# HCAI - ML Efficiency of Cancer Survival Prediction

Niklas TSCHEPPE (11808067), Stefan MILLONIG (11719218) and Lena KLAMBAUER (00931218)

Oktober 2024

## Abstract

This report describes a replication study on PiDeeL, a metabolic pathway-informed deep learning model presented in Kaynar et al [9], with the goal of measuring emissions of training machine learning models using CodeCarbon [13]. Two different systems were tested and emissions equivalent to 0.3 - 0.6 kg of CO2 per run were shown.

## 1 Introduction

The goal of this study is to evaluate energy consumption of machine learning models. This measure is relevant to compare CO2 emission levels between models and systems.

It was decided to test this in the course of a replication study of the PiDeeL model as presented in the associated paper [15] [9].

PiDeeL is a metabolic pathway-informed deep learning model. It reduces the parameter complexity by including metabolic pathway information. The PiDeeL paper [9] claims this improves survival analysis and pathological classification performance.

A replication study is the recreation of previously conducted and scientifically published experiments. Such a study does not attempt to improve on the original work. Instead an attempt is made to replicate the environent based on the description of the publication. The results are then compared to the original findings. The purpose of this is not only to detect errors in the original paper, but also to confirm that all parameters were accounted for in the description of the environment and that factors that were not described do not affect the outcome. It is also valuable to examine and describe if and how readers of the original work can apply its findings. [15] [3]

CodeCarbon [13] was used to calculate carbon dioxide emissions.

## 2 Methodology

To be able to measure the cost of training the ML models in the PiDeeL project, it was first necessary to reproduce the results of PiDeeL. The PiDeeL repository [7] provides a guide containing the following steps:

- Installation
- Prediction using pretrained PiDeeL
- Reproduction

This guide was used in the reproduction study described below.

As the second part of this project, the cost of a single execution of training the PiDeeL model on the dataset linked in the PiDeeL repository [7] was evaluated. For this, the CodeCarbon package [13] was used, which provides functionality to measure carbon dioxide emissions of executing Python code.

CodeCarbon expresses the carbon dioxide emissions as kilograms of CO2-equivalents, which are calculated as the product of the carbon intensity of the electricity consumed for computation, and the energy consumed by the computational infrastructure. [11]

## 3 PiDeeL Model [9]

The goal of PiDeeL is to supply surgeons with real time feedback during glinoma removal surgery.

Cancerous and healthy tissue can be discerned via NMR (Nuclear magnetic resonance) spectrum analysis. Manual live interpretation of HRMAS NMR (High Resolution Magic-Angle Spinning) data requires experts and technicians present. This makes it unfit for use during surgery.

Creating deep learning models is difficult due to a lack of HRMAS NMR spectra available. Instead improvement of shallow models is required.

PiDeeL incorporates metabolic pathway information in the model's architecture. Processing HRMAS NMR spectroscopy signals results in knowledge of the metabolites present. Previous models would use these metabolites as input to classify the pathology of the tissue. PiDeeL first applies biological knowledge to translate them into the biochemical pathways they imply. This is then used as the input for the neural network (NN). Therefore the NN does not need to learn relationships between metabolites and biochemical pathways. Simplifying its task reduces parameters, overfitting and irrelevant connections.

From this NN PiDeeL produces not only a classification of the tissue, but also a survival analysis to assist the surgeon in evaluating risk on removal.

The PiDeeL paper claims survival prediction with a c-index of 68.7%. [9]

## 4 Reproduction Study

The reproduction study was conducted following the guide provided in the PiDeeL repository [7]. Attempts at reproduction were made using different setups and operating systems, as well as using the original status of the PiDeeL repository and a later status after the project had been slightly adapted by the original authors due to issues that arose during the reproduction.

### 4.1 First Attempts on Windows

As the reproduction guide does not mention a specific operating system, the first try at reproduction was done on Windows (Microsoft Windows 10 Home 10.0.19045).

There was no problem cloning the repository, but trying to create a conda environment using the provided PiDeeL.yml file resulted in a PackageNotFound error for a lot of the listed packages. Since the guide also mentions the option to download the dependencies manually, this was done instead and the conda approach not investigated further at this time. The listed dependencies were

downloaded and installed using ‘pip install‘ with pip 23.2.1. The package versions were the ones that were chosen by default.

The next step in the guide is running ‘python predict.py –layer 3 –dev gpu‘ in ‘/run‘ to run a prediction of a pretrained PiDeeL model. For this attempt, the ‘–dev‘ parameter was set to ‘cpu‘ instead, to not have to install and set up CUDA. It was assumed that this would not affect the results, only the efficiency of the program.

This resulted in different results than the ones shown in the guide, as well as an InconsistentVersionWarning caused by scikit-learn. Multiple execution runs always resulted in the exact same results, so it was assumed that the model is deterministic without any pseudo-randomness. Trying to resolve this warning by changing the version of scikit-learn used resulted in further compatibility issues and so was abandoned.

There were no problems downloading the data set. Setting up the data to be used by the model required additional steps that were not described in the guide, specifically renaming a folder and setting up the sample folders in a way that was not described in the guide.

Next, the path in the ‘hyper_config.py‘ file was set to the absolute path to the ‘PiDeel‘ project folder. It is necessary to use forward instead of backward slashes in the path string, even when running the model on Windows. Otherwise the path will not be parsed correctly and necessary files will not be found.

The reproduction script was run as described in the guide in ‘/reproduction‘ using ‘python run_reproduction.py‘. Reading in the samples seemed to work correctly with this setup.

Afterwards, a few issues occured:

- Multiple FitFailed Warnings during the training of the Baseline Cox-PH model:
  "10 fits failed out of a total of 110. ...
  numpy.linalg.LinAlgError: Matrix is singular."
- Multiple FileNotFound Errors
- Some errors related to paths specified in the code not using the provided absolute path from hyper_config.py
- The line ‘os.system("../shap_analysis")‘ had to be changed to ‘os.chdir("../shap_analysis")‘ in ‘run_reproduction.py‘. This seems to be a bug in the reproduction script.
- The following assertion was raised multiple times:
  "AssertionError: Torch not compiled with CUDA enabled"
  It is unclear if and how the model can be run on the CPU without CUDA, or if this is even convenient.
- Multiple errors occured in the visualisation step using matplotlib. These seemed to be consequences of the errors occuring earlier in the pipeline.

There were a lot of problems trying to reproduce this project without a conda environment, therefore this approach was abandoned in favour of using a conda environment.

Additionally, trying to avoid using CUDA added some problems.

## 4.2 Issues on WSL and Native Linux & Communication with PiDeeL Authors

Since there were a lot of problems trying to run the reproduction on Windows, it was decided that further attempts would be made on Windows-Subsystem Linux (WSL) and native Linux.

The following list of issues occured both on WSL and native Linux. Since these were or seemed to be related to the the PiDeeL project itself, instead of mistakes during the setup of the reproduction, these were sent to the original authors of PiDeeL to elicit their help.

1. In the guide, under "Prediction using pretrained PiDeeL", ‘python predict.py –layer 3 –dev gpu‘ was used. To get the same results as shown, layer should be set to 4.
2. As opposed to the description in the guide, simple extraction of the zipped data provided was not sufficient. Renaming of files and restructuring of the data/samples folder is necessary.
3. Running predict.py results in a inconsistent version warning for scikit-learn and a suggestion that the results might not be correct because of this.
4. Multiple FileNotFound Errors occured when running run_reproduction.py.
5. Some errors related to paths specified in the code not using the provided absolute path from hyper_config.py occured when running run_reproduction.py.
6. Multiple FitFailed Warnings occured during the training of the Baseline Cox-PH model when running run_reproduction.py:
   "10 fits failed out of a total of 110. ...
   numpy.linalg.LinAlgError: Matrix is singular."
7. The line ‘os.system("../shap_analysis")‘ had to be changed to ‘os.chdir("../shap_analysis")‘ in ‘run_reproduction.py‘.

Additionally, no results were created when running run_reproduction.py. This was probably a consequence of the errors listed above.

The authors of the PiDeeL project were quick to respond and updated the PiDeeL repository to fix most of the issues:

1. The typo was fixed.
2. A description of the data folder structure was added to the guide.
3. The results were checked to be correct, so the inconsistent version warning could be ignored.
4. The problem was fixed by adding placeholder files so that github would track empty folders.
5. The paths were fixed.
6. No changes
7. The bug was fixed.

Additional future and deprecation warnings caused by using a specific version of pandas were assured to not interfere with the results.

## 4.3 Further Attempts & Remaining Issues

After PiDeel had been adapted, further attempts were made to run the reproduction of the PiDeeL program. This was done using both WSL and Google Colab.

### 4.3.1 WSL

In WSL, the following issues remained when running run_reproduction.py:

1. Multiple FitFailed Warnings occured during the training of the Baseline Cox-PH model :
   "10 fits failed out of a total of 110. ...
   numpy.linalg.LinAlgError: Matrix is singular."

2. During the training of the pathological classification models, the following error occurred: "RuntimeError: CUDA error: invalid device ordinal". This seemed to be related to PiDeeL using a different CUDA setup (device) in this step than in the rest of the program.

3. A figure (shap_path.pdf) opened in a new window instead of saving to a file, keeping the program from termination.

The execution did complete until the end and figures of the results were created, except where directly prohibited by the above errors.

Execution on the full dataset took a long time, so it was decided to attempt to use a subset of the data with an approximately similar distribution of labels for debugging until these issues could be resolved. Unfortunately, using this subset of the data resulted in additional Ill-formed matrix warnings, so the results of these runs are not very meaningful, except when it comes to debugging the setup.

#### 4.3.2 Google Colab

Google Colab [6] was also explored as an alternative platform for running the PiDeeL model. Colab offers a convenient environment for running machine learning models without the need for a local setup. This promised resolving compatibility issues experienced on Windows and WSL and would also decrease the running time due to its optimized hardware.

Initially, the standard instructions provided in the PiDeeL repository were followed. This involved downloading the necessary dependencies listed in the requirements.txt file, such as torch, numpy, pandas, torchtuples, scikit-learn, pycox, and matplotlib, and more. Installing these packages directly in the Colab environment using pip was attempted.

Unfortunately, this study encountered issues with the pandas package. The model and scripts provided were designed to work with pandas version 1.5.3. Unfortunately, Google Colab's environment is not fully compatible with pandas 1.5.3. Downgrading to pandas 1.5.3 in Colab led to conflicts with other dependencies, making the execution of the PiDeeL model impossible.

Another attempt was made running the model on a smaller subset of the dataset to speed up the process and reduce the computational load, but the compatibility issue with pandas persisted, rendering Colab unsuitable for the studies needs.

After these trials, it became clear that the incompatibility between pandas and Google Colab's environment prevented the execution of the PiDeeL model, and no meaningful results could be obtained from this platform. As a result, the study was reverted to using local environments.

### 4.4 Second communication with PiDeeL authors & Changes to PiDeeL code

To fix the remaining errors in WSL, further communication with the PiDeeL authors was attemped, but because of the time frame of this project, it was not possible to wait for a reply and / or fixes from their side.

Instead, the following changes were implemented by the authors of this study:

1. The FitFailed warnings during the training of the Baseline Cox-PH model could not be fixed and remain. It is unclear if these warnings are expected or an issue.

2. The "RuntimeError: CUDA error: invalid device ordinal" was fixed by using the same cuda device in this step as in the rest of the program.

3. The shap_path.pdf was fixed to save to file correctly.

After implementing these changes, execution of the PiDeel reproduction script using the full data set did complete with the expected figures being created.

### 4.5 Results of Reproduction

In [9] PiDeeL is compared to classic survival prediction methods Cox-PH and RSF as a baseline [1] [14]. It is also compared to DeepSurv. DeepSurv is a deep learning based survival predictor model. [8]

Running the reproduction script generated plots that show the performance of these models. Figures 1 and 2 shows the figures created by our systems 1 and 2 respectively. They compare performance to the baseline and DeepSurv models as presented by the PiDeeL paper [9].

The most notable difference between the two systems lies in the deviation in the c-index for the 2-layer DeepSurv model. Generally the two results are comparable to each other. A clear difference between either of the models and the original results can be seen. The reproduction does not show the same improvement in performance compared to other models as claimed by the PiDeeL paper. Therefore this study was not able to reproduce the results in a satisfying manner. This might be due to differences in hardware, setup, or other factors.

## 5 Emissions Evaluation

### 5.1 Integrating CodeCarbon

CodeCarbon [13] was installed as an additional package to the previously created PiDeeL conda environment. There were some compatibility issues, as the CodeCarbon package provided by conda was outdated and caused some issues with the newer Python functionality used by PiDeeL. Using the pip package manager to add the package to the conda environment resulted in the newest CodeCarbon version being installed, which fixed these issues.

A CodeCarbon OfflineEmissionsTracker object was added to the run_reproduction.py script. This tracker was set up to use the carbon intensity of electricity used in Austria for its emission calculations.

Each model being trained in PiDeeL, as well as other steps like data loading, visualization of results, and SHAP analysis, was tracked as a separate task, resulting in emissions data for each of these tasks.

### 5.2 Evaluated Systems

Since attempts to run the reproduction script on other operating systems ran into problems, the final emissions evaluation using CodeCarbon was done on two Windows machines using WSL. Both used Python version 3.11.4, CodeCarbon version 2.5.0, and a Conda environment set up using the .yml file provided in the PiDeeL repository. Other factors, like WSL version, CPU model, CPU count, GPU model etc. differed between the two systems. A detailed comparison can be found in the appendix in Table 2.

## 5.3 CodeCarbon Emissions Basis and Calculations

CodeCarbon calculates carbon dioxide (CO2) emissions, expressed as kilograms of CO2-equivalents, as the product of two main factors :

- Carbon intensity of the electricity consumed for computation, quantified as g of CO2 emitted per kilowatt-hour of electricity
- Energy consumed by the computational infrastructure, quantified as kilowatt-hours

### 5.3.1 Carbon Intensity

The carbon intensity of the consumed electricity is calculated as a weighted average of the emissions from the different energy sources that are used to generate electricity, including fossil fuels and renewables. In this evaluation the carbon intensity of electricity in Austria was used, which CodeCarbon specifies as 158.222 g CO2eq/kWh.[11][10]

### 5.3.2 Energy Consumption

Energy consumption is calculated as the sum of energy consumed by CPU, GPU, and RAM.

The Nvidia GPUs' energy consumption was tracked directly using the 'pynvml' library. The GPU power for the total execution was not given by the CodeCarbon output, though an average GPU power of all subtasks was later calculated manually.

For RAM and CPU, the energy consumption was calculated by multiplying the duration of the execution of the program by the power consumption of the hardware.

The total execution duration calculated by CodeCarbon was found to be much too low to be plausible. This is due to a line in the CodeCarbon code base[1], where the start time of the tracker is reset each time tracking of a new subtask is started. It is unclear if this is intentional and caused by using OfflineEmissionsTracker.start()/.stop() and OfflineEmissionsTracker.start_task()/.stop_task() in combination in a way that was not intended, or a bug. To mitigate this, the total duration was calculated manually from results that were created correctly. The plausibility of this approach was verifiable, because enough other outputs were given so that the duration could be calculated in multiple different ways and crosschecked.

For approximating the RAM power, CodeCarbon uses a constant ratio of 3 Watts per 8 GB. This was then multiplied by the duration to calculate the RAM energy consumption.

CPU energy consumption is usually tracked by CodeCarbon using the Intel Power Gadget on Windows and Mac, and using Intel RAPL files on Linux. Due to the use of WSL this was not possible for this evaluation, because CodeCarbon recognises WSL as a Linux system, but cannot find the RAPL files because they do not exist in this setup. Therefore, CodeCarbon switches into fallback mode: It detects which CPU hardware is currently in use and looks up its corresponding thermal design power (TDP) in a database. This resulted in a TDP of 45 W for System 1 [4] and 125 W for System 2 [5]. It is then assumed that 50 % of this TDP will be the average power consumption (22.5 W and 62.5 W respectively)[2]. This is then multiplied by the duration to calculate the CPU energy consumption.

[1] codecarbon/emissions_tracker.py line 480 [12]

[2] This can be mistakenly read as "50 % of a global constant" in [11] due to formatting issues. It was checked in the CodeCarbon code base and "50 % of the TDP" is used.

### 5.3.3 Emissions

The emissions in kg CO2-equivalents were then calculated by multiplying the total energy consumed by the carbon intensity. Additionally, an average emission rate was calculated by dividing the emissions by the duration of the program execution.

A summary of these calculations and results can be found in the appendix in Table 3.

## 5.4 Results of Emissions Evaluation

A comparison of the duration, emission and emission rate results of the two systems can be found in the appendix in Figure 3.

Analysis of the CodeCarbon results shows that running the PiDeel model took much longer on System 1 than System 2. This was expected, since System 2 has 16 CPU cores compared to System 1's 8. On top of that, System 2 has a much more highpowered GPU than System 1. Looking at the total duration provided by CodeCarbon, System 2 took only 59 % of the time of System 1 to complete the run. Averaging the duration of the different subtasks results in System 1 taking 55 % of the time of System 2. A detailed comparison of the different task durations can be found in the appendix in Figure 4.

Conversely, the emissions of System 2 are much higher than those of System 1. This is true for both the total provided by CodeCarbon and when averaging the results for the subtasks, with System 1 only causing 49 % and 59 % of emissions respectively when compared to System 2. A detailed comparison of the different task emissions can be found in the appendix in Figure 5.

Combining these two factors into an emission rate [kg CO2eq / s] shows that System 2 has much higher results here as well. The emission rate of System 1 is only 29 % of that of System 2 when looking at the total, and 31 % when looking at the average over subtasks. A detailed comparison of the different task emission rates can be found in the appendix in Figure 6.

CodeCarbon also provides comparison points with daily life activity to get a better understanding of the amount of emissions generated. The results of this comparison can be found in the appendix in Table 4, and are quite low.

# 6 Conclusions and Discussion

## 6.1 Reproduction Study

The PiDeeL project was reproduced on two systems using WSL. Reproduction on Windows and Google Colab was not possible. Some issues were encountered and resolved thanks to consultation by the authors of PiDeeL [9]. The different performance results when compared to the PiDeeL paper [9] are suspected to be due to differences in the setup.

## 6.2 Emissions Evaluation

The results seem to show a tendency that training a machine learning model on a system with better hardware causes higher total emissions, as well as higher emissions per time unit. Since the training of a machine learning model is in most cases not time sensitive, and is a task that is not done infinitely often for the same model, it seems that putting up with the longer runtime on a lower powered system should be feasible, and therefore recommended to reduce the environmental impact.

To prove the significance of these results, a larger study that includes multiple executions on different systems is needed.

Nevertheless, specifically the environmental impact of training the PiDeeL model looks to be quite low on both systems when compared to other daily activities, like driving a car or watching television.

# A Appendix

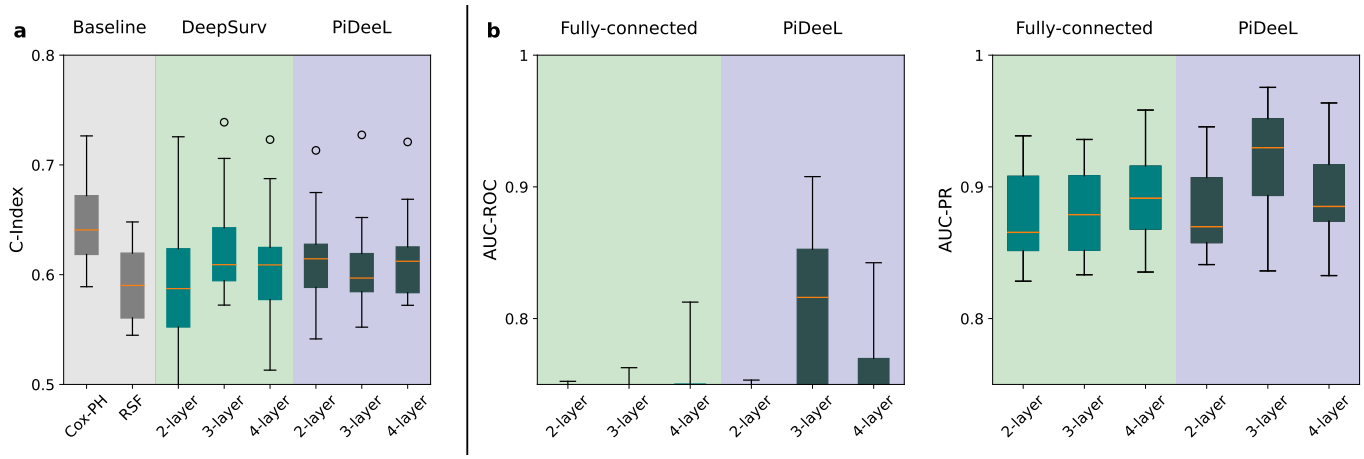


Figure 1: PiDeeL compared to baseline models on System 1 (8 CPU cores, lowpowered GPU)

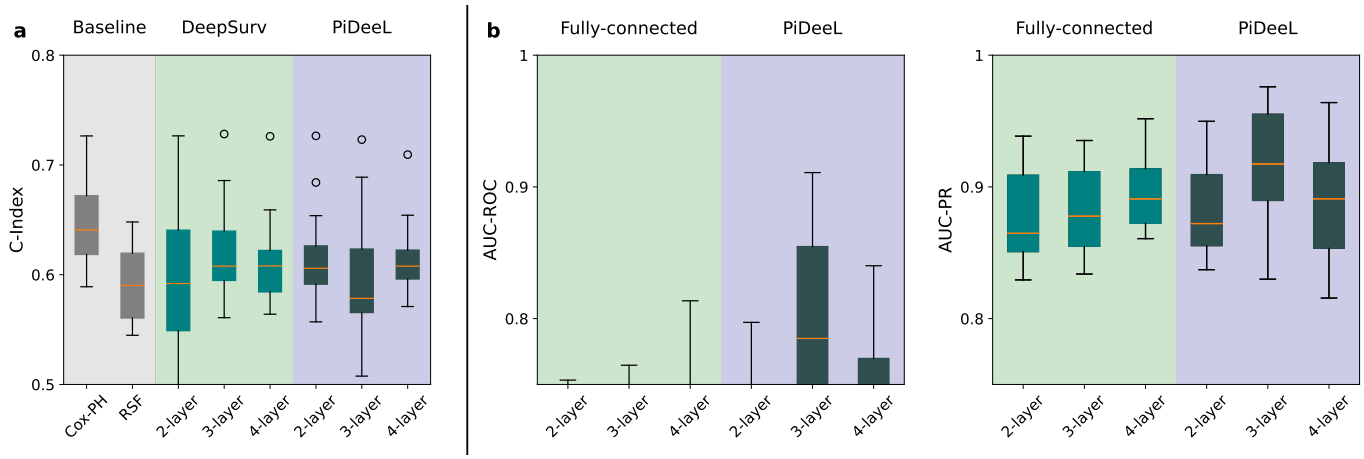


Figure 2: PiDeeL compared to baseline models on System 2 (16 CPU cores, highpowered GPU)

| | |
|---|---|
| **c-index** | time-dependent concordance index as presented by [2] |
| **ROC** | receiver-operating characteristic |
| **Cox-PH** | Cox proportional hazards model |
| **RSF** | random survival forests |
| **AUC-ROC** | area under curve - receiver-operating characteristic |
| **AUC-PR** | area under curve - precision recall |

Table 1: Legend for abbreviations in 1 and 2

| | **Operating System** | **CPU cores** | **CPU model** | **GPU count** | **GPU model** | **RAM size [GB]** |
|---|---|---|---|---|---|---|
| System 1 | Linux-5.15.146.1-microsoft-standard-WSL2-x86_64-with-glibc2.35 | 8 | Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz | 1 | NVIDIA GeForce GTX 1060 with Max-Q Design | 7.71 |
| System 2 | Linux-5.15.153.1-microsoft-standard-WSL2-x86_64-with-glibc2.35 | 16 | 11th Gen Intel(R) Core(TM) i9-11900K @ 3.50GHz | 1 | NVIDIA GeForce RTX 3090 | 15.54 |

Table 2: Comparison between the two systems that were used for the emissions evaluation

| | | | System 1 | System 2 |
|---|---|---|---|---|
| thermal design power (TDP) | [W] | from processor specifications | 45 | 125 |
| carbon_intensity | [g CO2eq/kWh] | from database | 158.222 | 158.222 |
| duration | [s] | tracked directly | 216999 | 127838 |
| cpu_power | [W] | = TDP * 50 % | 22.5 | 62.5 |
| gpu_power | [W] | average over subtasks | 9.475 | 43.551 |
| ram_power | [W] | = RAM size * 3 W / 8 GB | 2.890 | 5.826 |
| cpu_energy | [kWh] | = cpu_power * duration | 1.356 | 2.219 |
| gpu_energy | [kWh] | tracked directly | 0.520 | 1.788 |
| ram_energy | [kWh] | = ram_power * duration | 0.174 | 0.207 |
| energy_consumed | [kWh] | = cpu_energy + gpu_energy + ram_energy | 2.050 | 4.214 |
| emissions | [kg CO2eq] | = energy_consumed * carbon_intensity | 0.32442755 | 0.66681380 |
| emission_rate | [kg CO2eq/s] | = emissions / duration | 0.00000150 | 0.00000522 |

Table 3: Calculation of the total emissions using CodeCarbon

| | **kg CO2eq emissions** | **% of weekly energy consumption of american household** | **km driven** | **hours of television** |
|---|---|---|---|---|
| **kg CO2eq / unit** | **-** | **~74.35 / week** | **~0.13 / km** | **~0.047 / hour** |
| System 1 | 0.324 | 0.24 | 0.04 | 8.9 |
| System 2 | 0.667 | 0.50 | 0.09 | 18.2 |

Table 4: Comparison of PiDeeL emissions with daily life activities

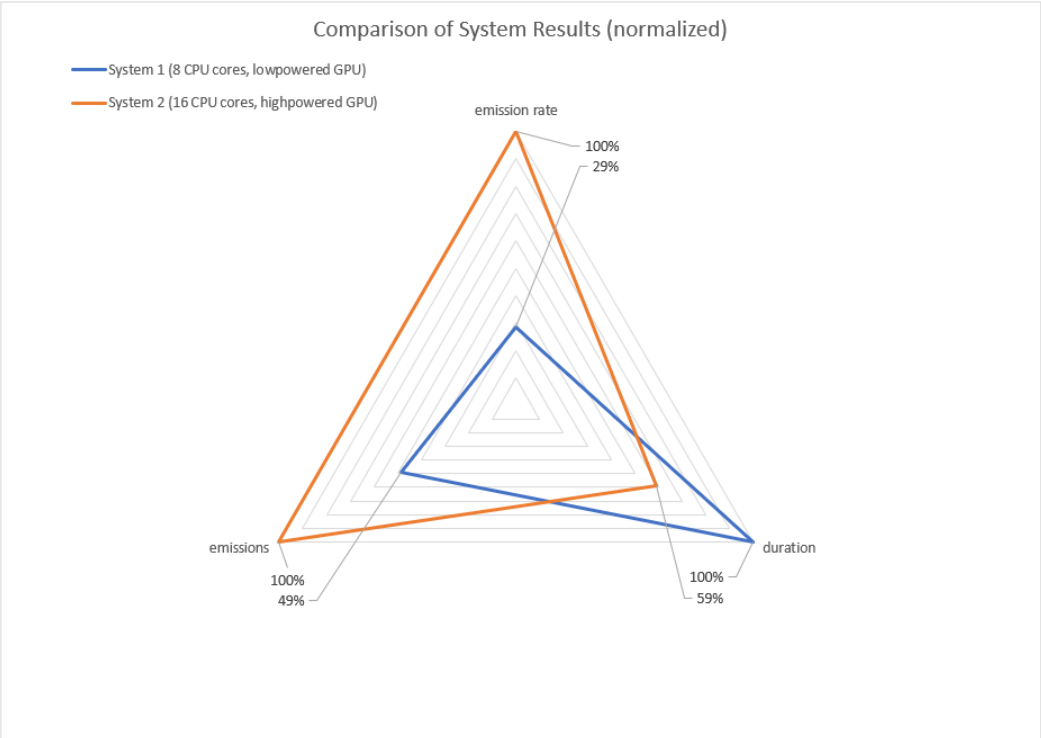


Figure 3: Comparison of system results

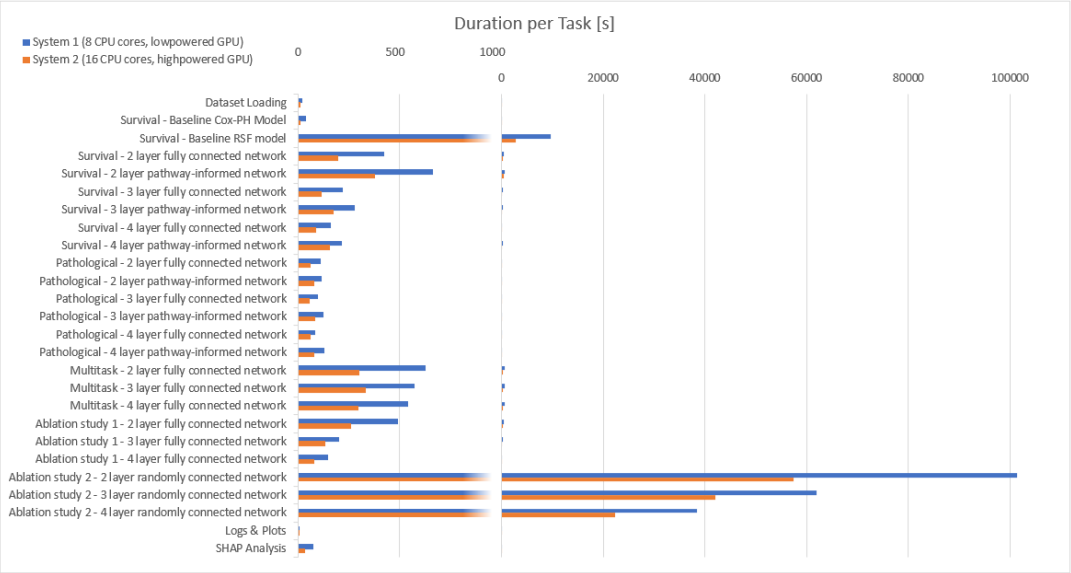


Figure 4: Duration of different subtasks in PiDeeL

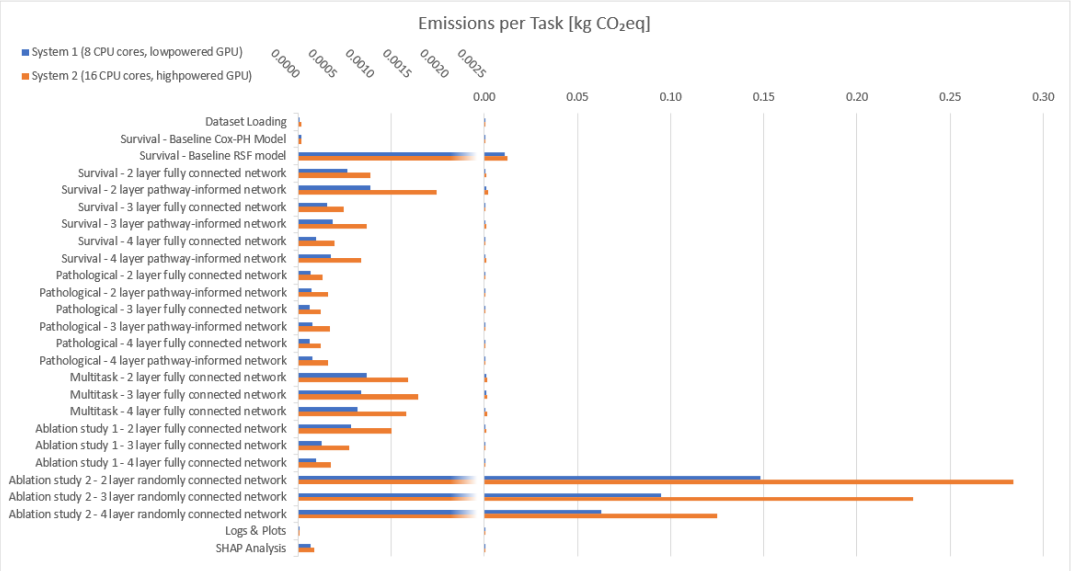


Figure 5: Emissions of different subtasks in PiDeeL

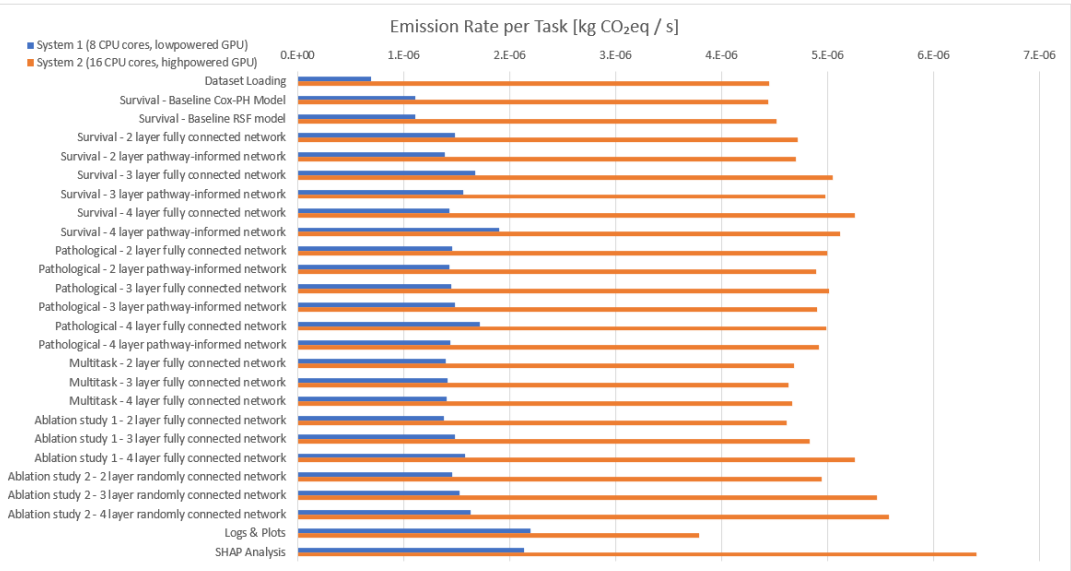


Figure 6: Emission rate of different subtasks in PiDeeL

# References


[1] Medhat Mohamed Ahmed Abdelaal and Sally Hossam Eldin Ahmed Zakria. Modeling survival data by using cox regression model. *American Journal of Theoretical and Applied Statistics*, 12(3):504–512, 2015.

[2] Laura Antolini, Patrizia Boracchi, and Elia Biganzoli. A time-dependent discrimination index for survival data. *Statistics in medicine*, 24:3927–44, 12 2005.

[3] Leonard E. Burman, W. Robert Reed, and James Alm. A call for replication studies. *Public Finance Review*, 38(6):787–793, 2010.

[4] Intel Deutschland GmbH. Intel® Core™ i7-7700HQ Prozessor. `https://www.intel.de/content/www/de/de/products/sku/97185/intel-core-i77700hq-processor-6m-cache-up-to-3-80-ghz/specifications.html`. Last accessed 2024-08-07.

[5] Intel Deutschland GmbH. Intel® Core™ i9-11900K Prozessor. `https://www.intel.de/content/www/de/de/products/sku/212325/intel-core-i911900k-processor-16m-cache-up-to-5-30-ghz/specifications.html?wapkw=11th%20Gen%20Intel(R)%20Core(TM)%20i9-11900K`. Last accessed 2024-08-07.

[6] Google. Google Colab. `https://colab.research.google.com/`. Last accessed 2024-08-07.

[7] A. Ercument Cicek Gun Kaynar. PiDeeL Repository. `https://github.com/ciceklab/PiDeeL`. Last accessed 2024-08-07.

[8] Jared L Katzman, Uri Shaham, Alexander Cloninger, Jonathan Bates, Tingting Jiang, and Yuval Kluger. Deepsurv: personalized treatment recommender system using a cox proportional hazards deep neural network. *BMC medical research methodology*, 18:1–12, 2018.

[9] Gun Kaynar, Doruk Cakmakci, Caroline Bund, Julien Todeschi, Izzie Jacques Namer, and A Ercument Cicek. PiDeeL: metabolic pathway-informed deep learning model for survival analysis and pathological classification of gliomas. *Bioinformatics*, 39(11):btad684, 11 2023.

[10] BCG GAMMA Comet.ml Haverford College MILA. CodeCarbon Global Energy Mix. `https://github.com/mlco2/codecarbon/blob/master/codecarbon/data/private_infra/global_energy_mix.json`. Last accessed 2024-08-07.

[11] BCG GAMMA Comet.ml Haverford College MILA. CodeCarbon Methodology. `https://mlco2.github.io/codecarbon/methodology.html`. Last accessed 2024-08-07.

[12] BCG GAMMA Comet.ml Haverford College MILA. CodeCarbon Repository. `https://github.com/mlco2/codecarbon`. Last accessed 2024-08-07.

[13] BCG GAMMA Comet.ml Haverford College MILA. CodeCarbon Website. `https://mlco2.github.io/codecarbon/index.html`. Last accessed 2024-08-07.

[14] Steven J Rigatti. Random forest. *Journal of Insurance Medicine*, 47(1):31–39, 2017.

[15] Krishna Tiwari, Sarubini Kananathan, Matthew G Roberts, Johannes P Meyer, Mohammad Umer Sharif Shohan, Ashley Xavier, Matthieu Maire, Ahmad Zyoud, Jinghao Men, Szeyi Ng, Tung V N Nguyen, Mihai Glont, Henning Hermjakob, and Rahuman S Malik-Sheriff. Reproducibility in systems biology modelling. *Molecular Systems Biology*, 17(2):e9982, 2021.