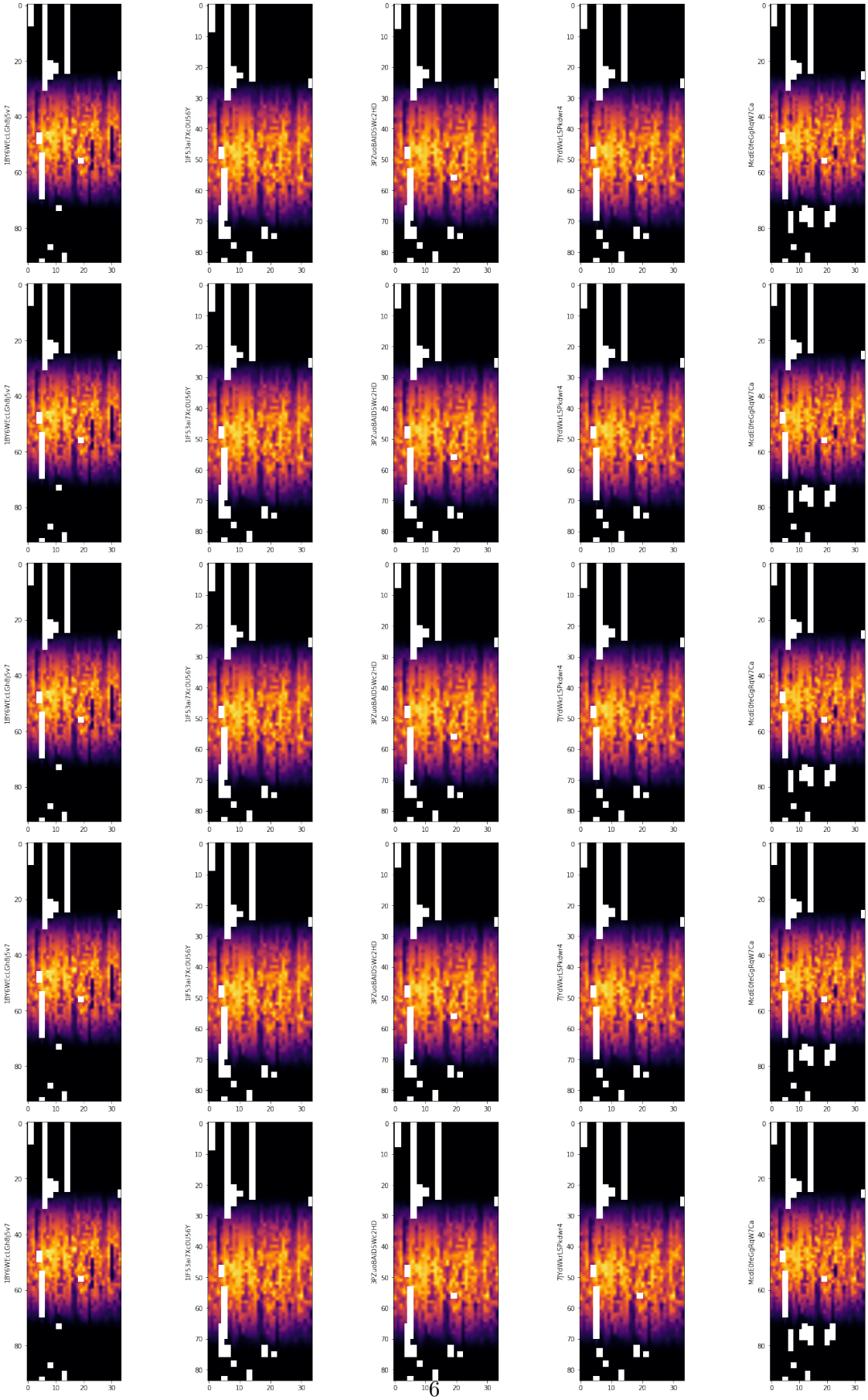
[14]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesGen01["AC_POWER"].keys()))
))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
```

```
    for ax, source in zip(axesRow, sourcesGen01["AC_POWER"].items()):
        ax.imshow(source[1], cmap="hot",  interpolation='bilinear')
        ax.set_ylabel(source[0])

plt.show()
```

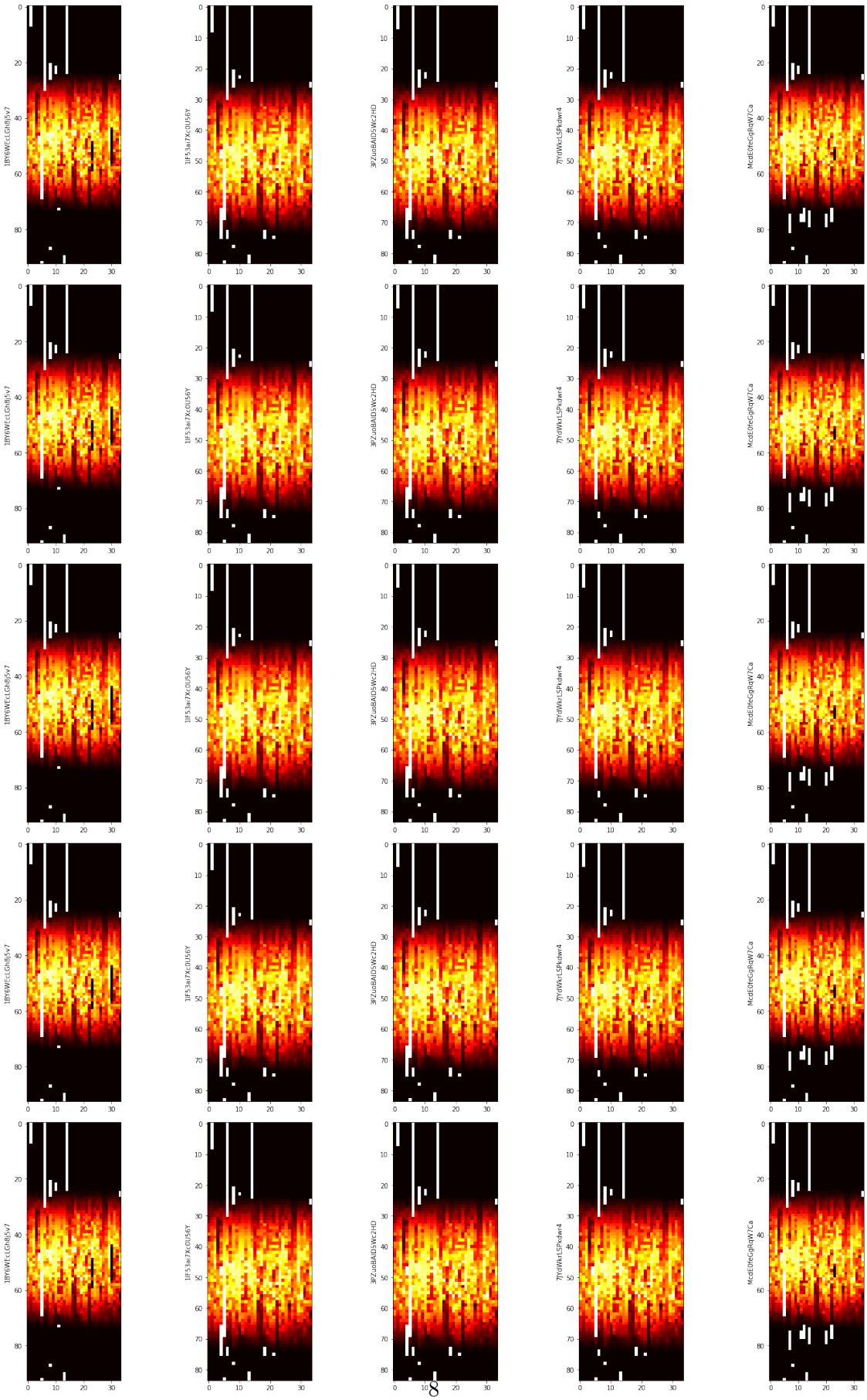

1BY6WEcLGhBj5v7
1IF53ai7XcOU56Y
3PZuoBAID5Wc2HD
7fYdWxcLSPkdwr4
McdE0feCgRqW7Ca

DC

[13]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesGen01["AC_POWER"].keys()))
))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen01["DC_POWER"].items()):
        ax.imshow(source[1], cmap="inferno", interpolation='bilinear')
        ax.set_ylabel(source[0])

plt.show()
```


1BY6WEcLGhBj5v7
1IF53ai7XcOU56Y
3PZuoBAID5Wc2HD
7fYdWxrLSPkdwr4
McdE0feCgRqW7Ca

[12]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesGen01["AC_POWER"].keys()))
))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen01["AC_POWER"].items()):
        ax.imshow(source[1], cmap="hot")
        ax.set_ylabel(source[0])

plt.show()
```


1BY6WEcLGhBj5v7
1IF53ai7XcOU56Y
3PZuoBAID5Wc2HD
7fYdWxcLSPkdwr4
McdE0feGqRqW7Ca

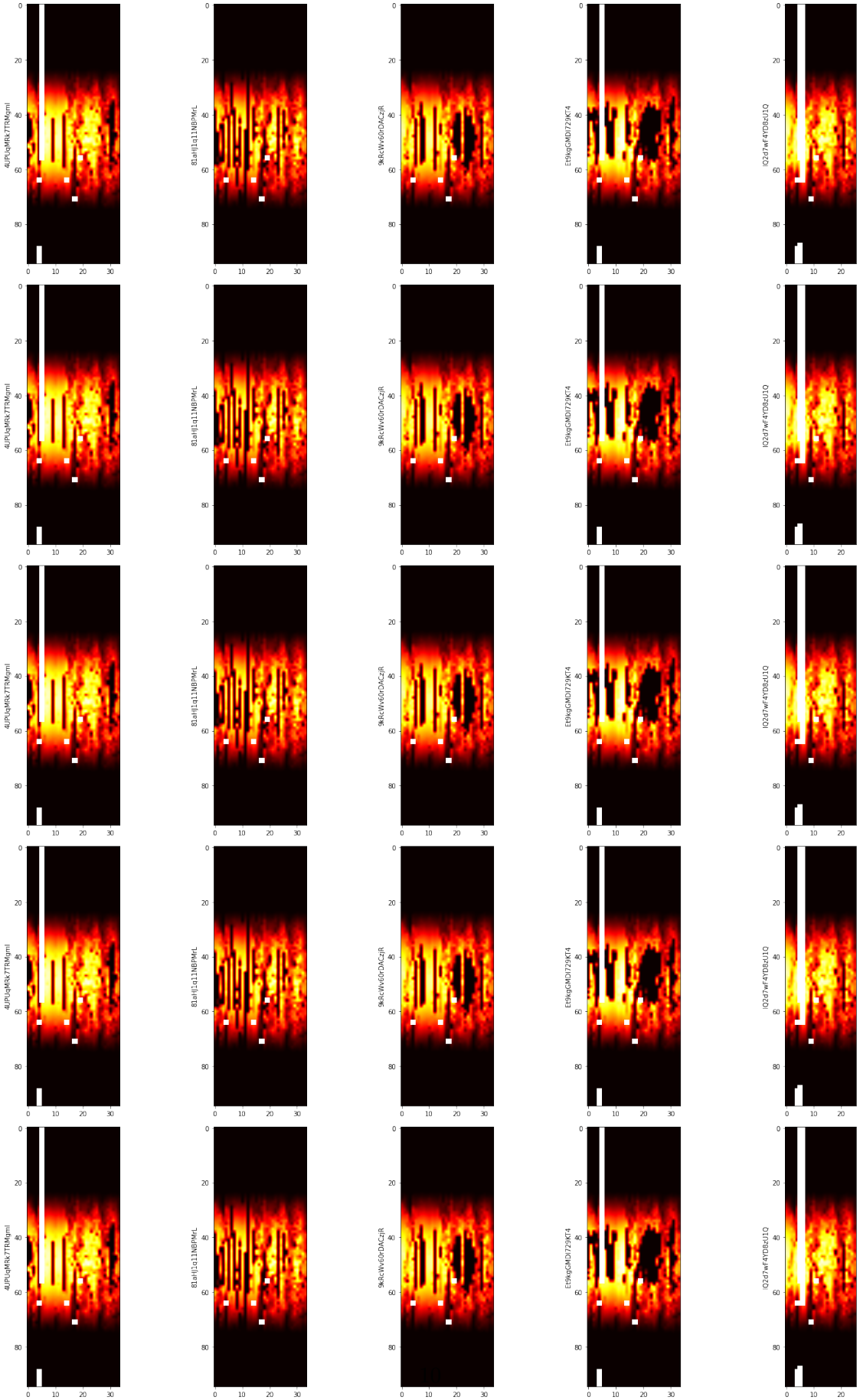
2.2.2 GEN02

The view of the DataSet Gen02 “Plant_2_Generation_Data.csv” is unusable.Shows signs that the missing values in the dataset have been replaced with 0, a valid value for measurements.This implies that the DataSet 02, even if it were treated, would aggregate incorrect information to the execution.

Dataset 02 will be discarded, which implies less data for training and testing, important for approaches such as cross-validation.

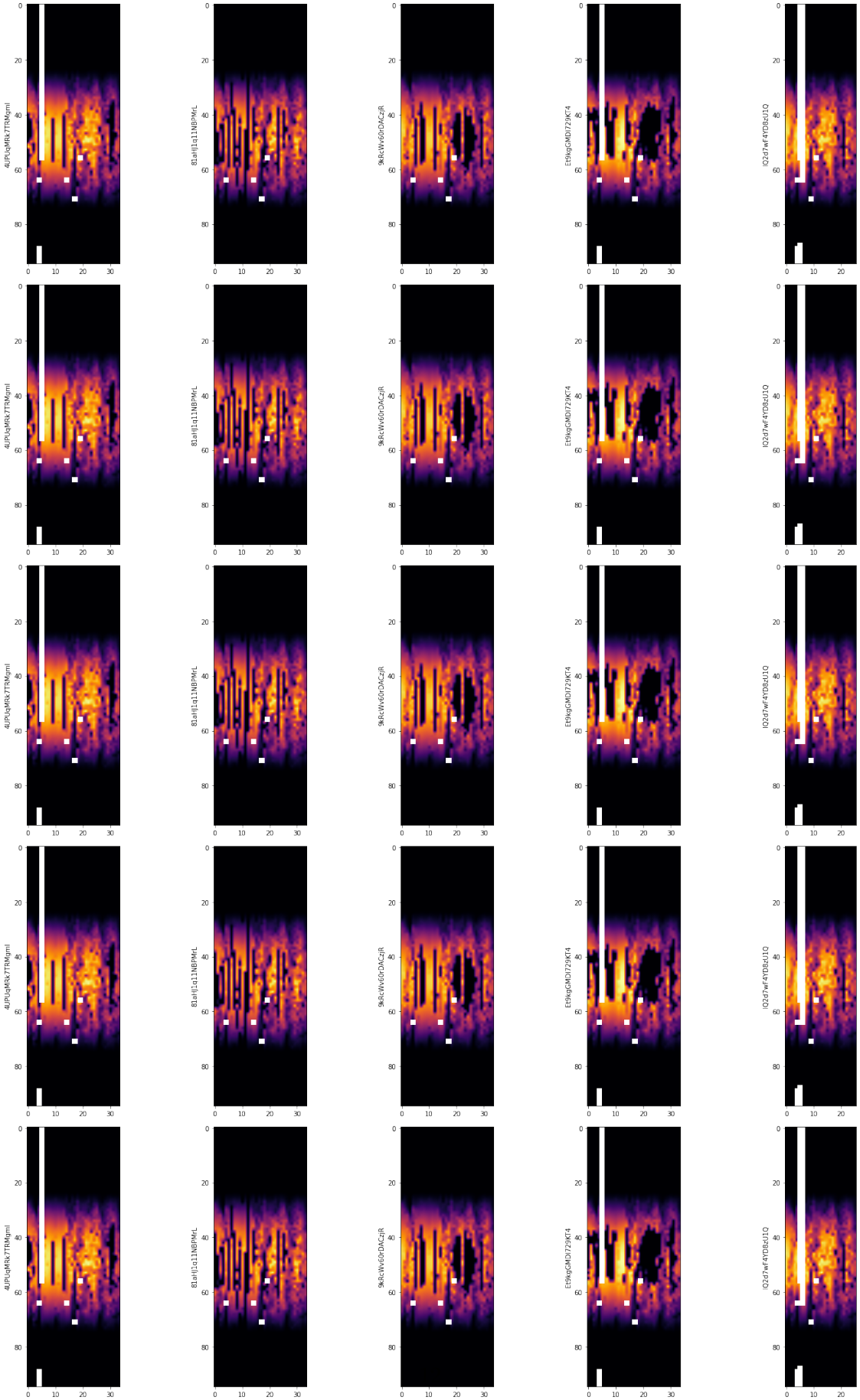
AC

[11]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesGen01["AC_POWER"].keys()))
    ))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen02["AC_POWER"].items()):
        ax.imshow(source[1], cmap="hot", interpolation="bilinear")
        ax.set_ylabel(source[0])

plt.show()
```



DC

[10]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesGen01["AC_POWER"].keys()))
    ))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen02["DC_POWER"].items()):
        ax.imshow(source[1], cmap="inferno", interpolation="bilinear")
        ax.set_ylabel(source[0])

plt.show()
```

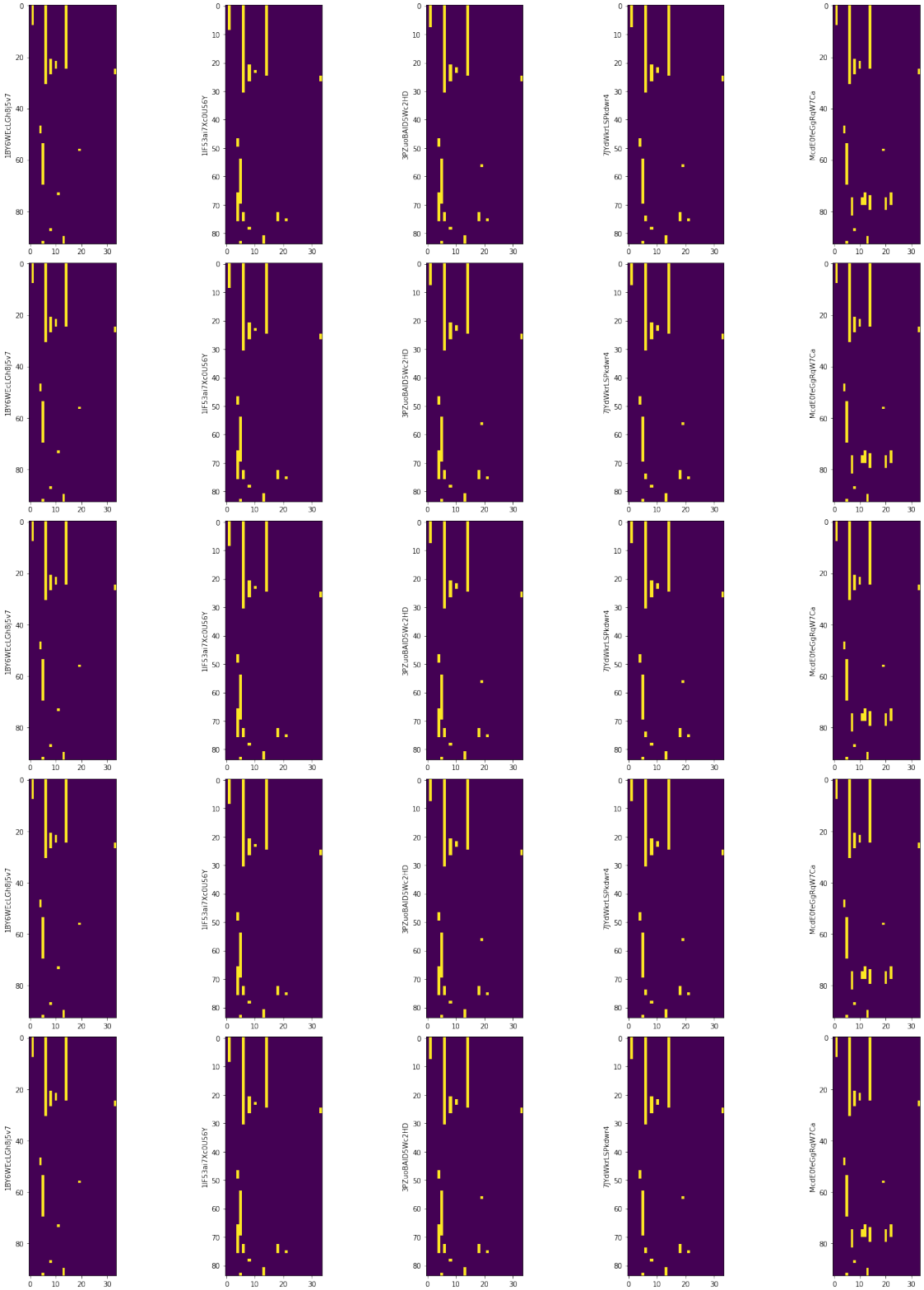

4UPUqMRk7TRMqmI
81aHJ1q11NBPMrL
9kRcWv60rDACzjR
Et9kgGMDJ729KT4
IQ2d7wF4YD8zU1Q

2.3 1.3. Visualization of missing elements

2.3.1 GEN01

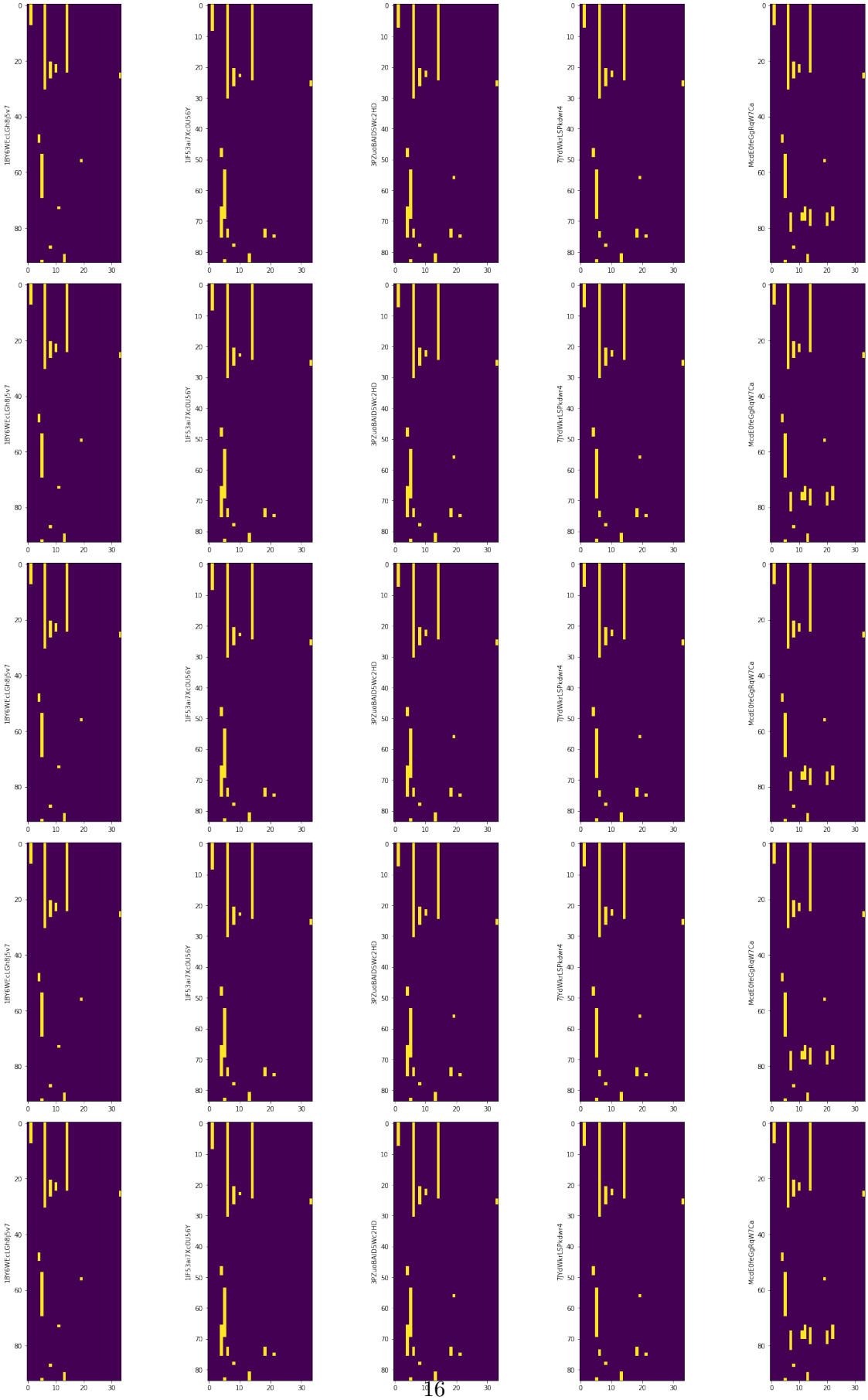
AC

[9]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesGen01["AC_POWER"].keys()))
    ))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen01["AC_POWER"].items()):
        ax.imshow(source[1].isnull().values)
        ax.set_ylabel(source[0])

plt.subplots_adjust(top=0.85)
plt.show()
```


1BY6WEcLGhBj5v7
1iF53ai7XcOU56Y
3PZuoBAlDSWc2HD
7JYdWxtLSPkdwr4
McdE0feGqRqW7Ca

DC

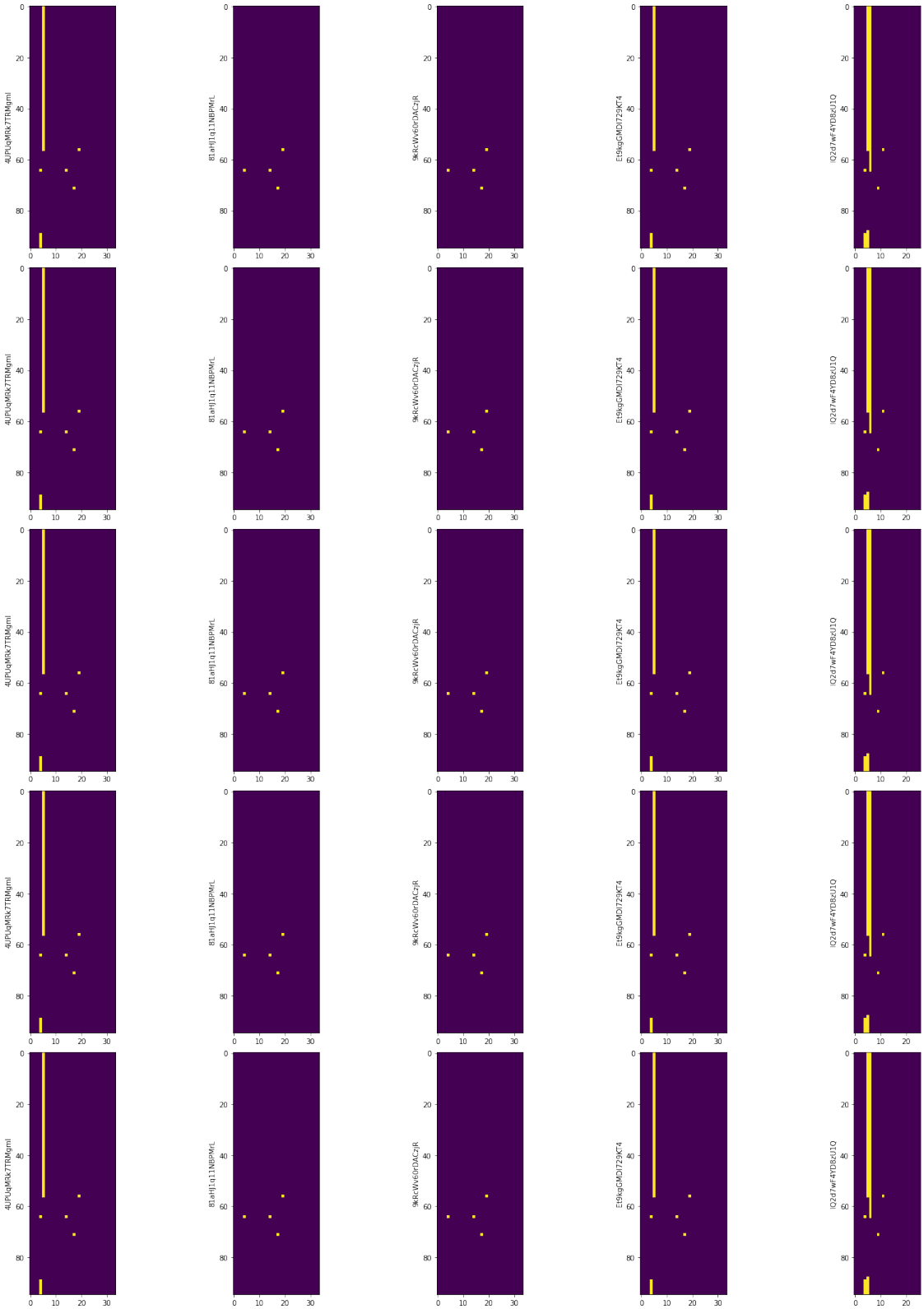
[8]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesGen01["DC_POWER"].keys()))
    ))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen01["DC_POWER"].items()):
        ax.imshow(source[1].isnull().values)
        ax.set_ylabel(source[0])

plt.show()
```


1BY6WEcLGhBj5v7
1IF53ai7XcOU56Y
3PZuoBAID5Wc2HD
7fYdWxcLSPkdwr4
McdE0feCgRqW7Ca

2.3.2 GEN02

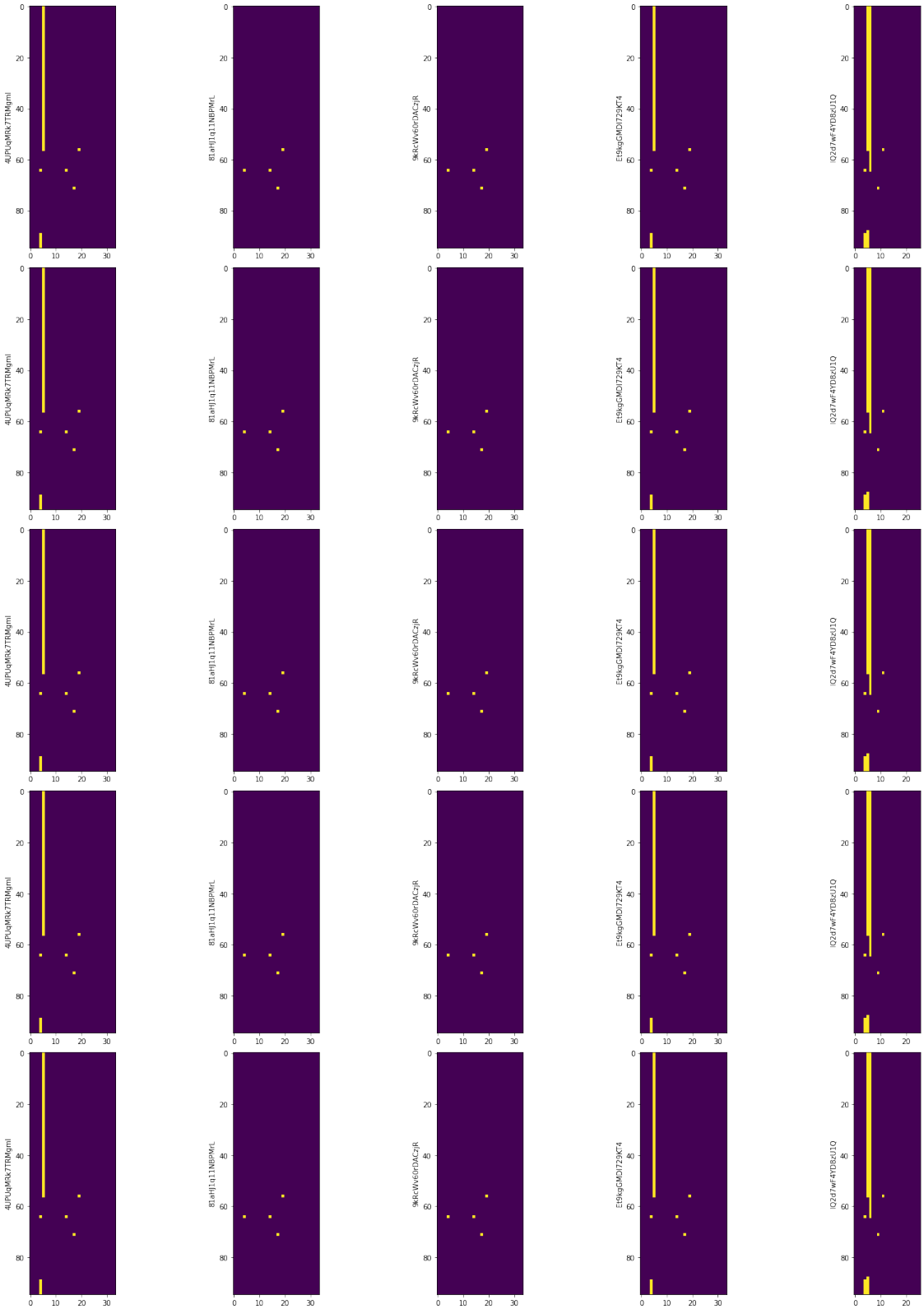
AC

[7]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesGen01["AC_POWER"].keys()))
    ))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen02["AC_POWER"].items()):
        ax.imshow(source[1].isnull().values)
        ax.set_ylabel(source[0])

plt.subplots_adjust(top=0.85)
plt.show()
```


4UPUqM0k7TRMgmI
81aHJ1q11NBPMrL
9kRcWv6OrDACzjR
Et9kgGMOJ729KC4
IQ2d7wF4YDBzJ1O

DC

[6]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesGen01["DC_POWER"].keys()))
    ))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen02["DC_POWER"].items()):
        ax.imshow(source[1].isnull().values)
        ax.set_ylabel(source[0])

plt.subplots_adjust(top=0.85)
plt.show()
```

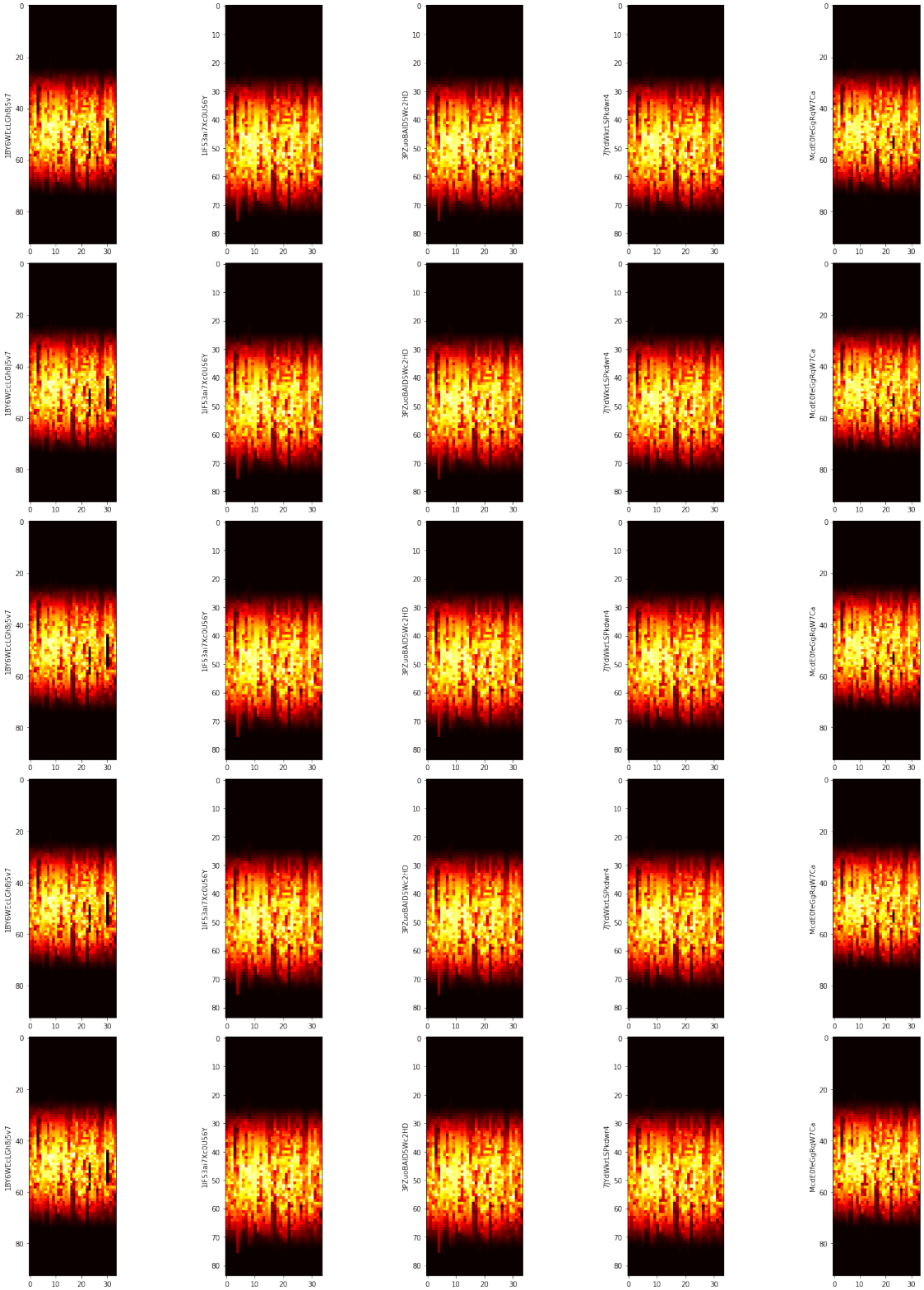


2.4 1.4. Interpolate

2.4.1 GEN01

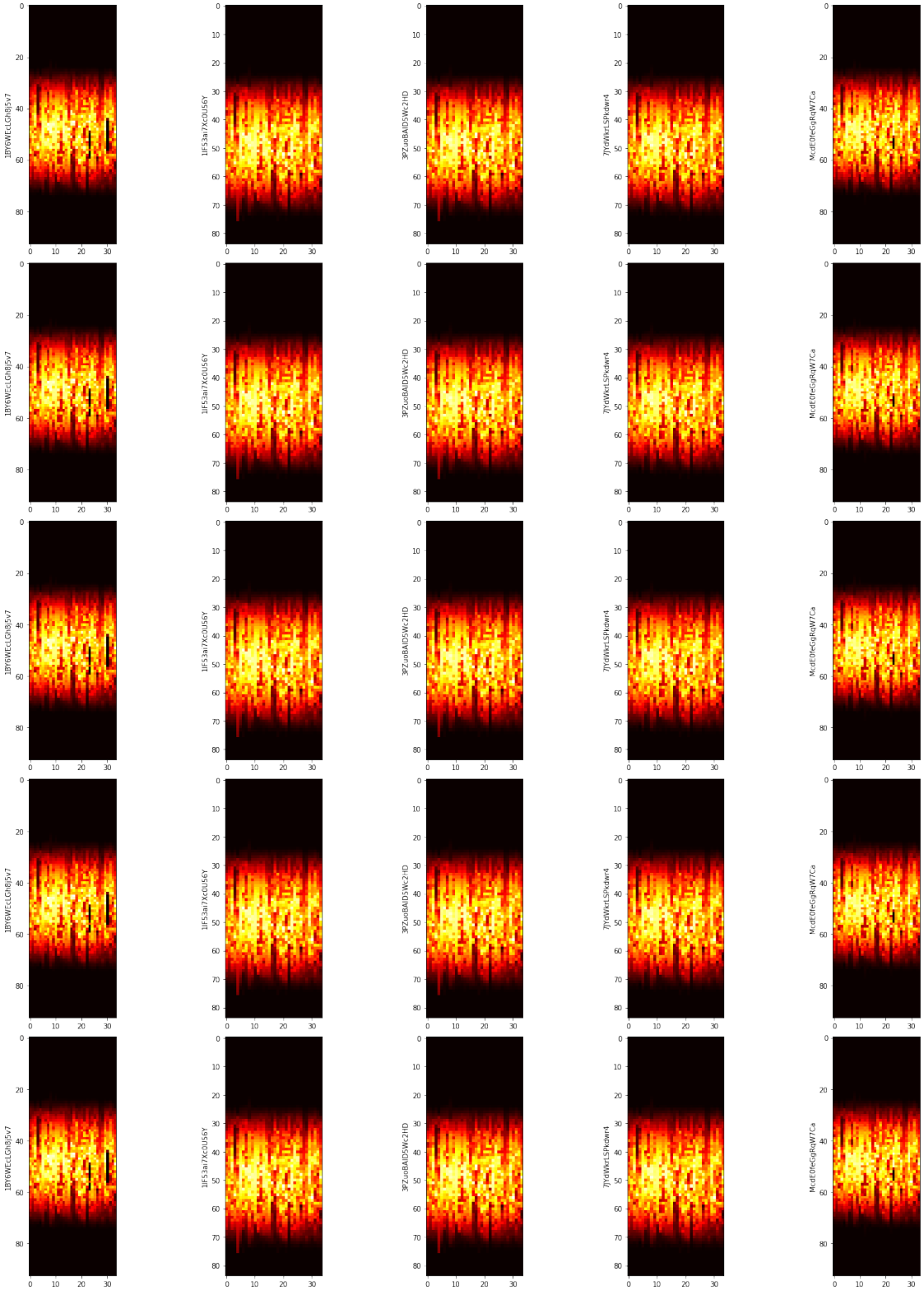
AC

[5]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesGen01["AC_POWER"].keys()))
    ))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

# Works with daily and high resolution data for interpolar given interval␣
 ↪length.

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen01["AC_POWER"].items()):
        df = source[1].copy(deep=True)
        df.index = pd.to_datetime(df.index,  format='%H:%M:%S')
        ax.imshow(df.interpolate(method="time").interpolate(method="linear",␣
 ↪axis=1), cmap="hot")
        ax.set_ylabel(source[0])

plt.subplots_adjust(top=0.85)
plt.show()
```



DC

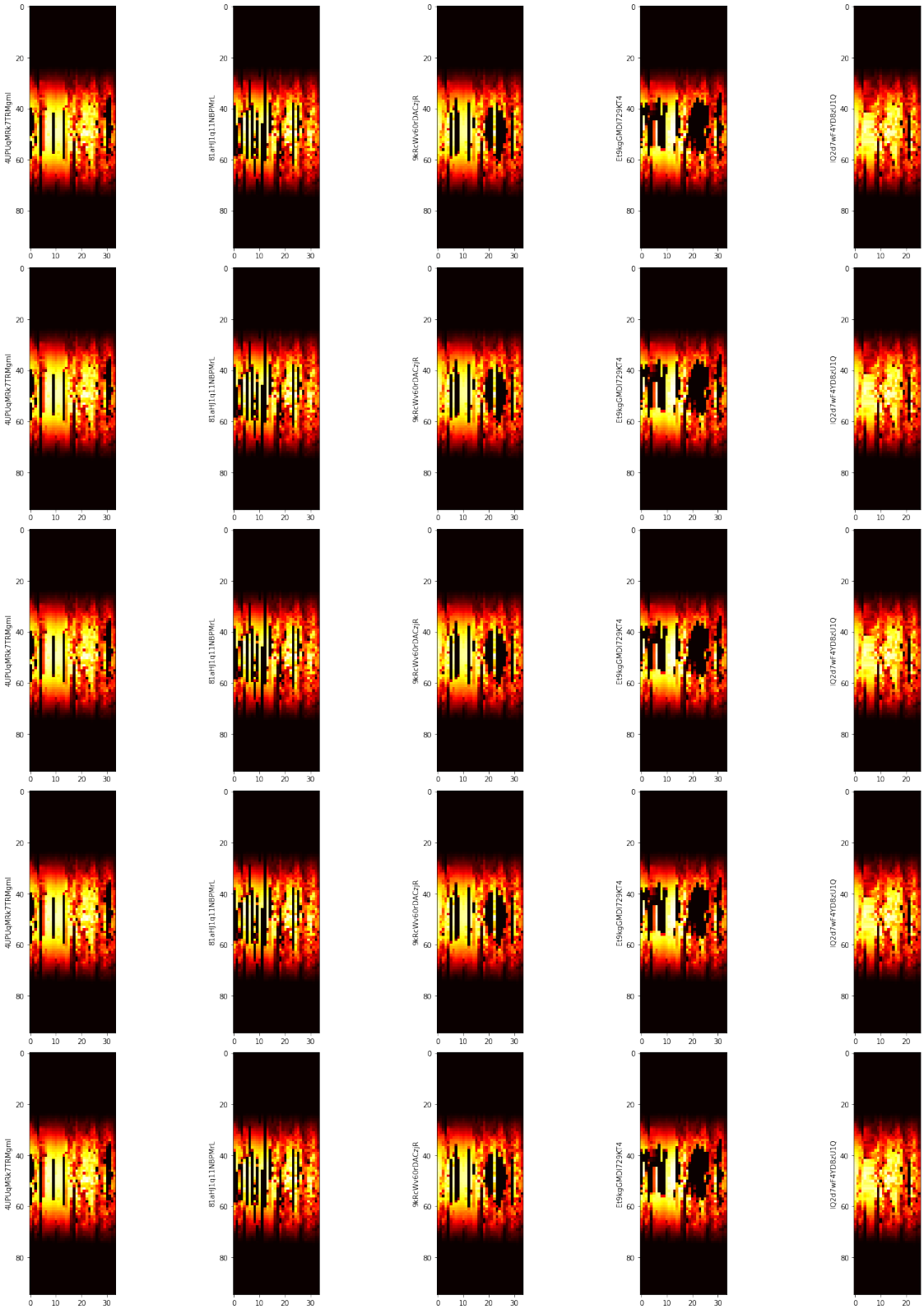
[15]:
```
rowsSquare = math.ceil(math.sqrt(len(list(sourcesGen01["DC_POWER"].keys()))))

fig, axes = plt.subplots(nrows=rowsSquare, ncols=rowsSquare, figsize=(20, 30))
fig.tight_layout()

# Works with daily and high resolution data for interpolar given interval␣
 ↪length.

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen01["DC_POWER"].items()):
        df = source[1].copy(deep=True)
        df.index = pd.to_datetime(df.index,  format='%H:%M:%S')
        ax.imshow(df.interpolate(method="time").interpolate(method="linear",␣
 ↪axis=1), cmap="hot")
        ax.set_ylabel(source[0])

plt.subplots_adjust(top=0.85)
plt.show()
```


1BY6WEcLGh8j5v7
1iF53ai7XcOU56Y
3PZuoBAID5Wc2HD
7JYdWxt5Pkdwr4
McdE0feGqRqW7Ca

2.4.2 GEN02

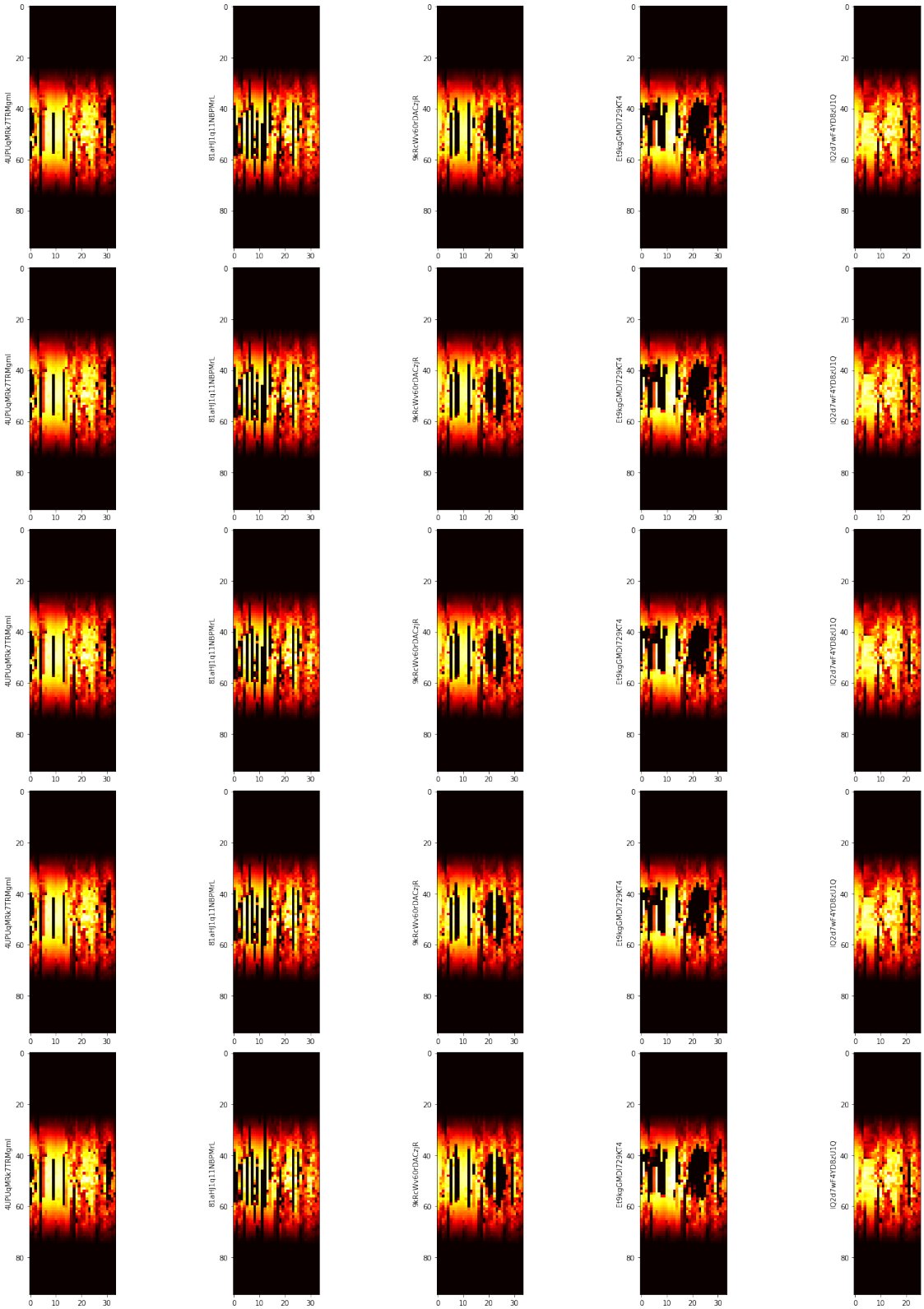
AC

[16]:
```
rowsSquare = math.ceil(math.sqrt(len(list(sourcesGen02["AC_POWER"].keys()))))

fig, axes = plt.subplots(nrows=rowsSquare, ncols=rowsSquare, figsize=(20, 30))
fig.tight_layout()

# Works with daily and high resolution data for interpolar given interval␣
 ↪length.

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen02["AC_POWER"].items()):
        df = source[1].copy(deep=True)
        df.index = pd.to_datetime(df.index,  format='%H:%M:%S')
        ax.imshow(df.interpolate(method="time").interpolate(method="linear",␣
 ↪axis=1), cmap="hot")
        ax.set_ylabel(source[0])

plt.subplots_adjust(top=0.85)
plt.show()
```



DC

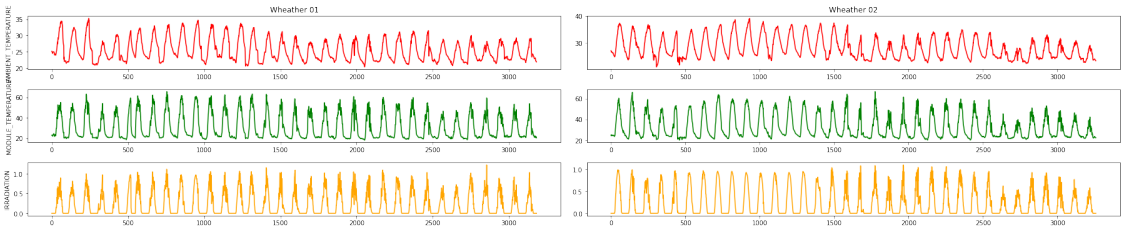
[17]:
```
rowsSquare = math.ceil(math.sqrt(len(list(sourcesGen02["DC_POWER"].keys()))))

fig, axes = plt.subplots(nrows=rowsSquare, ncols=rowsSquare, figsize=(20, 30))
fig.tight_layout()

# Works with daily and high resolution data for interpolar given interval␣
 ↪length.

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesGen02["DC_POWER"].items()):
        df = source[1].copy(deep=True)
        df.index = pd.to_datetime(df.index,  format='%H:%M:%S')
        ax.imshow(df.interpolate(method="time").interpolate(method="linear",␣
 ↪axis=1), cmap="hot")
        ax.set_ylabel(source[0])

plt.subplots_adjust(top=0.85)
plt.show()
```



3 2. Weather

3.1 2.1. Initial view

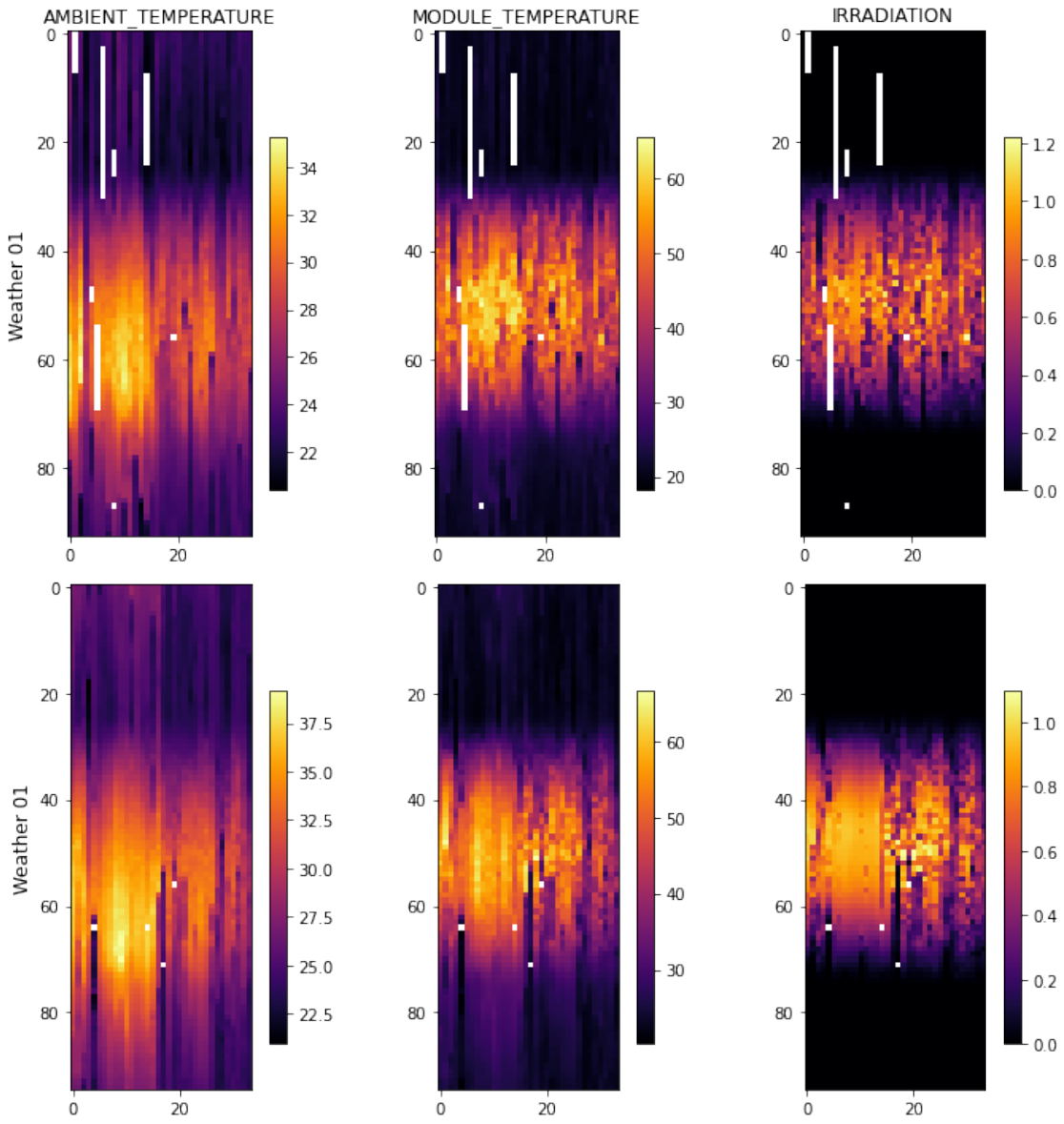
[18]:
```
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(25, 5))
fig.tight_layout()

axes[0][0].set_title("Wheather 01")
dfWheather01['AMBIENT_TEMPERATURE'].plot(color="red", ax=axes[0][0])
dfWheather01['MODULE_TEMPERATURE'].plot(color="green", ax=axes[1][0])
dfWheather01['IRRADIATION'].plot(color="orange", ax=axes[2][0])

axes[0][1].set_title("Wheather 02")
dfWheather02['AMBIENT_TEMPERATURE'].plot(color="red", ax=axes[0][1])
dfWheather02['MODULE_TEMPERATURE'].plot(color="green", ax=axes[1][1])
dfWheather02['IRRADIATION'].plot(color="orange", ax=axes[2][1])

axes[0][0].set_ylabel("AMBIENT_TEMPERATURE", rotation=90)
axes[1][0].set_ylabel("MODULE_TEMPERATURE", rotation=90)
axes[2][0].set_ylabel("IRRADIATION", rotation=90)

plt.show()
```



3.2 2.2. View heatmap by sources in time-range

[19]:
```
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(10, 10))

a0 = axes[0][0].imshow(groupsWeather01["AMBIENT_TEMPERATURE"], cmap="inferno")
a1 = axes[0][1].imshow(groupsWeather01["MODULE_TEMPERATURE"], cmap="inferno")
a2 = axes[0][2].imshow(groupsWeather01["IRRADIATION"], cmap="inferno")

a3 = axes[1][0].imshow(groupsWeather02["AMBIENT_TEMPERATURE"], cmap="inferno")
a4 = axes[1][1].imshow(groupsWeather02["MODULE_TEMPERATURE"], cmap="inferno")
a5 = axes[1][2].imshow(groupsWeather02["IRRADIATION"], cmap="inferno")

fig.colorbar(a0, ax=axes[0][0], location='right', anchor=(0, 0.3), shrink=0.7)
fig.colorbar(a1, ax=axes[0][1], location='right', anchor=(0, 0.3), shrink=0.7)
fig.colorbar(a2, ax=axes[0][2], location='right', anchor=(0, 0.3), shrink=0.7)
```

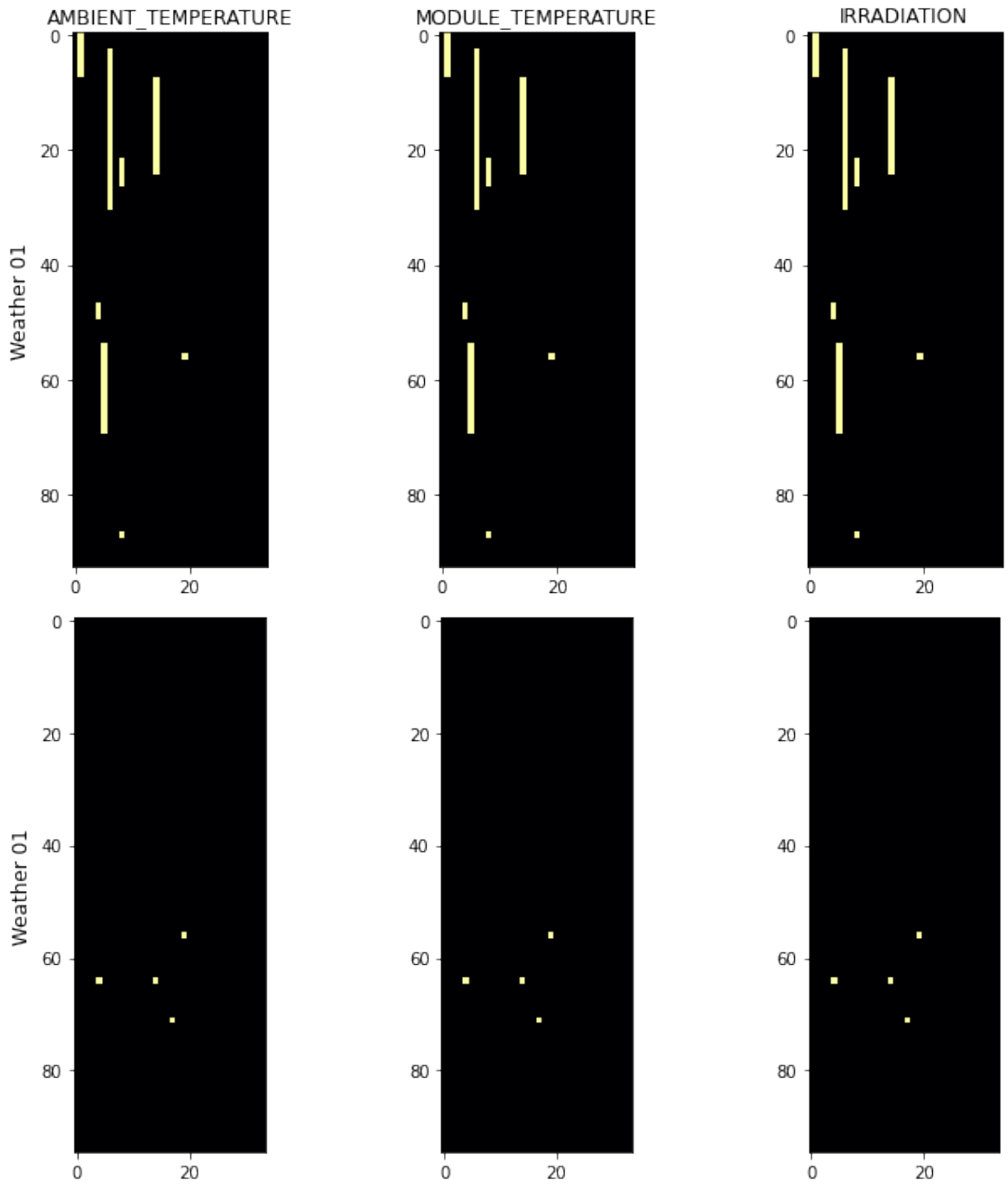
```
fig.colorbar(a3, ax=axes[1][0], location='right', anchor=(0, 0.3), shrink=0.7)
fig.colorbar(a4, ax=axes[1][1], location='right', anchor=(0, 0.3), shrink=0.7)
fig.colorbar(a5, ax=axes[1][2], location='right', anchor=(0, 0.3), shrink=0.7)


axes[0][0].set_ylabel("Weather 01", rotation=90, size='large')
axes[1][0].set_ylabel("Weather 02", rotation=90, size='large')

axes[0][0].set_title("AMBIENT_TEMPERATURE")
axes[0][1].set_title("MODULE_TEMPERATURE")
axes[0][2].set_title("IRRADIATION")

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



3.3 2.3. Visualization of missing elements

[20]:
```
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(10, 10))

plt.rcParams['figure.figsize'] = [15, 15]

axes[0][0].imshow(groupsWeather01["AMBIENT_TEMPERATURE"].isnull().values,␣
↪cmap="inferno")
axes[0][1].imshow(groupsWeather01["MODULE_TEMPERATURE"].isnull().values,␣
↪cmap="inferno")
```

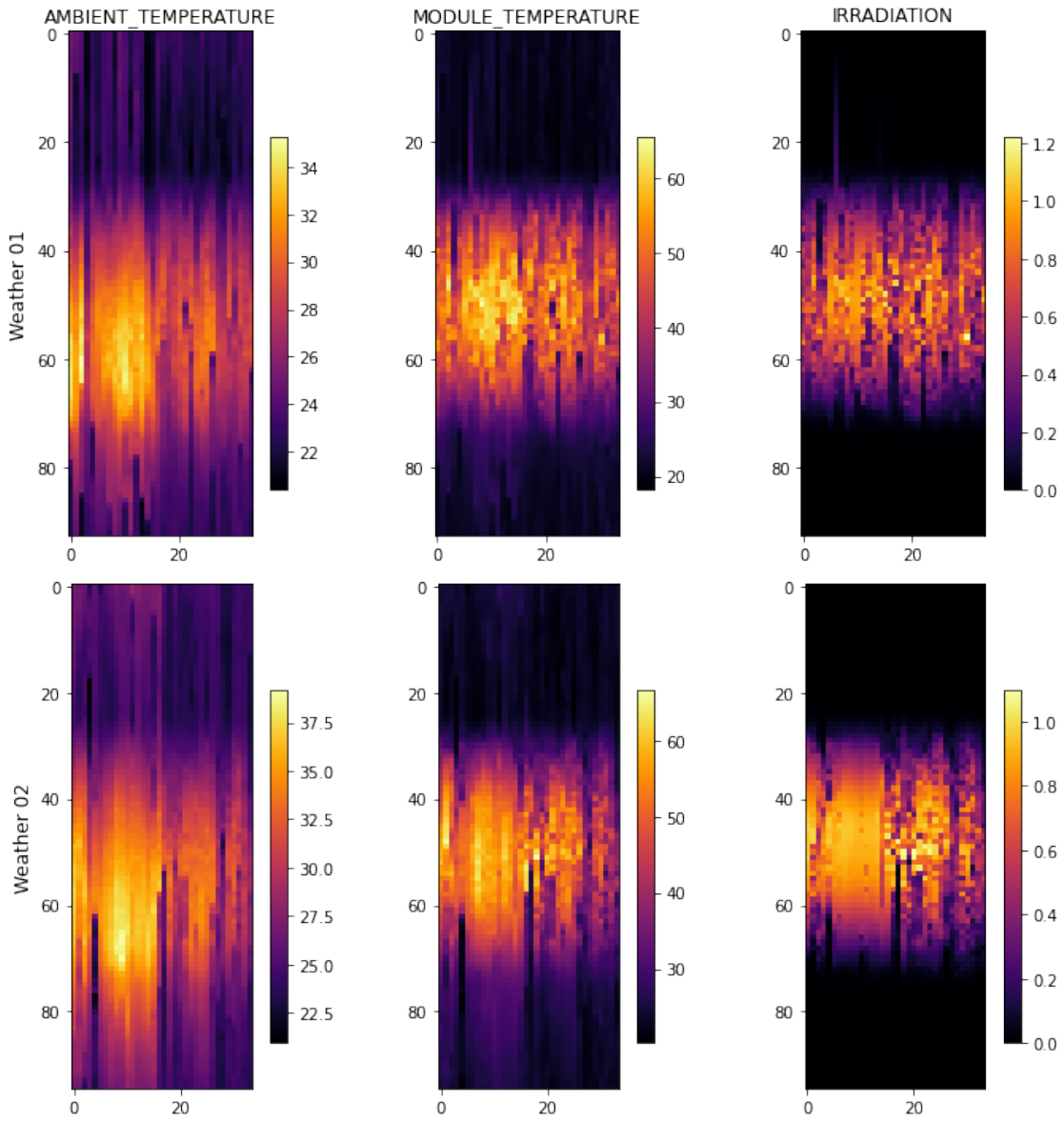
```
axes[0][2].imshow(groupsWeather01["IRRADIATION"].isnull().values,␣
 ↪cmap="inferno")

axes[1][0].imshow(groupsWeather02["AMBIENT_TEMPERATURE"].isnull().values,␣
 ↪cmap="inferno")
axes[1][1].imshow(groupsWeather02["MODULE_TEMPERATURE"].isnull().values,␣
 ↪cmap="inferno")
axes[1][2].imshow(groupsWeather02["IRRADIATION"].isnull().values,␣
 ↪cmap="inferno")

axes[0][0].set_ylabel("Weather 01", rotation=90, size='large')
axes[1][0].set_ylabel("Weather 02", rotation=90, size='large')

axes[0][0].set_title("AMBIENT_TEMPERATURE")
axes[0][1].set_title("MODULE_TEMPERATURE")
axes[0][2].set_title("IRRADIATION")

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



3.4 2.4. Interpolate

[22]:
```
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(10, 10))

a0 = axes[0][0].imshow(groupsWeather01["AMBIENT_TEMPERATURE"].interpolate(
    method="time").interpolate(method="linear", axis=1), cmap="inferno")
a1 = axes[0][1].imshow(groupsWeather01["MODULE_TEMPERATURE"].interpolate(
```

```
    method="time").interpolate(method="linear", axis=1), cmap="inferno")
a2 = axes[0][2].imshow(groupsWeather01["IRRADIATION"].interpolate(
    method="time").interpolate(method="linear", axis=1), cmap="inferno")

a3 = axes[1][0].imshow(groupsWeather02["AMBIENT_TEMPERATURE"].interpolate(
    method="time").interpolate(method="linear", axis=1), cmap="inferno")
a4 = axes[1][1].imshow(groupsWeather02["MODULE_TEMPERATURE"].interpolate(
    method="time").interpolate(method="linear", axis=1), cmap="inferno")
a5 = axes[1][2].imshow(groupsWeather02["IRRADIATION"].interpolate(
    method="time").interpolate(method="linear", axis=1), cmap="inferno")

fig.colorbar(a0, ax=axes[0][0], location='right', anchor=(0, 0.3), shrink=0.7)
fig.colorbar(a1, ax=axes[0][1], location='right', anchor=(0, 0.3), shrink=0.7)
fig.colorbar(a2, ax=axes[0][2], location='right', anchor=(0, 0.3), shrink=0.7)
fig.colorbar(a3, ax=axes[1][0], location='right', anchor=(0, 0.3), shrink=0.7)
fig.colorbar(a4, ax=axes[1][1], location='right', anchor=(0, 0.3), shrink=0.7)
fig.colorbar(a5, ax=axes[1][2], location='right', anchor=(0, 0.3), shrink=0.7)


axes[0][0].set_ylabel("Weather 01", rotation=90, size='large')
axes[1][0].set_ylabel("Weather 02", rotation=90, size='large')

axes[0][0].set_title("AMBIENT_TEMPERATURE")
axes[0][1].set_title("MODULE_TEMPERATURE")
axes[0][2].set_title("IRRADIATION")

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

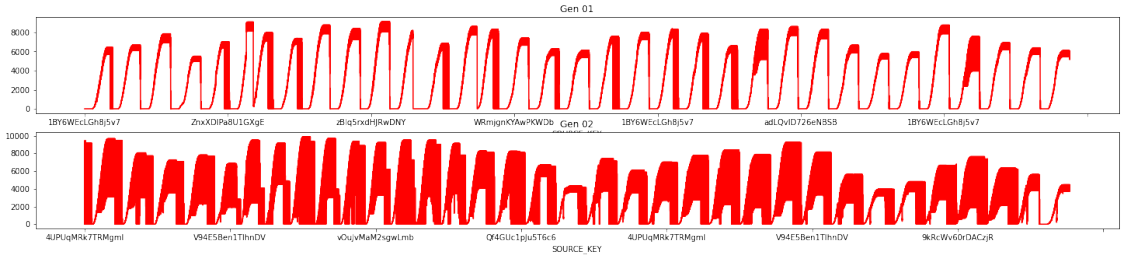


4 3. Daily Yield

4.1 3.1. Initial view

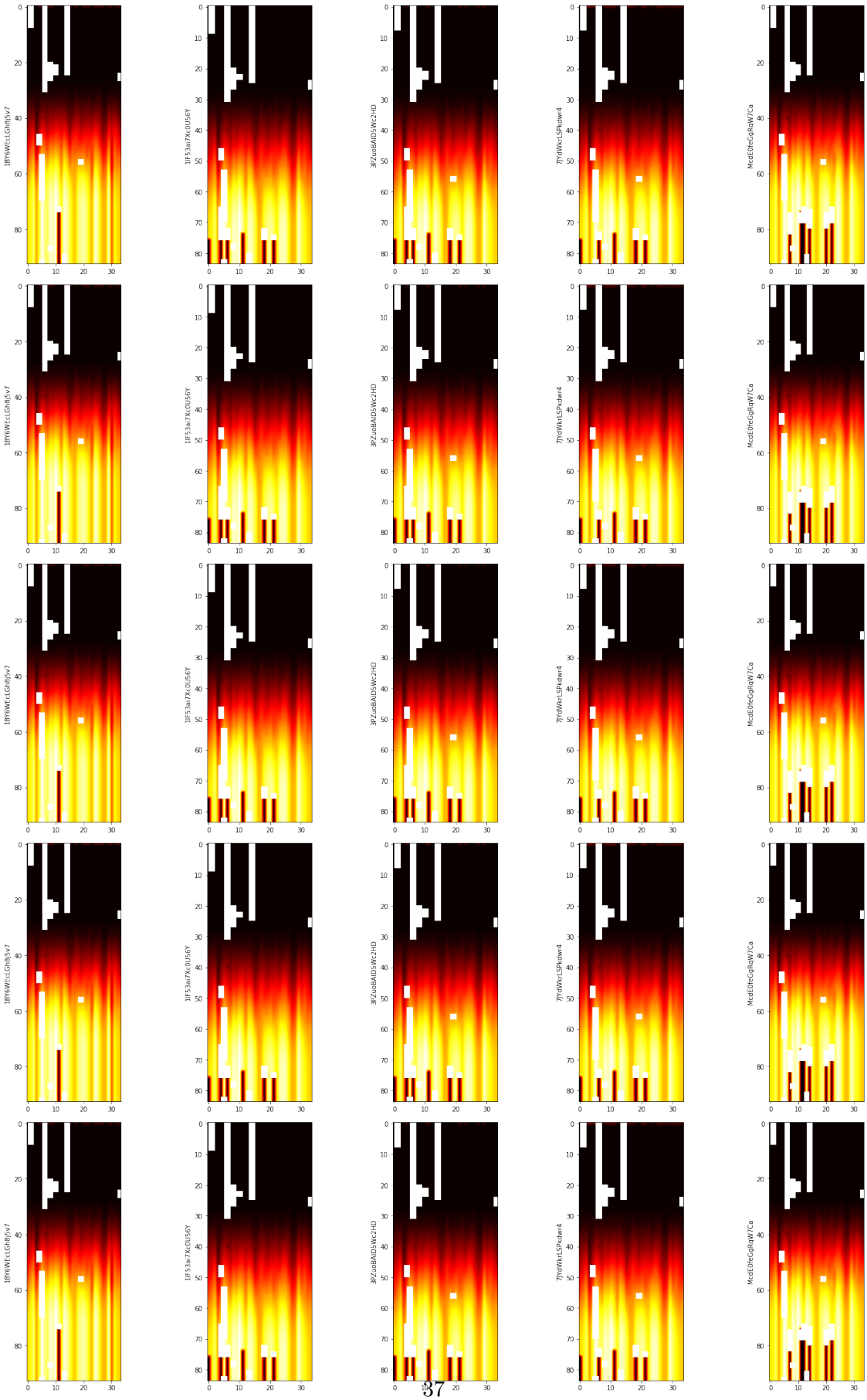
[20]:
```
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(25, 5))

axes[0].set_title("Gen 01")
dfGen01['DAILY_YIELD'].plot(color="red", ax=axes[0])

axes[1].set_title("Gen 02")
dfGen02['DAILY_YIELD'].plot(color="red", ax=axes[1])
```

```
plt.show()
```



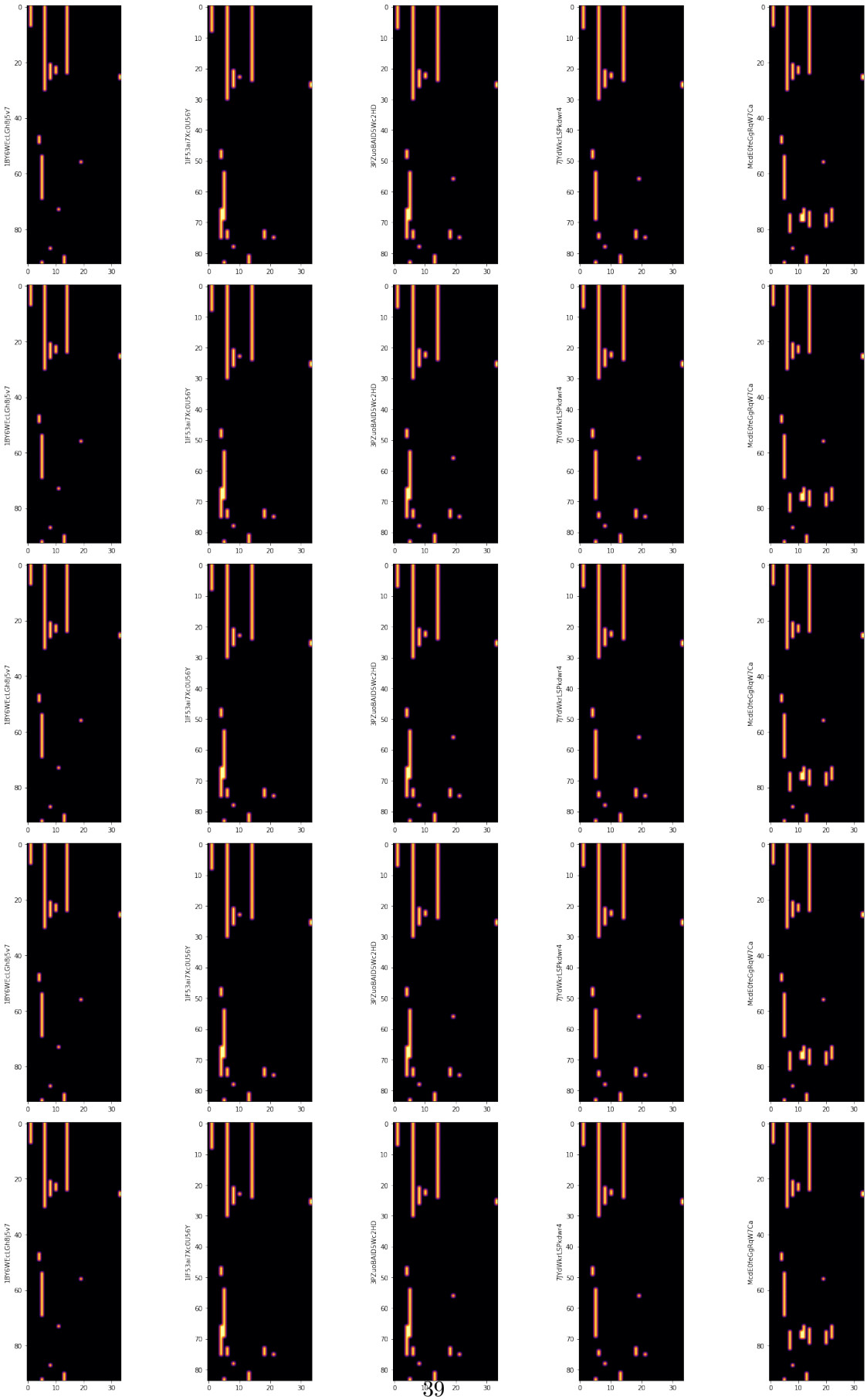
4.2 3.2. View heatmap by sources in time-range

[3]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesDailyYield["gen01"].keys()))
))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesDailyYield["gen01"].items()):
        ax.imshow(source[1], cmap="hot",  interpolation='bilinear')
        ax.set_ylabel(source[0])

plt.show()
```


1BY6WEcLGhBJ5v7
1IF53ai7XcOU56Y
3PZuoBAlD5Wc2HD
7fVdWxrLSPkdwr4
McdE0feCgRqW7Ca

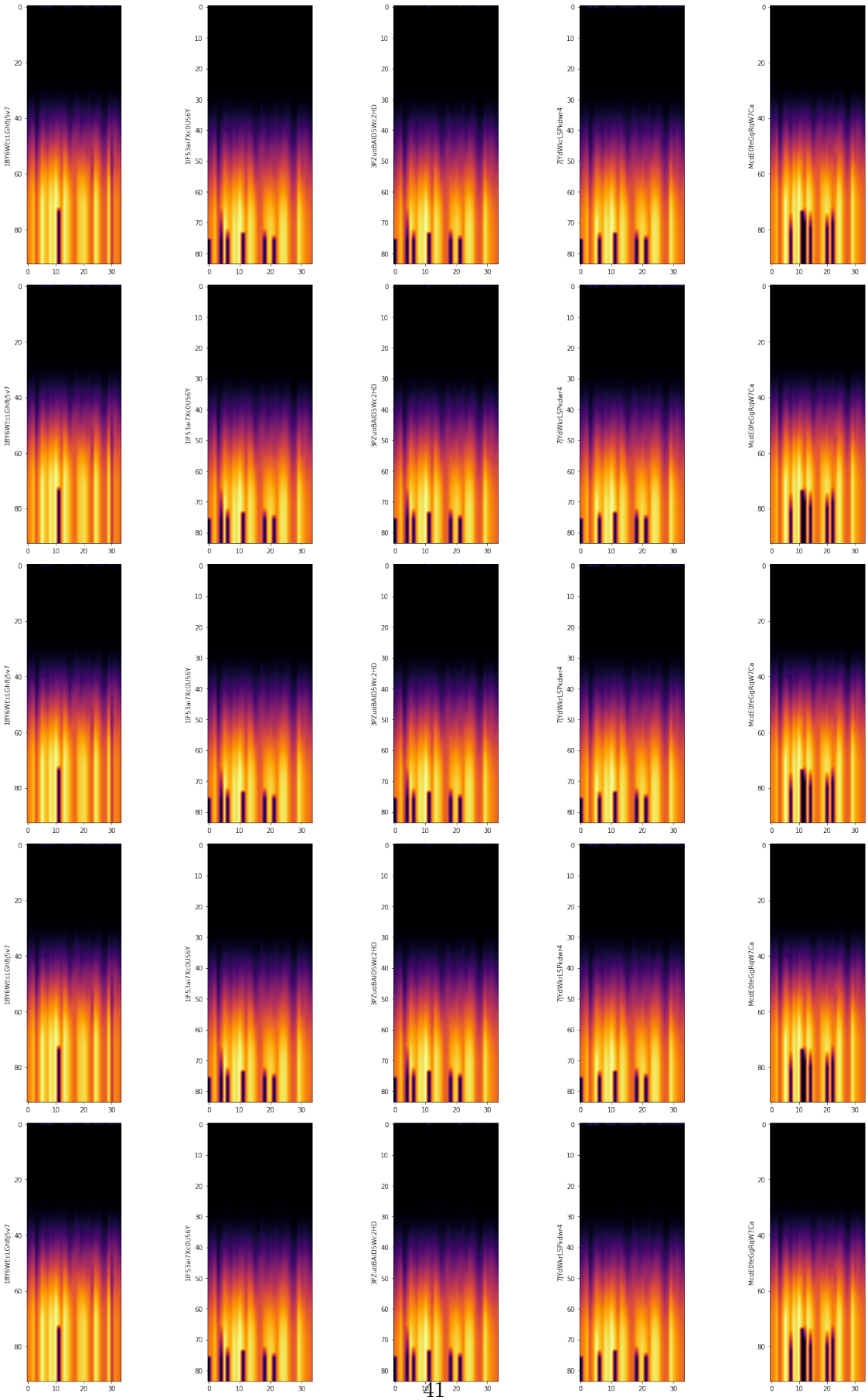
4.3 3.3. Visualization of missing elements

[5]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesDailyYield["gen01"].keys()))
))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesDailyYield["gen01"].items()):
        ax.imshow(source[1].isnull().values, cmap="inferno", ␣
 ↪interpolation='bilinear')
        ax.set_ylabel(source[0])

plt.show()
```


1BY6WEcLGhBj5v7
1IF53ai7XcOU56Y
3PZuoBAlD5Wc2HD
7fYdWxcLSPkdwr4
McdE0feCgRqW7Ca

4.4 3.4. Interpolate

[7]:
```
square = math.ceil(
    math.sqrt(
        len(list(sourcesDailyYield["gen01"].keys()))
))

fig, axes = plt.subplots(nrows=square, ncols=square, figsize=(20, 30))
fig.tight_layout()

for axesRow in axes:
    for ax, source in zip(axesRow, sourcesDailyYield["gen01"].items()):
        ax.imshow(
            source[1].interpolate(
                method="time").interpolate(
                    method="linear", axis=1),
            cmap="inferno",
            interpolation='bilinear'
        )
        ax.set_ylabel(source[0])

plt.show()
```


1BY6WEcLGhBj5v7
1IF53ai7XcOU56Y
3PZuoBAlD5Wc2HD
7fYdWxcLSPkdwr4
McdE0feCgRqW7Ca