CS696: Advancing Grid Carbon Intensity Forecasts with Dynamic Data Integration in CarbonCast

Rohan Lekhwani
rlekhwani@umass.edu
University of Massachusetts Amherst
Amherst, MA, USA

Diptyaroop Maji
dmaji@cs.umass.edu
University of Massachusetts Amherst
Amherst, MA, USA

Prashant Shenoy
shenoy@cs.umass.edu
University of Massachusetts Amherst
Amherst, MA, USA

Abstract

An increasing focus on the use of renewable energy for electricity generation has brought greater attention to shifting demands to low-carbon regions and times. Forecasting a grid's carbon intensity is a challenging task in this direction. In this project, we extend CarbonCast, a hierarchical machine learning approach for providing multi-day grid carbon intensity forecasts. We explore multiple real-time weather forecast services and make the process of generating carbon intensity forecasts end-to-end. Additionally, we reduce the granularity of source data from 3-hour windows to 1-hour windows which allows CarbonCast first-tier models access to more continuous data. We add these features for the US and European regions. All code as a part of this project has been made open-source within the CarbonCast repository.[1].

1 Introduction

Electric grids across the world, use a combination of renewable (e.g. solar, wind, hydro) and non-renewable (e.g. coal, natural gas) sources to generate electricity. The U.S. Energy Information Administration (EIA) predicts the annual electricity consumption for U.S. to be around 5.5 trillion kWh by 2050 [2]. Pertaining to this, it becomes imperative to minimize the amount of carbon emitted for every unit of electricity generated.

The contribution of each source towards the electricity generated by a grid varies with time of the day and season of the year. Carbon-aware systems are essential to shift workloads to period when renewable sources of energy are being used to generate electricity. Forecasts of the grid's electricity generation are a key requirement of such carbon-aware systems.

Carbon intensity refers to the average amount of carbon in *grams* per *kWh* unit of electricity generated. Recently, commercial services like ElectricityMap and Watttime are providing real-time carbon intensity data and forecasts. Unlike CarbonCast, their models are proprietary and pay-walled. Additionally, these services are limited to short-term (day-ahead) forecasting. CarbonCast provides upto 96 hour (4 days) carbon intensity forecasts.

In this project, we build upon CarbonCast's open-source codebase and extend it to have real-time predictions instead of the earlier bulk predictions. We make the process of computing and forecasting carbon intensities for a region and time end-to-end. Previously, CarbonCast weather forecasts were obtained at a 3 hour granularity and assumed to be constant for this duration. With newer sources of data, we reduce this window to a 1-hour window of data thereby adding greater continuity to the data, resulting in better predictions by the models.

[1]https://github.com/carbonfirst/CarbonCast/tree/rohan-independent-study

2 Background

In this section, we provide a background on the carbon emission factor of each source, the average carbon intensity calculation and a breakdown of CarbonCast. A knowledge of these three is an important factor for understanding the rest of this report.

2.1 Carbon Emission Factor

The carbon emission factor for a source is defined as the amount of carbon emitted into the atmosphere per unit of electricity generated by that source. It is measured in *g/kWh*. The following are the two types of CEFs for any source:

- The operational emissions when a source is converted into electricity account for the direct emission factor.
- The operational emissions in addition to the infrastructural emissions account for the lifecycle emission factor.

We consider a total of 8 sources when accounting for electricity generation. The "Other" sources correspond to the ones not included above and similar to [3] their CEF is assumed to be the same across all regions.

2.2 Average Carbon Intensity

The average carbon intensity per unit of electricity generated in a region is the weighted average of carbon emitted by each source due to electricity generated by them. Mathematically, the average carbon intensity (in g/kW h) of a region at any time is as follows:

$$(CarbonIntensity)_{avg} = \frac{\Sigma(E_i * CEF_i}{\Sigma E_i} \tag{1}$$

where E_i is the electricity generated *(MW)* by a Source i and CEF_i is the CEF *(g/kWh)* of that source.

2.3 CarbonCast

Carbon Cast is a grid carbon intensity forecasting software that can make use of electricity and weather data to generate multi-day forecasts. As Figure shows, it has a two-tier architecture.

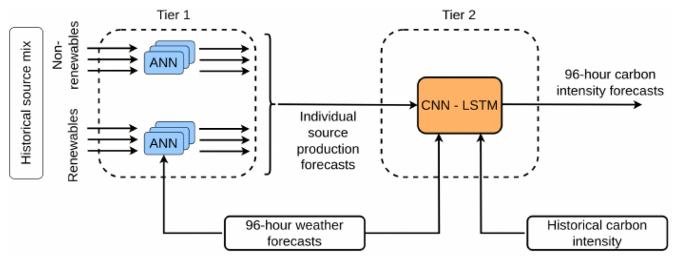
Tier 1
Tier 2
Historical source mix
Non-renewables
Renewables
ANN
ANN
Individual source production forecasts
CNN - LSTM
96-hour carbon intensity forecasts
96-hour weather forecasts
Historical carbon intensity

Figure 1. The two-tiered CarbonCast architecture. Figure from [3].

2.3.1 First-tier Design.

The CarbonCast first-tier uses a set of ANN models, one for each generation source, to predict the electricity production from that source for the next 96 hours. Temporal features like hour-of-day, hour-of-year, weekdays and weekends are factored into these models. Weather forecasts form additional inputs for renewable sources of energy.

2.3.2 Second-tier Design.

The second tier takes these first-tier predictions along with weather forecasts to predict the hourly carbon intensity of electricity in that region for the next 4 days. Using a combination of CNN and LSTM models helps account for noisy or missing data. The 1-D CNN layers extract high-level "short-term" temporal features from inputs which help the LSTM learn "longer-term" temporal patterns within the time series.

3 Contributions

In this section, we present the contributions to Carbon Cast made by this project.

3.1 EIA Parser

The *data/scripts/eiaParser.py* script is responsible for fetching the quantity of electricity generated in MWh using each of the 8 sources. The *getProductionDataBySourceTypeDataFromEIA* method is responsible for calling the API endpoint[2] with an API key. The provider has a default public API key which is used. The *parseEIAProductionDataBySourceType* method is responsible for parsing the dataset received in response and saving it as a CSV file within *<REGION>/day/<date>.csv* where *<REGION>* is the corresponding ISO region.

[2]https://api.eia.gov/v2/electricity/rto/fuel-type-data/data

3.2 ENTSOE Parser

The *data/scripts/entsoeParser.py* script outputs a parsed CSV similar to *eiaParser.py*. Instead of using an API endpoint, it makes use of the ENTSOE Python client made available by the provider [4]. Additionally, the labels used by ENTSOE differ from those used by EIA. These are reclassified to match with the labels used by EIA. For example, Fossil Brown coal/Lignite, Fossil Hard coal, Fossil Oil shale and Fossil Peat are reclassified as COAL.

3.3 Fetching Real-time Weather Data

Weather data in GRIB2 format is fetched using the NOMADS weather API endpoint [3]. The accessible data through this endpoint is up to 10 days in the past from the current date. The *getWeatherData_NOMADS.py* file is responsible for fetching weather data for parameters referenced in Table 1 when specified a region. The region to latitude-longitude mappings are done within this file.

Parameter	Level
U GRD / V GRD	HTGL:10
TMP / DPT	HTGL:2
DSWRF	SFC:0
A PCP	SFC:0

Table 1. Weather parameters and levels

The *dataCollectionScript* is responsible for getting the average values of each of the weather parameters and generating 96-hour real-time values for each day. The *cleanWeatherData* file then generates a weather forecast and saves it as *<ISO>_weather_forecast.csv* which is then used as input to the models.

3.4 End-to-end Forecasts

Previously, each stage of CarbonCast needed to be run separately by a human in a sequential manner. The *run.py* script added to the root of the project allows running the entire workflow end-to-end from downloading electricity and weather data to a 96-hour carbon intensity forecast CSV. The following are the stages that the script runs in order:

- Download the EIA/ENTSOE source electricity production data for all regions within the *<REGION>/day/<date>.csv* location.
- Generate direct and lifecycle emission files for the specified region and date. The file is also pre-processed by moving the "other" source column at the end.
- Download weather data for the specified region and for each of the parameters specified in the previous section.

[3]https://nomads.ncep.noaa.gov/cgi-bin/filter_gfs_0p25.pl

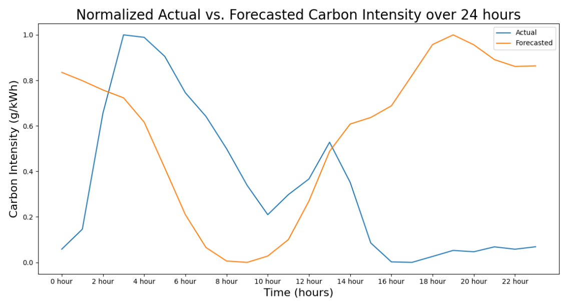

Figure 2. Normalized actual vs forecasted carbon intensity values over a 24 hour period. The data is from the CISO region for May 22, 2023

- Generate parameter averages for each of the weather components and use them to generate a weather forecast file to be used as input to the first-tier models.
- Run the *firstTierTesting* script to generate the day-ahead forecasts and combine fields from the weather forecast generated above.
- Files generated are now passed to the second-tier model using the *secondTierTesting* scripts and the final 96-hour carbon intensity forecasts are obtained.

Running all the steps end-to-end including the data download ones takes about 6 minutes for a single region and date.

$$python3\ run.py < REGION >< date > \tag{2}$$

The above command can be used to run the entire workflow end-to-end. The *<date>* must be in the format *YYYY-MM-DD*.

3.5 Comparison

Figure 2 represents the actual vs forecasted values of carbon intensity for the CISO region on May 22, 2023. The forecasts are not completely accurate which is expected for this independent study since no new models were trained. Additionally, the save models used for the prediction were trained on data from January to December 2020 which makes them outdated for current predictions. The lower accuracy may also be attributed to a lack of normalization performed while training the models.

4 Challenges and Takeaways

The main challenges were finding a reliable data source for the weather data which additionally had a higher rate limit. The previously used GFS weather forecast archive data source [1] took 5 minutes for a single response irrespective of the size of the request and had a hard limit on the number of concurrent jobs that could be submitted. The problem was solved by using the NOMADS weather API endpoint.

The project provided me an opportunity to learn about the importance of carbon-aware systems under an excellent mentor. Moreover, I also got a chance to understand how CEF and carbon intensities for regions are calculated.

5 Conclusion

Carbon-aware systems help reduce the carbon-emissions of workloads by scheduling them at times when maximum energy generated is using renewable sources. To enable these systems, carbon intensity forecasts are necessary. Carbon Cast, a system to predict hourly carbon intensity in the electrical grids, can provide average carbon intensity forecasts for up to 96 hours.

In this project, we extend Carbon Cast to make use real-time electricity and weather data. Additionally, we make the process of generating carbon intensity forecasts an end-to-end one. Using newer sources of data we are able to reduce the sliding-window of data from 3 hours to 1 hour thereby enabling the models with more data.

In the future, this project can be extended to regions other than the U.S. and Europe. Optimizing the machine learning models used in both the tiers will also be a beneficial direction to work upon.

References

[1] [n. d.]. GFS weather forecast archive. https://rda.ucar.edu/datasets/ds084.1/.

[2] [n. d.]. US Energy Information Administration. 2021. https://www.eia.gov/energyexplained/electricity/use-of-electricity.php. Retrieved July 28, 2022 from.

[3] Diptyaroop Maji, Prashant Shenoy, and Ramesh K. Sitaraman. 2022. CarbonCast: Multi-Day Forecasting of Grid Carbon Intensity. In *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation* (Boston, Massachusetts) *(BuildSys '22)*. Association for Computing Machinery, New York, NY, USA, 198–207. https://doi.org/10.1145/3563357.3564079

[4] Jan Pecinovsky and Frank Boerman. [n. d.]. *entsoe-py*.