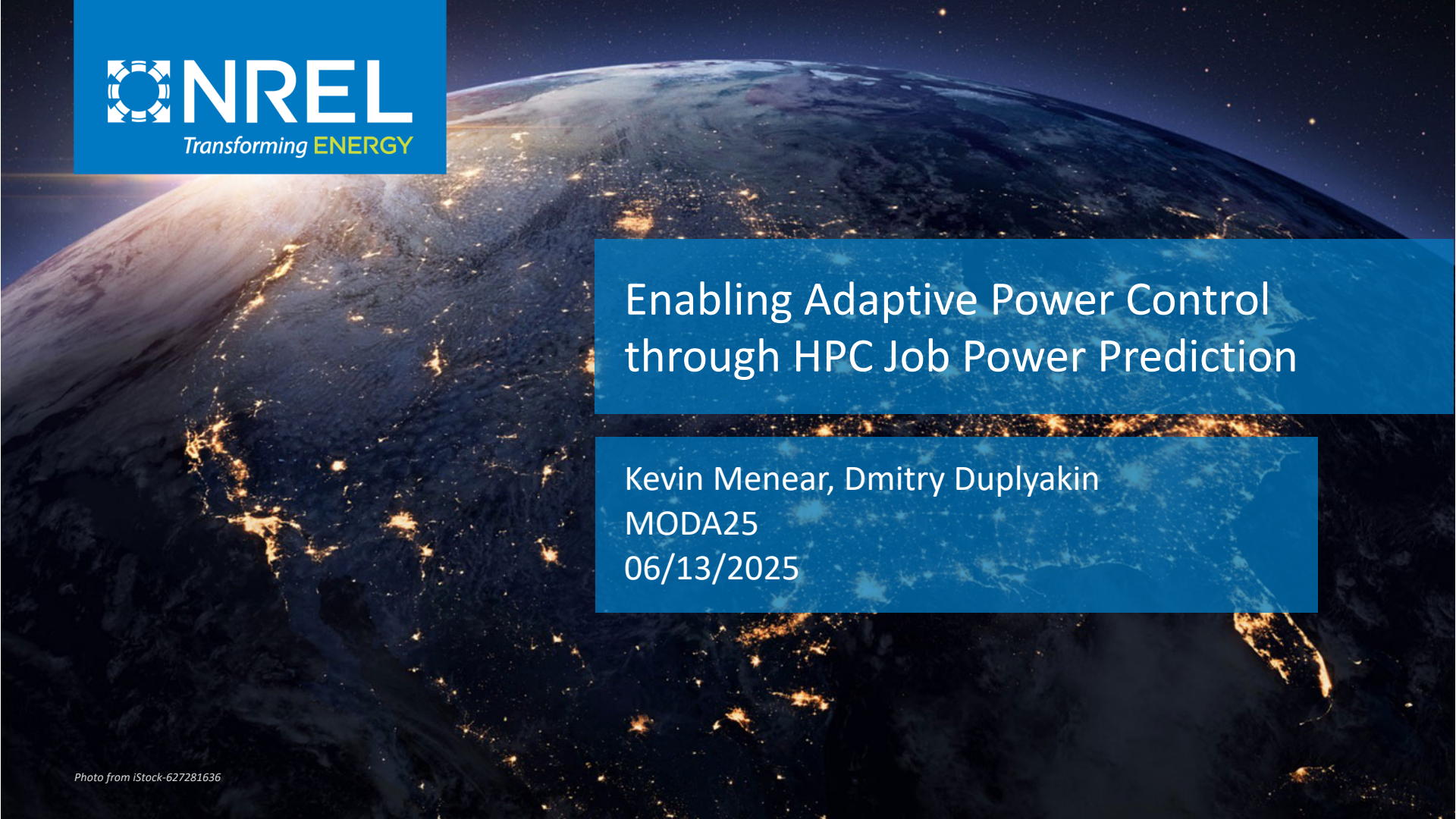


Enabling Adaptive Power Control through HPC Job Power Prediction

Kevin Menear, Dmitry Duplyakin
MODA25
06/13/2025

Photo from iStock-627281636

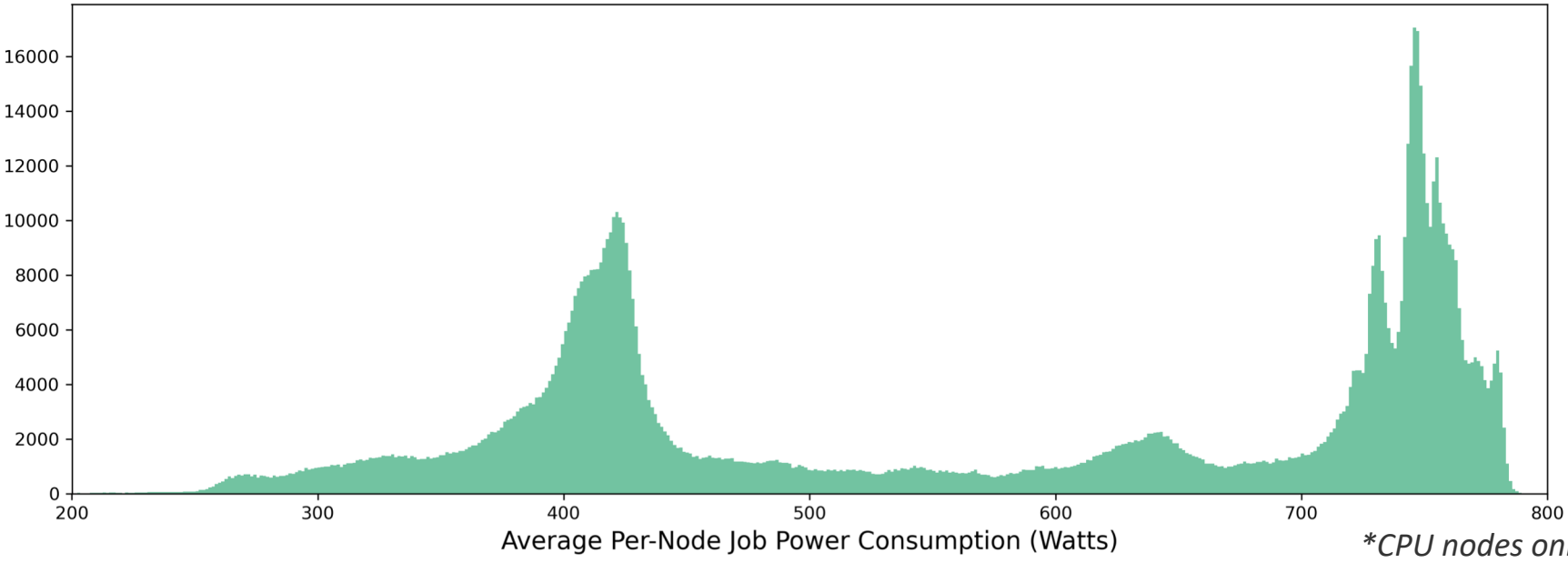
Load Flexibility

AI and large-scale simulations are **driving up datacenter power use**, straining the grid

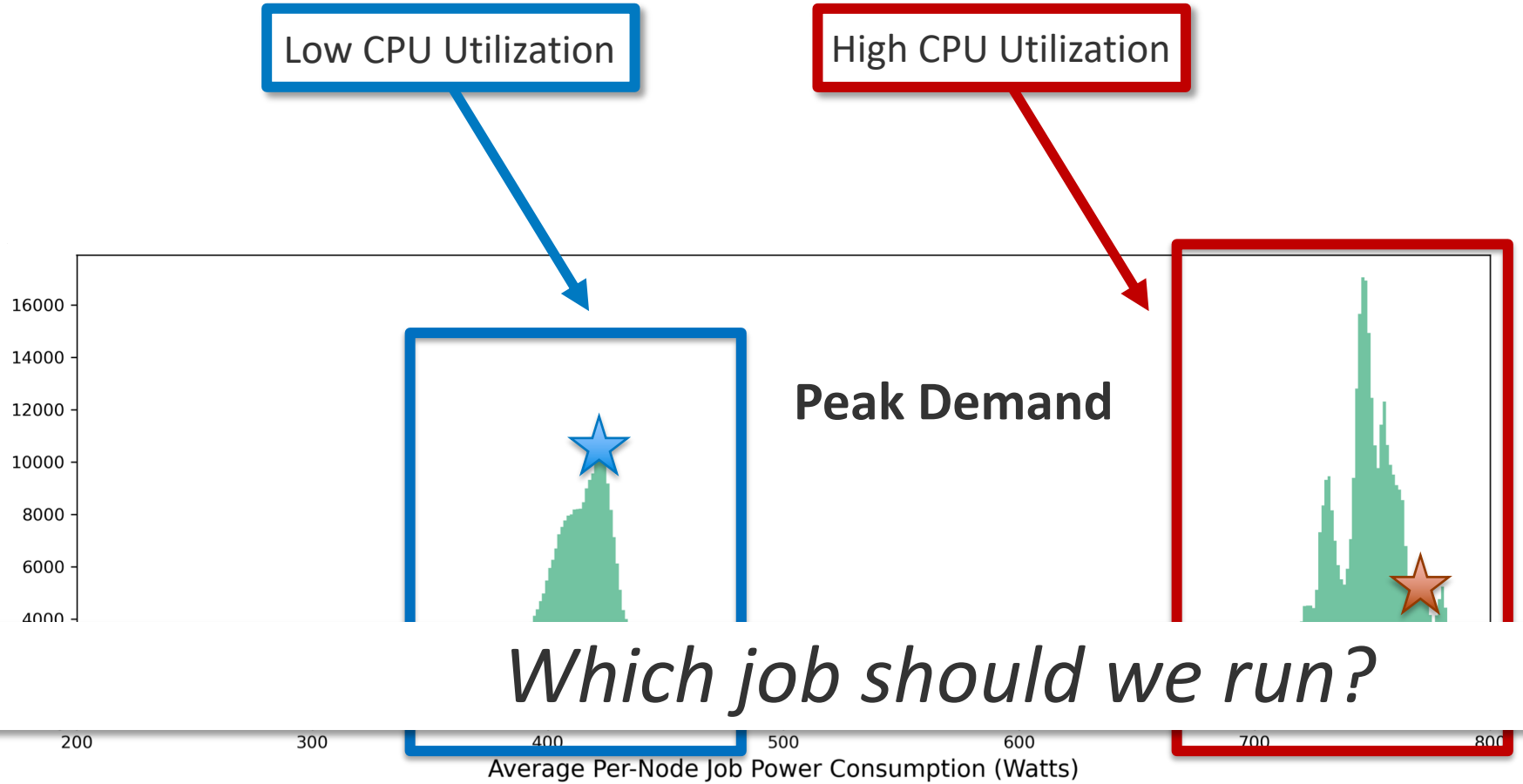
Just 1% load curtailment could enable **126 GW of new datacenter load** *without grid expansion**

Load flexibility can **expedite new datacenter construction and decrease costs**

*Norris, T., Profeta, T., Patino-Echeverri, D., & Cowie-Haskell, A. (2025). Rethinking Load Growth: Assessing the Potential for Integration of Large Flexible Loads in US Power Systems.



*CPU nodes only



Which job should we run?

…but we need to know job power consumption ***in advance.***

How can we predict per-job power consumption?

Available Data

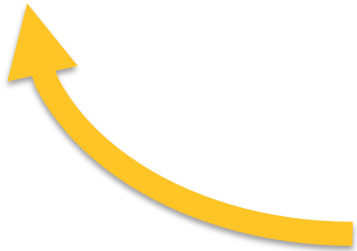
- Only data available at job submission time (*before the job runs*)
- **Metadata**
  - Submit time, user name, account
- **User-provided information**
  - Resources requested, modules to load, commands to execute
  - Single line (*potentially multiple commands*) → Submit line
  - Shell script → Job script

State of the Art (Traditional Approach)

- **Extract meaningful information** from the job script/submit line
  - Feature extraction
- **Preprocess data**
  - Feature engineering
    - Categorical features must be encoded (transformed to numerical)
    - Label encoding, one-hot encoding, target encoding…
      - More recently, encoding with language models!
- **Train a machine learning model/kNN with job features**

State of the Art (Traditional Approach)

- **Extract meaningful information** from the job script/submit line
  - Feature extraction



This is the problematic step!

- Domain expert decides what to keep and what to discard
- Drops most task-specific tokens (e.g. descriptive job names, software flags, Python code)

What if we could use all the information in the job script?

What if we could use all the information in the job script?

The challenge is, it is mostly unstructured text

The challenge is, it is mostly unstructured text

```
User:              [SCRUBBED_USER]
Account:           [SCRUBBED_ACCOUNT]
Partition:         standard
Job Type:          python
Job Name:          [SCRUBBED_JOB_NAME]
QOS:               normal
Submit Line:       sbatch [SCRUBBED_LAUNCH_SCRIPT]

Script:
#!/bin/bash
#SBATCH --account=[SCRUBBED_ACCOUNT]
#SBATCH --time=40:00:00
#SBATCH --job-name=[SCRUBBED_JOB_NAME]
#SBATCH --nodes=1                # number of nodes
#SBATCH --output=[SCRUBBED_PATH]/stdout/[SCRUBBED_JOB_NAME]_%j.o
#SBATCH --error=[SCRUBBED_PATH]/stdout/[SCRUBBED_JOB_NAME]_%j.e
#SBATCH --qos=normal             # extra feature

echo "Running on: $HOSTNAME, Machine Type: $MACHTYPE"

python -c '
from nsrdb.nsrdb import NSRDB
NSRDB.run_data_model(
    "[SCRUBBED_PATH_ROOT]/",           # root data directory
    "20201014",                        # target date
    "[SCRUBBED_PATH]/surfrad_meta.csv",
    freq="5min",
    var_list=None,
    dist_lim=2.0,
    max_workers=1,
    max_workers_regrid=1,
    log_level="INFO",
    log_file="data_model/data_model.log",
    job_name="[SCRUBBED_JOB_NAME]",
    factory_kwargs={
        "cld_opd_dcomp":       {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "cld_press_acha":      {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "cld_reff_dcomp":      {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "cloud_fraction":      {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "cloud_probability":   {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "cloud_type":          {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "refl_0_65um_nom":     {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "refl_0_65um_nom_stddev_3x3": {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "refl_3_75um_nom":     {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "surface_albedo":      {"source_dir": "[SCRUBBED_PATH]"},
        "temp_11_0um_nom":     {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "temp_11_0um_nom_stddev_3x3": {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "temp_3_75um_nom":     {"pattern": "[SCRUBBED_PATH_PATTERN]"}
    },
    mlclouds=True
)'
```

The challenge is, it is mostly unstructured text

…but that's exactly the kind of input LLMs are designed to understand.

```
User:              [SCRUBBED_USER]
Account:           [SCRUBBED_ACCOUNT]
Partition:         standard
Job Type:          python
Job Name:          [SCRUBBED_JOB_NAME]
QOS:               normal
Submit Line:       sbatch [SCRUBBED_LAUNCH_SCRIPT]

Script:
#!/bin/bash
#SBATCH --account=[SCRUBBED_ACCOUNT]
#SBATCH --time=40:00:00
#SBATCH --job-name=[SCRUBBED_JOB_NAME]
#SBATCH --nodes=1                # number of nodes
#SBATCH --output=[SCRUBBED_PATH]/stdout/[SCRUBBED_JOB_NAME]_%j.o
#SBATCH --error=[SCRUBBED_PATH]/stdout/[SCRUBBED_JOB_NAME]_%j.e
#SBATCH --qos=normal             # extra feature

echo "Running on: $HOSTNAME, Machine Type: $MACHTYPE"

python -c '
from nsrdb.nsrdb import NSRDB
NSRDB.run_data_model(
    "[SCRUBBED_PATH_ROOT]/",           # root data directory
    "20201014",                        # target date
    "[SCRUBBED_PATH]/surfrad_meta.csv",
    freq="5min",
    var_list=None,
    dist_lim=2.0,
    max_workers=1,
    max_workers_regrid=1,
    log_level="INFO",
    log_file="data_model/data_model.log",
    job_name="[SCRUBBED_JOB_NAME]",
    factory_kwargs={
        "cld_opd_dcomp":       {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "cld_press_acha":      {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "cld_reff_dcomp":      {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "cloud_fraction":      {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "cloud_probability":   {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "cloud_type":          {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "refl_0_65um_nom":     {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "refl_0_65um_nom_stddev_3x3": {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "refl_3_75um_nom":     {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "surface_albedo":      {"source_dir": "[SCRUBBED_PATH]"},
        "temp_11_0um_nom":     {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "temp_11_0um_nom_stddev_3x3": {"pattern": "[SCRUBBED_PATH_PATTERN]"},
        "temp_3_75um_nom":     {"pattern": "[SCRUBBED_PATH_PATTERN]"}
    },
    mlclouds=True
)'
```

HPC Job Script & Metadata

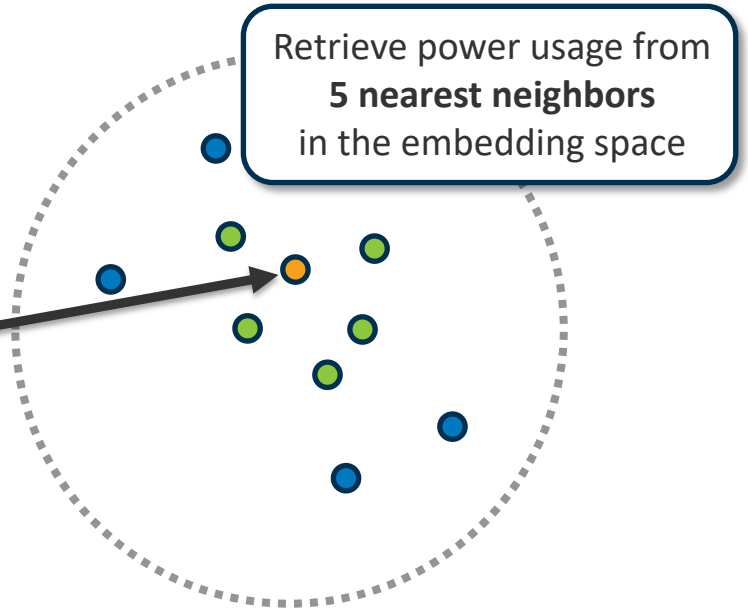
```
User: jdoe
Account: science
Partition: standard
Job Type: vasp
Job Name: example_job
QOS: normal
Submit Line: sbatch job.sh
Script: #!/bin/bash --login
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=36
#SBATCH --time=12:00:00
#SBATCH --job-name=example_job
#SBATCH --partition=standard
module load vasp
srun vasp_std
```

Encoded with LLM

**Linq-Embed-Mistral*

4,096 Dimension Vector Embedding

$$\begin{bmatrix} 0.342 \\ -0.209 \\ \vdots \\ 0.893 \\ 0.416 \end{bmatrix}$$



Predicted power is *weighted average* of nearest neighbor power usage

$$\hat{P}_{\text{job}} = \frac{\sum_{i=1}^{5} \cos(\theta_i) \cdot P_i}{\sum_{i=1}^{5} \cos(\theta_i)}$$

$\hat{P}_{job} \coloneqq$ *new job predicted power*

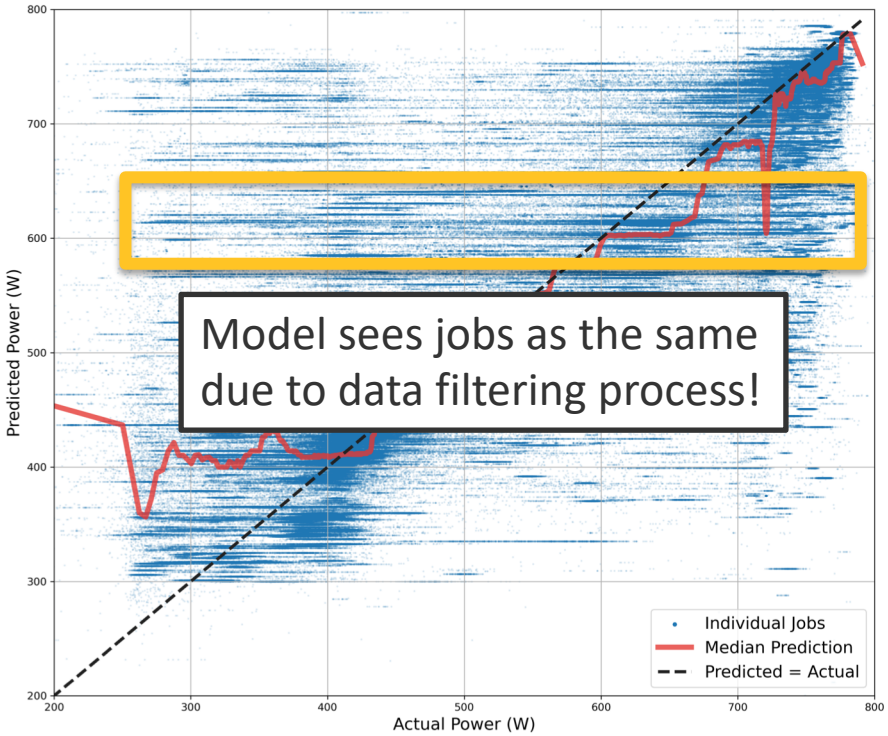
$i \coloneqq$ *completed job (neighbor)*

$\cos(\theta_i) \coloneqq$ *cosine similarity*

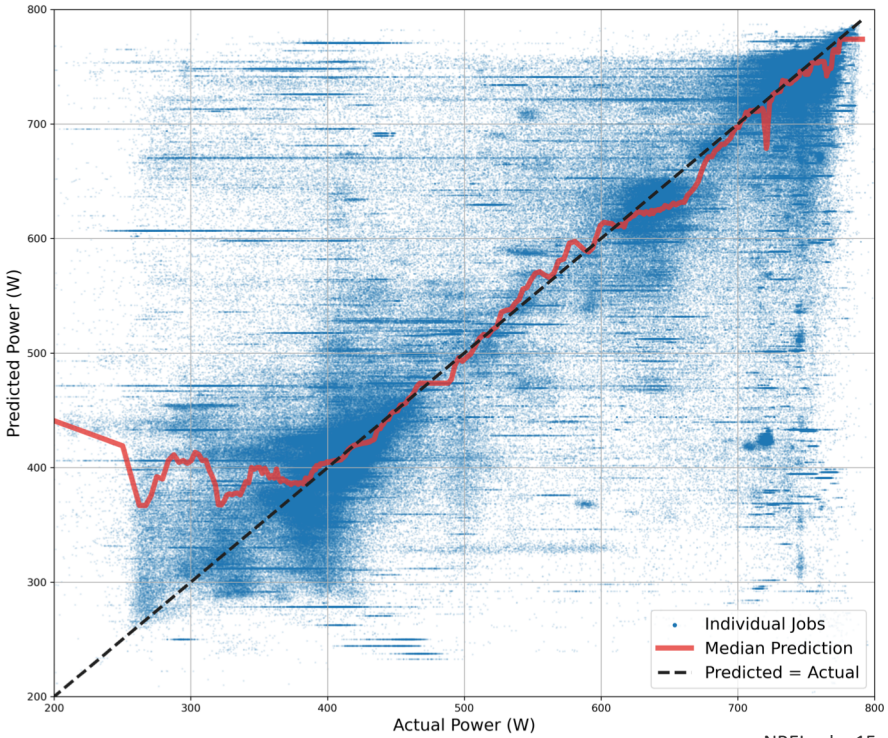
$P_i \coloneqq$ *neighbor power usage*

Results

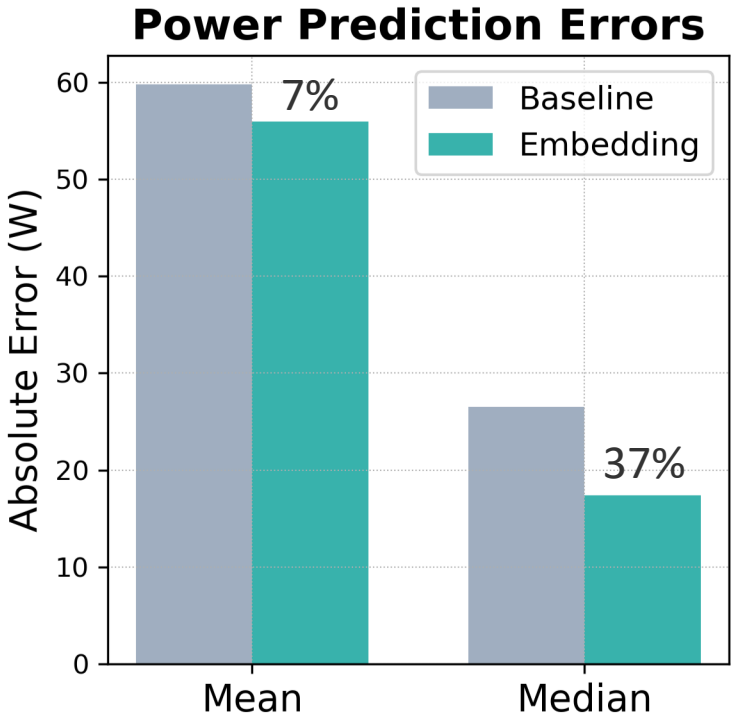
Baseline (Machine Learning Model)

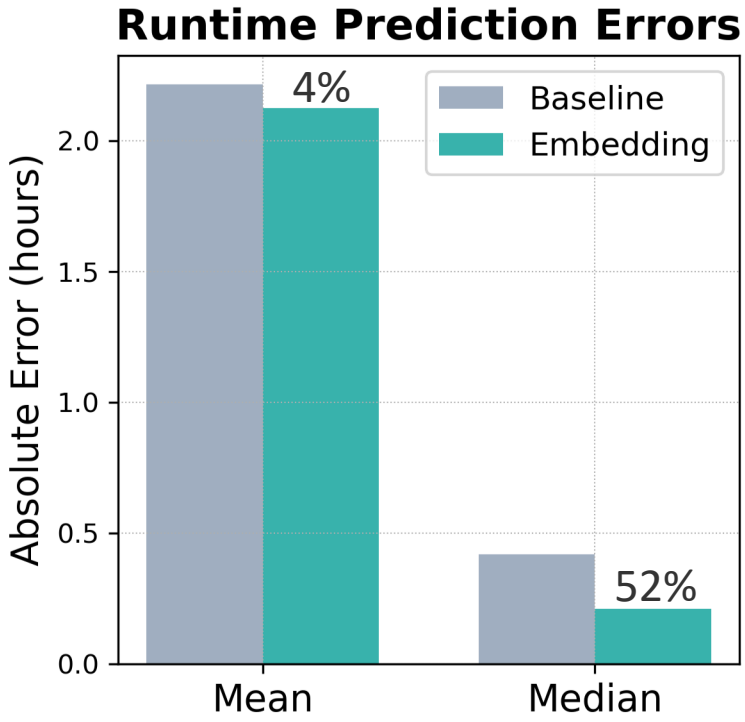


New Approach (Semantic Search)



Results

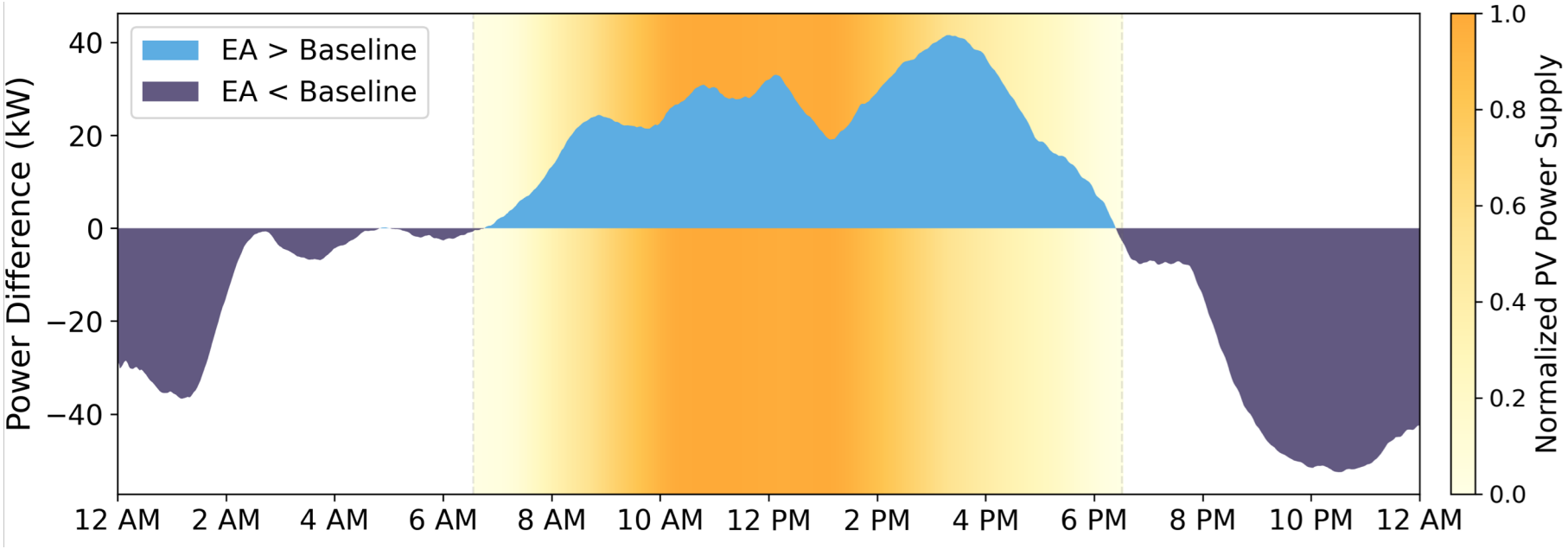




Can predict other metrics **without any additional overhead**

Current Application

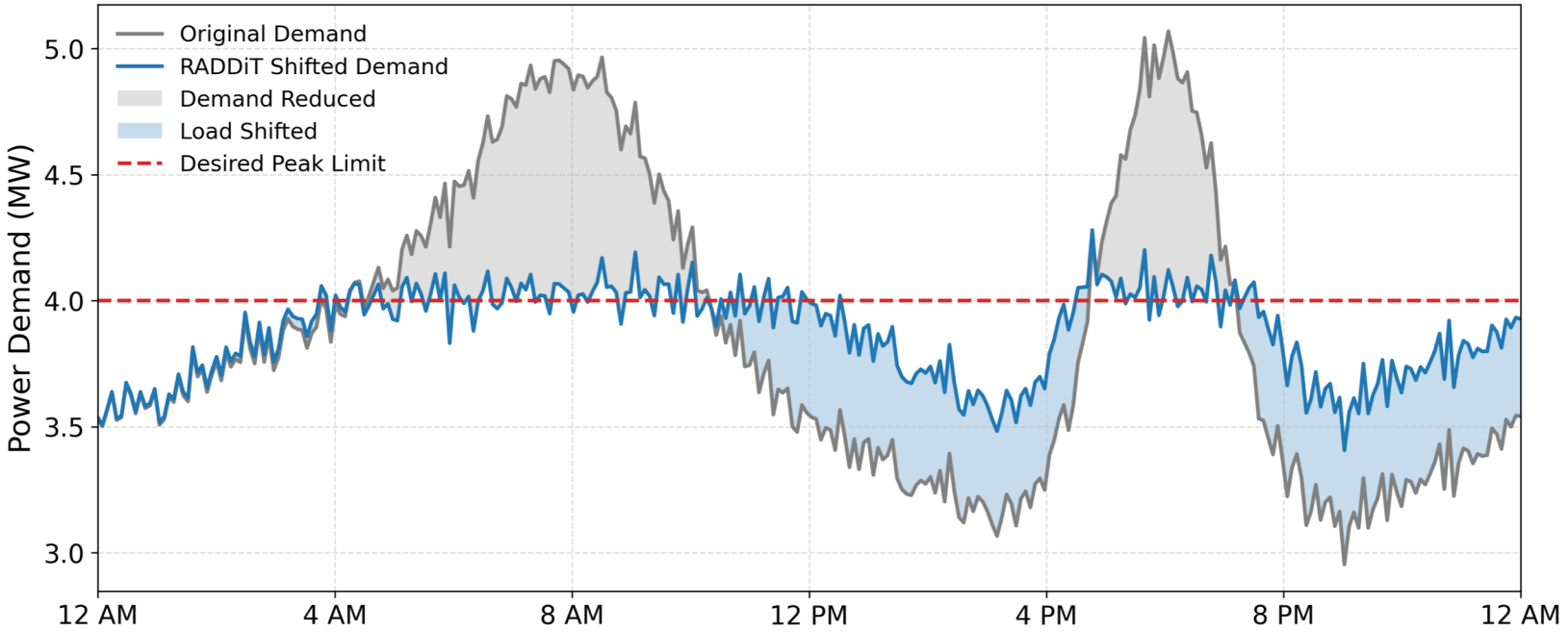
(Simulated) Aligning Kestrel power usage with behind-the-meter PV power availability



EA = Energy-Aware Scheduling

Future Objective

(Mockup) Shifting load to reduce peak power demand




NREL
Transforming ENERGY
Thank you!
Kevin.Menear@nrel.gov
Photo from iStock-627281636