Project 4: Green and Cheap Machine Learning Models – DAML

Table of Contents

Team Members

In alphabetical order:

- Baran Deniz Korkmaz
- Bar Tura
- Niklas Kemper

Supervised by Bertrand Charpentier.

Introduction

With the advent of large-scaled machine learning applications boosted by enormous datasets and high computation capabilities, the power consumption of the training of machine learning models reached into serious levels.[1] The reports suggest that the energy consumption of machine learning research started to occupy a significant portion of the annual energy consumption which needs to be paid attention for the environmental purposes. In our study, we aim at proposing a solution which predicts the power consumption of the neural network architectures before running them. Therefore, we target raising awareness towards the issue of global climate change and to bring an incentive for the research community to consider the environmental aspects of their studies. Our work focuses on predicting the power consumption of a neural network architecture when applied on an inference task for a single forward pass of a given batch of inputs. In the second section, we will present some background in the literature and motivation. In the third section, we introduce our main contributions alongside with the methodology we followed. The main contributions of our work includes the largest dataset that can be used for future studies to the best of our knowledge and the proposed methods which approach the task of prediction in different aspects. The fourth section includes the evaluation of proposed methods and compares with the previously published study called NeuralPower[9]. The fifth section includes some additional experiments which were conducted during the development phase and found to be useful. Finally, we conclude our work and briefly mention about the possible future directions.

Related Work & Motivation

Machine learning research and development has drastically increased in volume over the last decade. An analysis of publications on arXiv reveals that the number of papers in AI has increased by a factor of 65 in a timespan of 7 years.

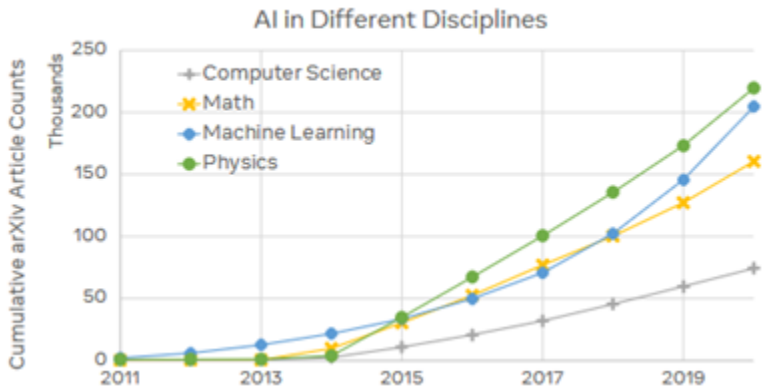


Figure 1. Increase of ML related publications over the years on arXiv [1]

Still, the number of publications does not reveal the full picture. Alongside with the individual project outputs, the model sizes, the data that's used by these models, and the systems on which the model is trained or embodied grow as well.

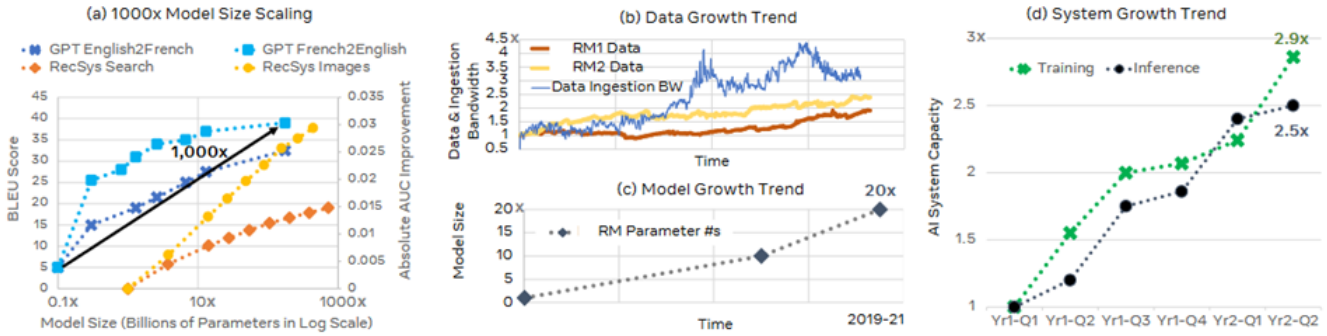


Figure 2. (a, c) Increase in model parameters (b) Increase in data (d) Increase in system size [1]

As the research output increases, the set of metrics used to measure the outcome of research converges to a limited one: commonly including efficiency, accuracy, and errors of suitable shape.

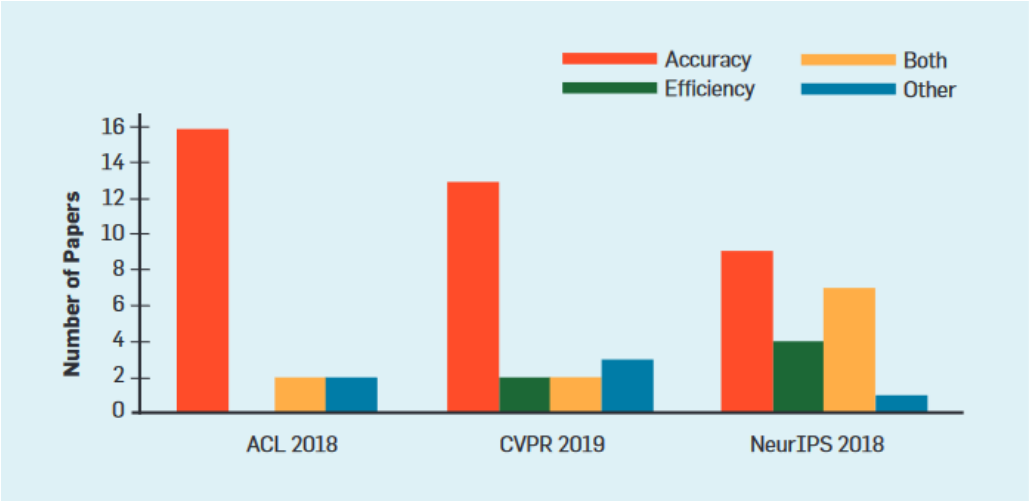


Figure 3. Common metrics used in papers published in ACL18, CVPR19, and NIPS18 [2]

However, these metrics obfuscate the ecological effect. As can be seen from the figure below, it is possible to save four times the energy while losing only 0.0004 model performance measured in one of the common metrics. Therefore, addressing what that four times saving entails is due.

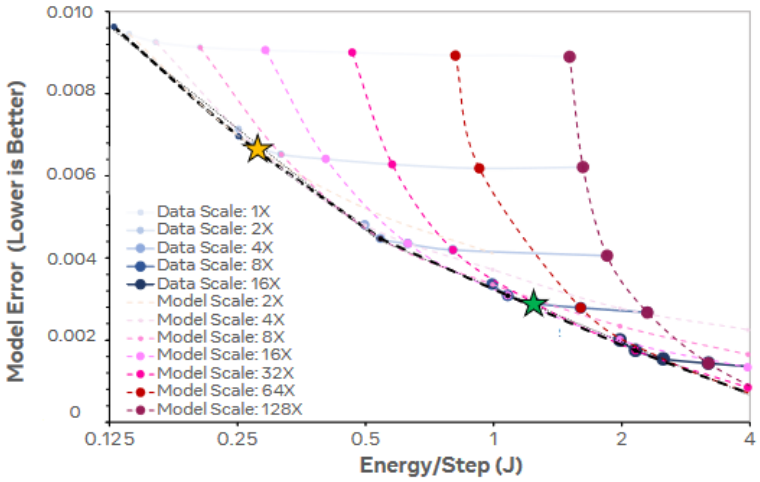


Figure 4. Model Error vs Energy Consumption per Step [1]

Below are the CO2 emission levels of three different models as well as the emission levels of three common products /services. We can see that training RM - 1, a recommender model from Facebook, is the equivalent of running a refrigerator for 3150 years. Training GPT-3 emits a sum of CO2 that is 4.6 times a car would throughout its lifetime. Training BERT with Neural Architecture Search (NAS) results in an ecological footprint equivalent to 3.4 roundtrips between New York (NY) and San Fransisco (SF). It is clear that the CO2 emissions resulting from training neural networks are tremendously high, and need to be accounted for.

Table 1. CO2 Emissions of Training Three Different Models [3]

Model	Training Carbon Emission (Million kg)
RM - 1	0.41
GPT- 3	0.552
BERT - NAS	0.62

Table 2. CO2 Emissions of Three Different Products [4]

Product	Carbon Emission (Million kg)
NY-SF Round trip	0.18
Car (lifetime)	0.12
Refrigerator (yearly)	0.00013

However, this is not limited to training. The figure below lists the CO2 emission levels of several models in terms of their training as well as inference. The values for inference are normalized with the training time, that is, if a model was trained for n GPU days on a single GPU, the associated inference emissions are those that the model would emit in n days running in inference mode. It can be seen that the inference emission can be as large as twice the training emission, as in the case of LM, a language model from Facebook. What's more, a rough approximation yields that, running 1000 models with a similar size to that of RM - 1 for one year results in 30 million kg of CO2, which would be 5% of the greenhouse gas (GHG) effect created by the agriculture industry. Considering that the emissions by the agriculture industry makes up 11% of the total GHG effect, it is possible for a mere total of 1000 models to emit CO2 that is 0.55% of the total GHG emissions. Keeping in mind the reach of ML research outlined previously, inference of 1000 models is a drastic underapproximation of the total effect. As such, it is plausible to assume that this number could easily shoot up to 2-3%. Therefore, the negative externalities that the ML research brings about in the form of CO2 emission must be addressed immediately.

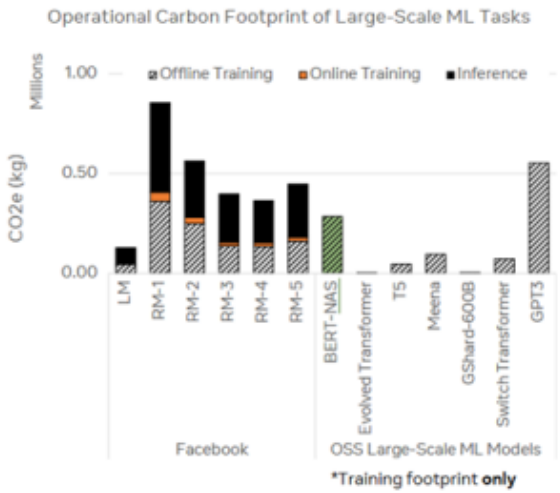


Figure 5. Training and Inference CO2 Emissions of Different Models [1]

There is only a handful of works that aim to assess the carbon emission levels of ML models. Li et al. examine off-the-shelf Deep Convolutional NN models in terms of their energy consumption [5]. The architectures examined by the authors are: AlexNet, OverFeat, VGG, and GoogLeNet. They make use of different accelerators and perform tests on multiple frameworks. The models are examined both at a wholistic and reductionistic level (layer by layer). Notable findings of the work are as follows

- Power consumption and energy consumption are to be observed in parallel, as even though a setting might have higher power consumption, it might significantly reduce training time.
- Usage of cuDNN reduces energy consumption by 42% compared to native GPU utilization.
- Even though mostly idle, CPU accounts for 20% to 40% of the energy consumption.
- CPU consumption is highly affected by the degree of multithreading.
- [For AlexNet] Convolutional layers make up 87% of the consumption, FC layers 10% and activation layers 5%.
- An increase in the batch size will result in higher energy consumption but might reduce energy consumption per image processed.

Strubell et al. focus on common NLP architectures [4]. They train Transformer, ELMo, BERT, and GPT-2 exactly according to the details specified in the respective papers and measure the carbon emissions associated with the trainings. They make use of nvidia-smi and Intel RAPL to gather the power consumption of GPU, CPU, and DRAM during the training. Then, they obtain the energy consumption by multiplying the power readings with the training time. The result of this is then multiplied by an average Power Usage Effectiveness (PUE), which is a measure of how much of the energy consumption goes into the compute process and how much into peripheral processes such as cooling. Having obtained the effective energy consumption, they multiply the result by a rough kWh to CO2 conversion rate provided by the U.S. Environmental Protection Agency. They obtain striking results such as that training BERT on GPU is the equivalent of a trans-American flight, and that training BERT with NAS on 8 P100s has a carbon emission sum four times that of a car has throughout its lifecycle.

Lacoste et al. present a tool for estimating carbon emissions of ML models [6]. Their assumption is that any significant work nowadays is conducted via cloud services. Therefore, they have gathered data regarding CO2 emissions of different grid locations and cross-referenced them with known GPU server locations from the three major cloud providers: Google Cloud Platform, Microsoft Azure and Amazon Web Services. In doing so, they hope to get an approximate quantification of CO2 emissions based on energy consumption. In wake of seeing the variance in CO2 emitted per kWh, they propose two main actions. First, choose cloud providers wisely: Some providers have transitioned to more environment-friendly options such as renewables. Second, choose data center location: The availability of environment-friendly compute is bottlenecked by the infrastructure of the country in which the data center is located.

Henderson et al. also present a tool for CO2 emission quantification [7]. Whereas the former tool bases the estimation merely on type of GPU, runtime, service provider, and data center location, the latter work by Henderson collects a wide variety of information ranging from GPU model to memory usage to disk write speed. These data are used to get a refined estimate of the energy sum (by means of nvidia-smi etc.). The obtained sum is then multiplied with the carbon intensity of the local energy grid, the information on which is obtained through open sources. Following that, they propose the following actions to be taken:

- Forming of energy efficiency leaderboards
- Running the experiments on carbon-friendly regions
- Overall green approaches by the industry leaders to set environment-friendly standards
- Being mindful of whether the gains to be obtained by the experiment is worth the associated natural costs.

Methodology

There are two possible ways to assess the carbon emission levels of a given neural network. The first one is to make use of hardware monitoring packages such as Intel RAPL or Nvidia-smi. Given any process, these packages provide the power consumption in real time, which can be multiplied with time elapsed to obtain the energy expenditure. The second option is to train a predictor on a dataset consisting of neural network architectures and their energy consumption levels, and afterwards inductively predict the energy consumption of any given architecture.

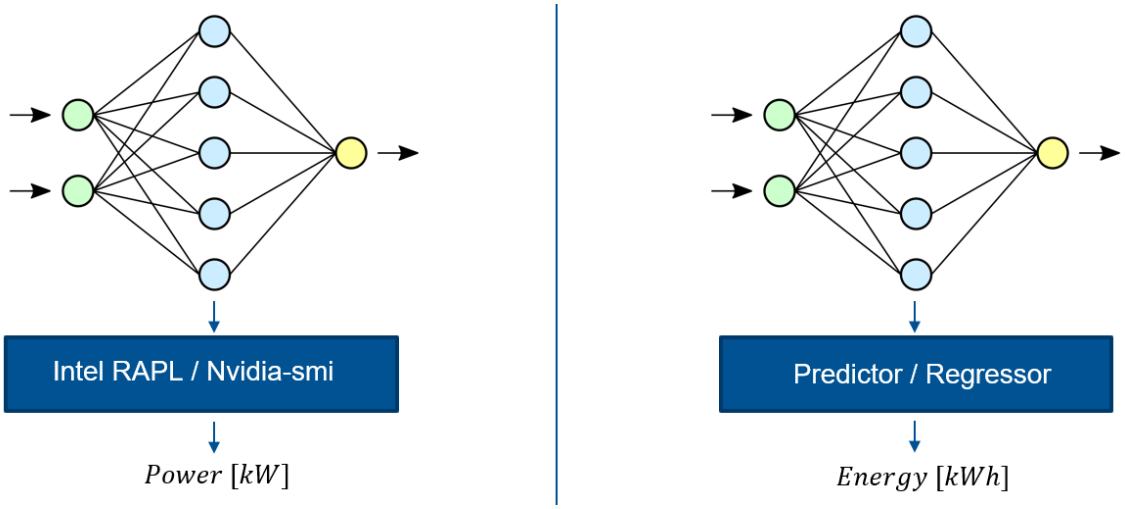


Figure 6. Two options to assess the carbon emission levels of neural networks

Here, we focus on the latter option, as we would like to know the carbon emission before runtime. Put concretely, we aim to construct a predictor model that takes in as input any neural network architecture, and predicts the energy consumption levels thereof, which is directly correlated with carbon emission levels. Furthermore, we focus only on inference, that is, the forward pass, as we believe the ecological effects of training are like to be outweighed by those of inference in the long run.

To this end, we make use of several tools. Firstly, we operate only on architectures built using TensorFlow. Second, we utilize an energy measurement tool called codecarbon to build our dataset [8]. Finally, we populate different architectures from the Keras Application Zoo.

The challenges associated with this problem can be listed as follows.

- There is no dataset pairing architectures with energy consumptions. One needs to be created from scratch.
- The configuration space of neural networks is quite large. It is virtually impossible to address each dimension of this space.
- TensorFlow introduces an abstraction layer. Models are created on a high-level, but operations are carried out in high dependency to lower level (CUDA, cuDNN). Moreover, there are many factors and operations hidden from the user, such as garbage collection, memory access, etc. These cannot be easily incorporated into the predictor model.

Dataset Generation

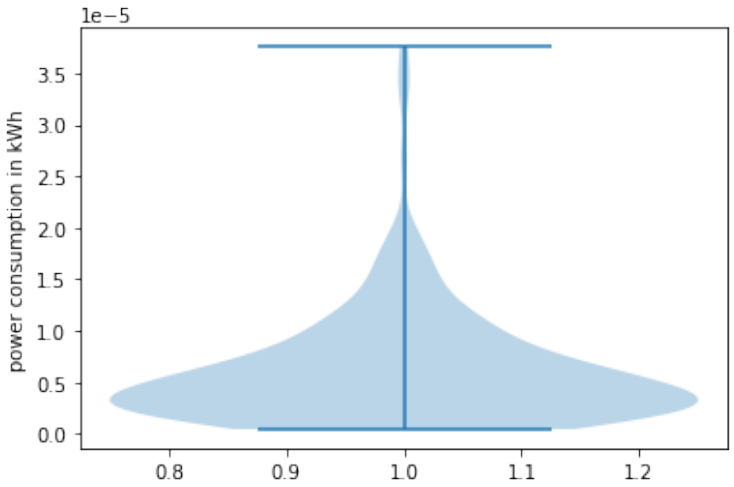
Dataset Collection

In order to train the energy predictors it was necessary to create a dataset with different machine learning models and their energy consumption for a single forward pass. For this we measured the power consumption of multiple forward passes through a model on randomly generated input on a gpu with the measuring tool CodeCarbon and averaged them. In addition, we also measured the energy consumption of each layer in the model separately and isolated from all other layers, again on random input.

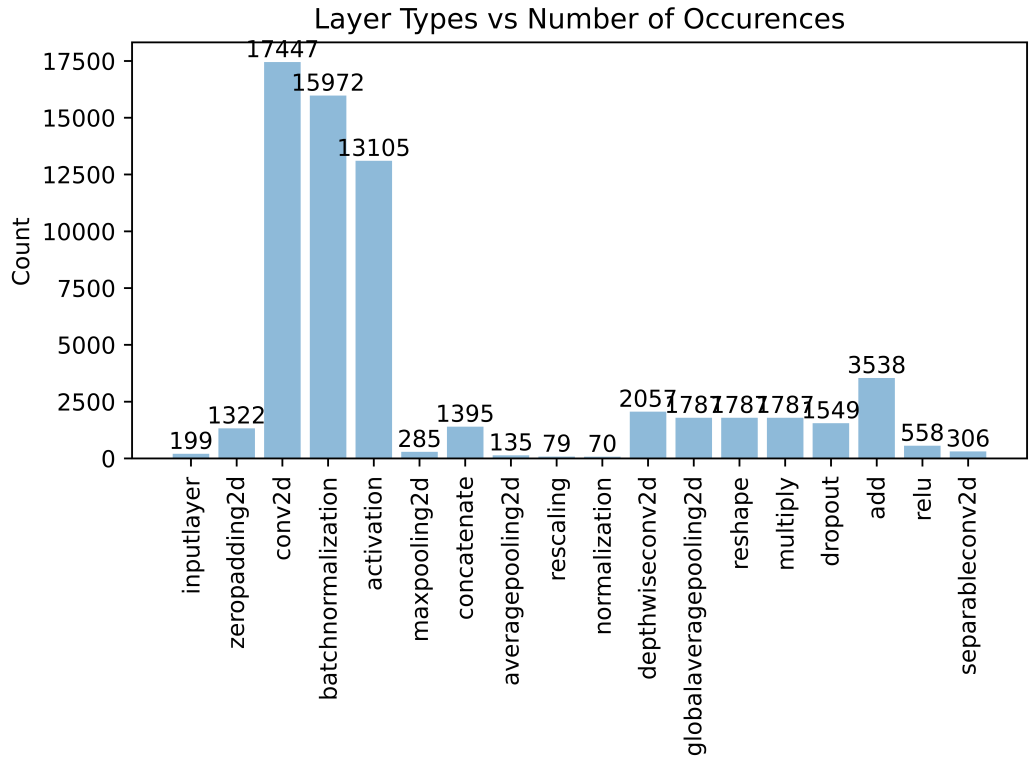
The dataset consists of the pretrained neural networks available in tensorflow evaluated for different input sizes and batch sizes. Because of computational resource constraints we could not isolate the nodes for measurements on these models. Additionally, we were only able to run 500 forward passes for each model and for each layer. To be able to evaluate the models on different input sizes we needed to add fully connected layers to the models (as all other fully connected layers are dropped by tensorflow when executing the model for an input size that is different from the input size it was built for). In total we evaluated 22 pretrained models for three different batch sizes (16,32,64) and three different input sizes (300x300x3, 400x400x3, 500x500x3).

Dataset Analysis

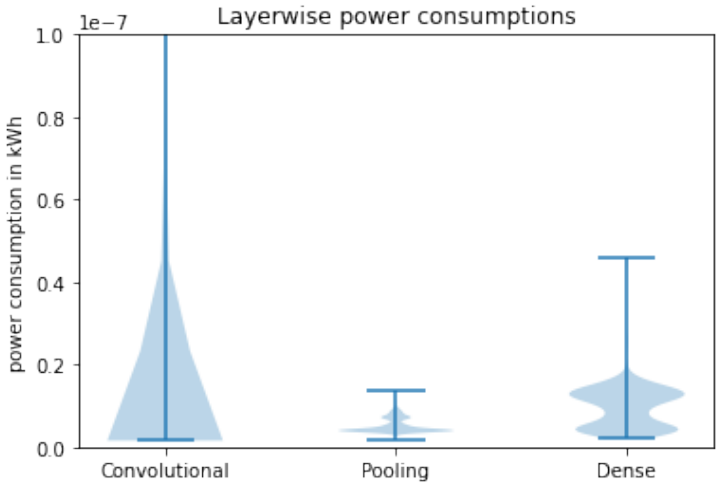
In the dataset the (model-wise) power consumptions range from 4.6e-7 to 3.8e-5:



The following diagram shows the number of occurrences of different layer types in our dataset:



As one would expect, the different layer types have different energy consumptions (visualised for the most important layer types):



Proposed Methods

The problem of predicting the power consumption of a neural network architecture can be tackled in two different approaches. The first approach is to divide a given neural network architecture into its layers and making predictions on layer level. The second approach aims at making the prediction for the overall power consumption of a given model directly. This section introduces the proposed methods for both approaches and in the next section we are going to present how they perform on the overall power consumption of a neural network architecture.

1.Layer-Wise Approaches

The layer-wise approaches suggest that the distinct operations, i.e. the layer types, are handled separately from each other. For each type of layer, a separate prediction model is trained on the training dataset which contains the instances of layers of the corresponding type. In that sense, let us first start with the mechanism behind creating the datasets for each layer type. First, we split the dataset into training and test splits in order to separate the layers that are contained in the models of the test set from the rest. Later, we use those splits for extracting the type of layers of interest. By extracting the layer types, we mean creating the feature vectors which represent the characteristics of a layer. In summary, we extract the feature vectors from the layers of a given type from the models that are contained in the training set. The model is trained in order to predict for a given layer type using the aforementioned extracted feature vectors and corresponding target power consumption values of the layers. This process has been repeated for all available types of layers in the dataset for generalizing the overall power consumption of a model. During the test time, we predict for layers of the models in the test set sequentially by using the previously trained models for distinct layer types in order to form the final prediction for the overall model consumption. The features extracted for each layer type in order to create the dataset for each type of layer are summarized in the table below.

	Features											
Layer Type	**Batch Size**	**Number of Layers**	**Input Size**	**Output Size**	**Hidden Size**	**Number of Filters**	**Kernel Size**	**Stride**	**Pooling Size**	**Padding**	**Target Shape**	**Rate**
Dense	X	X	X	X	X							
Convolutional	X	X	X	X		X	X	X				
Pooling	X	X	X	X		X (Default=1)		X	X			
Input Layer	X	X	X									
Padding	X	X	X	X						X		
Normalization	X	X	X	X								
Activation	X	X	X									
Rescaling	X	X	X									
Reshape	X	X	X								X	
Dropout	X	X	X									X
Add	X	X		X								
Multiply	X	X		X								
Concatenate	X	X		X								

We used three ensemble-learning based methods as layer-wise approaches which are CatBoost [10], XGBoost [11], and Random Forest. The reason why we used ensemble-learning based methods is to take advantage of their strength in generalization. In addition, the number of features forming a feature vector representing a type of layer is low for training a neural-network based method. In our experiments, even though we tried out different methods such as Gated Residual Networks or Residual Connections, the neural network had difficulty to learn such a large dataset with a small number of available features. However, the ensemble-learning based methods mentioned above performed surprisingly well in spite of low number of features available in the dataset for representing layers. One final remark is that we used polynomial regression method as a baseline for our studies. Please refer to our wiki page Week 7&8 - Initial Model-Wise and Layer-Wise Proposals to see how polynomial regression performs for a smaller subset of our dataset and how we decided to use it as a baseline by comparing our results with the results by Cai et al (2017) [9].

2.Model-Wise Approaches

Lorem Ipsum

Evaluation

This section brings everything into together by presenting how layer-wise and model-wise approaches perform on predicting the overall power consumption of a neural network architecture. The table below summarizes our results for both layer-wise approaches and model-wise approaches. Since layer-wise approaches are trained in a layer-wise fashion, the table include their performance for different types of layers as well.

Large Dataset	**RMSPE (%)**	
	Layer-wise Models	**Model-wise Models**

Layer Type	Polynomial Regression	NeuralPower [9]	CatBoost	XGBoost	Random Forest	MLP	RNN	Embedded-RNN
Dense	29.7	7.08	25.8	24.4	24.9			
Convolutional	132.05	10.23	17.8	18.9	14.2			
Pooling	38.83	7.37	22.2	17.1	14.75			
Input Layer	24.9		23.7	18.1	11.9			
Padding	65.5		31.6	21.2	28.1			
Normalization	30.9		15.1	14.3	13.06			
Activation	87.3		17.7	17.09	13.1			
Rescaling	18.9		14.7	10.5	9.2			
Reshape	24.7		21.7	20.7	21.1			
Dropout	25.6		9.7	9.7	9.6			
Add	29.3		8.8	8.8	8.7			
Multiply	35.9		13.6	14.2	11.3			
Concatenate	29.03		17.03	14.5	9.6			
Overall Model	**-**	**-**	**15.15**	**14.15**	**12.26**	**47%**	**41%**	**56%**

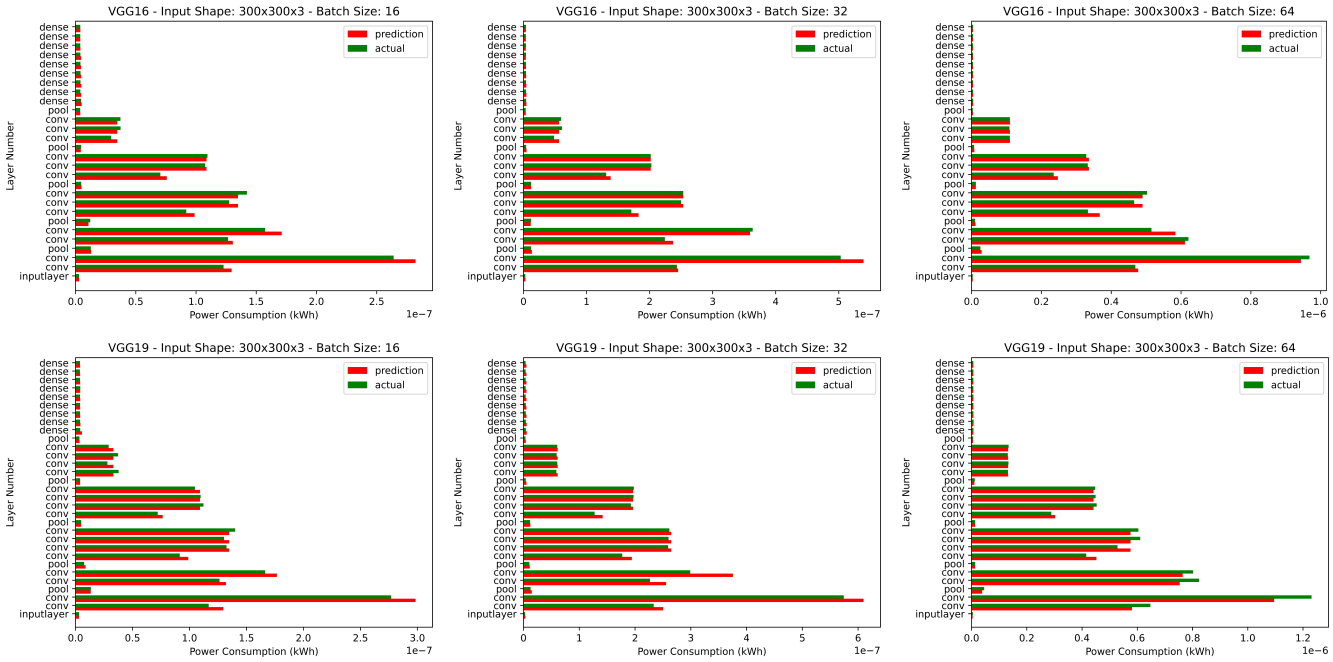
For the evaluation of our models, we used the metric of **Root Mean Square Percentage Error (RMSPE)** since it measures the percentage of the deviation from the target value. Therefore, it is robust against different scales of target values since it measures the deviation as percentage. Another advantage is that it enables comparing the performance with NeuralPower [9] since they presented their results in this metric. Below, you can find the formulation for RMSPE loss.

$$RMSPE = \sqrt{\frac{1}{n}\sum_{i=1}^{N}(\frac{y_i - \hat{y}_i}{y_i})}$$

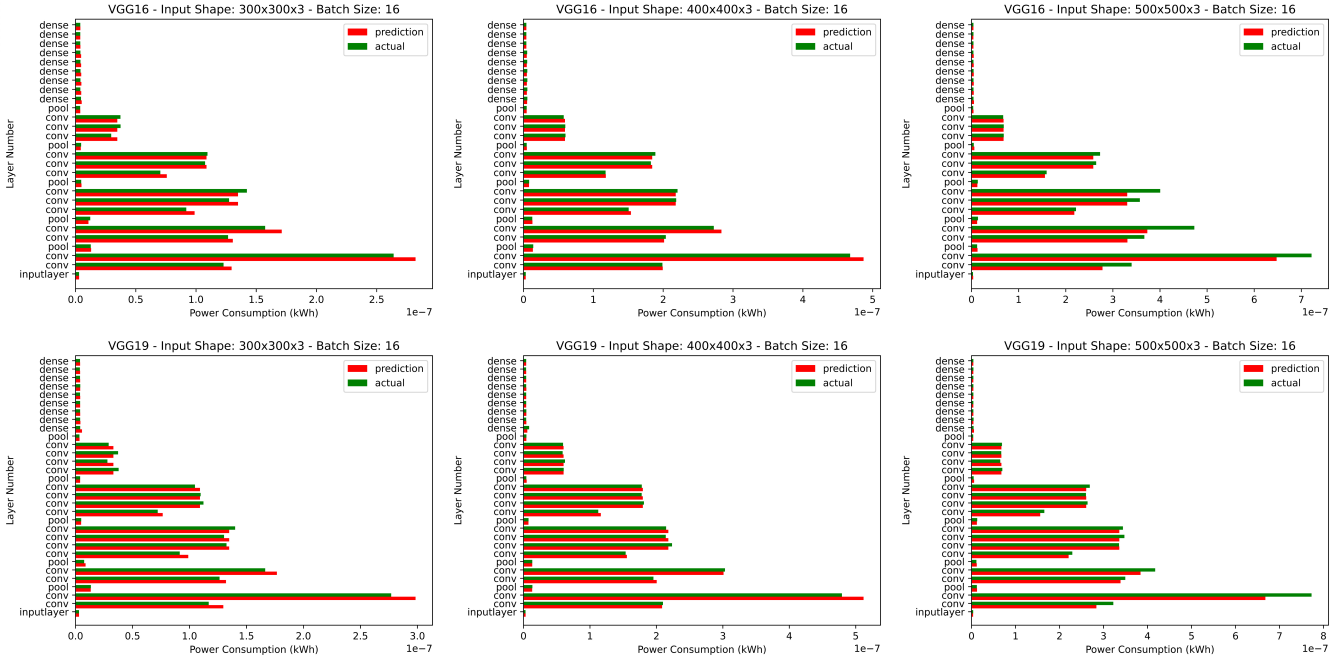
In order to enable a better comparison, let us add one final table which shows the sizes of the datasets used by our work and the study of NeuralPower [9].

Datasets	Ours			NeuralPower [9]
Type of Layer	**Training**	**Test**	**Total**	**Total**
Activation	11070	2593	13663	
Add	2937	601	3538	
Concatenate	1027	368	1395	
Convolutional	15850	3960	19810	858
Dropout	1208	341	1549	
Fully Connected (Dense)	1431	360	1791	116
Input Layer	159	40	199	
Multiply	1379	408	1787	
Normalization	12901	3141	16042	
Padding	1113	209	1322	
Pooling	338	82	420	216
Rescaling	61	18	79	
Reshape	1379	408	1787	

First, we would like to start into our discussion with the evaluation of layer-wise approaches. The results suggest that the ensemble-learning methods generalize well for the unseen data on the prediction of the power consumption of a neural network architecture even though the size of the dataset that we have is extremely large compared to the study of NeuralPower. [9] The poor generalization of polynomial regression method on layer-wise predictions is another evidence that their generalization might fail for large datasets which contain more configurations of different neural networks. We can make this conclusion because we already shown in our results (Please refer to Week 7&8 - Initial Model-Wise and Layer-Wise Proposals) that if we use a dataset of similar sizes as of NeuralPower, the results look similar to each other. However, in a wider context, the ensemble-learning based methods perform much better than the methods analyzed so far in terms of generalization. Their performance is more robust to the different configurations applied to the models in the dataset. Among these methods, the Random Forest, performs slightly better than CatBoost and XGBoost as it can be seen in the table above. The model-wise approaches, however, cannot perform as well as the layer-wise approaches. It is understandable since the layer-wise approaches are trained separately on the power consumption values of each layer instance. However, still we should note that the performance of layer-wise approaches on the generalization on the unseen layers shouldn't be ignored since it is challenging to achieve. The first conclusion is that the layer-wise approaches perform better as they are trained on the power consumption values of each layer forming a model. Second, the Random Forest performs slightly better than the other proposed ensemble-learning based methods with an RMSPE of 12.26%. Next, we want to evaluate the performance of Random Forest on the prediction of the power consumption of the layers in a model deeper. Afterwards, we would like to explicitly compare its performance with CatBoost and XGBoost by visualizing their predictions for the overall power consumption of the models in the test set. Let us also note that we are going to focus deeper on the evaluation of layer-wise approaches as they outperform the model-wise approaches because of the abovementioned reasons.



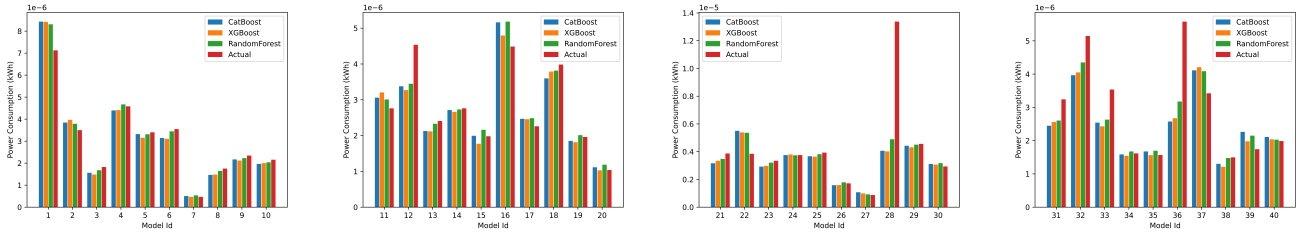
Caption: Layerwise power consumption predictions by Random Forests on VGG16 and VGG19. The input shape is fixed as 300x300x3 and the batch size vary from 16 to 64.



Caption: Layerwise power consumption predictions by Random Forests on VGG16 and VGG19. The batch size is fixed as 16 and the input shapes vary from 300x300x3 to 500x500x3.

We would like to remind that the architectures of VGG16 and VGG19 have been followed by fully-connnected layers of different hidden sizes in order to populate the number of fully connected layers in the dataset and to enable the use of different input shapes since the final dense layers are dropped when you customize the input shape. First, we should observe that the power consumptions of dense layers are similar even though we drastically changed the hidden sizes ranging from 4096 to 10. We think that it stems from the fact that the modern GPUs can manage to optimize the vector based affine transformations efficiently. Please observe that the power consumption values increase with respect to the increase in the input shape and batch size as well. The model performance decreases as the input shape increases in general. In the first plot, we significantly observe that the performance on both VGG16 and VGG19 decreases when the input shape is 500x500x3. It is possible to make the same conclusion for the batch size as well. We think that it comes from the fact that even though there is a linear fashion in the power consumption with respect to the input shape and the batch size, it is not perfect for generalization. Even though all these key remarks on the performance, it is fair to conclude that the Random Forests work surprisingly well in general. For example, for the family of VGG when they are customized by the abovementioned input shapes and batch sizes (18 models in total), the error value of RMSPE is 4.8% which is significantly good.

Finally, the following plot shows the performance of our proposed methods for predicting the overall consumption of a model.



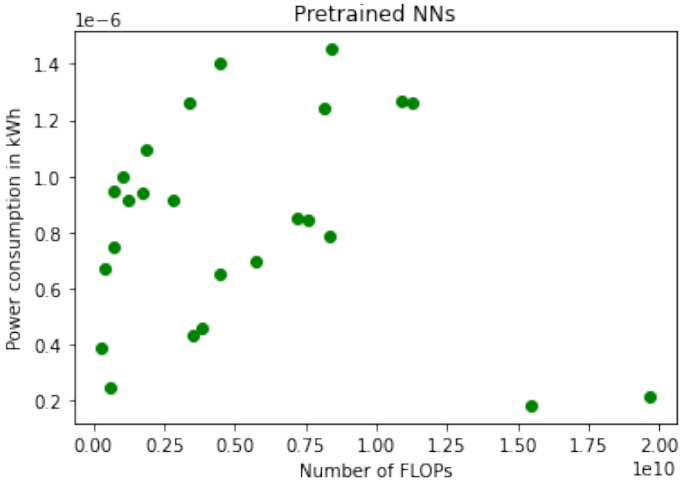
Caption: The predictions of different proposed methods on the overall power consumption of models in the test set compared to their ground truth consumption values measured by CodeCarbon.

The plot above indicates that the proposed methods perform well in general for predicting the overall consumption of a model when the predictions made for each layer are summed up to form the prediction for the overall power consumption for a given model. We observe that some model instances such as model with id 12, 28 and 36 in the test set act as an outlier in the test set. It stems from the fact that it is not possible to perfectly measure the power consumption by CodeCarbon if you don't isolate the GPU on which you perform the measurements from the rest of the cluster. Thus, we decided to leave those models from our test set since they pose obvious measurement errors. We would like to note the RMSPE error values for CatBoost, XGBoost, and Random Forest as 20.54%, 19.86%, 17.3% respectively before the outliers were removed from the test set. In general, all of the proposed methods work well for predicting the overall energy consumption for an unseen model. The best method is the Random Forest which was trained by MSE loss and it performs an RMSPE of **12.26%** on the large dataset. One final remark is that the proposed method, Random Forest, is easy and fast to train which makes it eco-friendly compared to the deep-learning based complex methods.

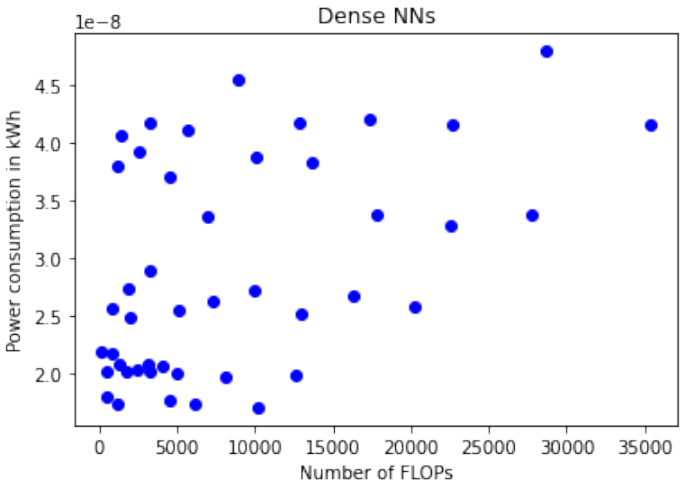
Further Experiments

Correlation Between FLOPs and the Power Consumption

A common metric in ML research is the number of FLOPs of a model. Unfortunately, as [] have shown the number of FLOPs is not (perfectly) correlated with the energy consumption. In order to recreate this result, we evaluated the number of FLOPs and the power consumptions (on GPU with batch size 1) for the pretrained networks available in tensorflow:



The number of FLOPs is even for very simple models consisting only of fully-connected layers not perfectly correlated with the energy consumption:



We conclude that it is not enough to only consider the number of FLOPs in order to predict the energy consumption.

Additivity Factor

A very important question for the choice of a predictor was whether we can assume layer additivity:

> **Layer Additivity**
>
> The power consumption of a forward pass through a model is equal to the sum of the power consumptions of each layer, measured individually.

Layer additivity can only be true if layer independence is true and if there is no additional overhead between layers.

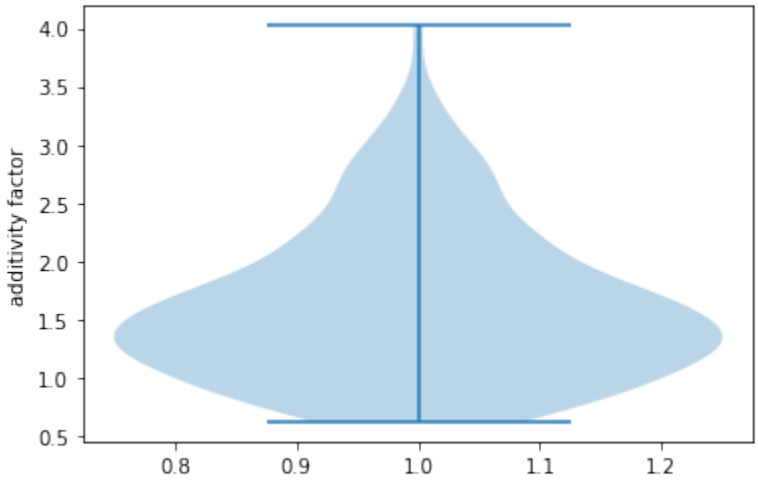
> **Layer Independence**
>
> The power consumption of a forward pass through a certain layer is *not* influenced by other layers.

In order to see if our dataset fulfils the layer additivity assumption, we defined an additivity_factor:

> **Additivity Factor**
>
> The additivity factor of a model consisting of layers 1,…, is defined as
>
> $$\text{additivity\_factor}(M) = \frac{P(M)}{\sum_{l \in M} P(l)}$$
>
> where () denotes the measured power consumption of a forward pass through .

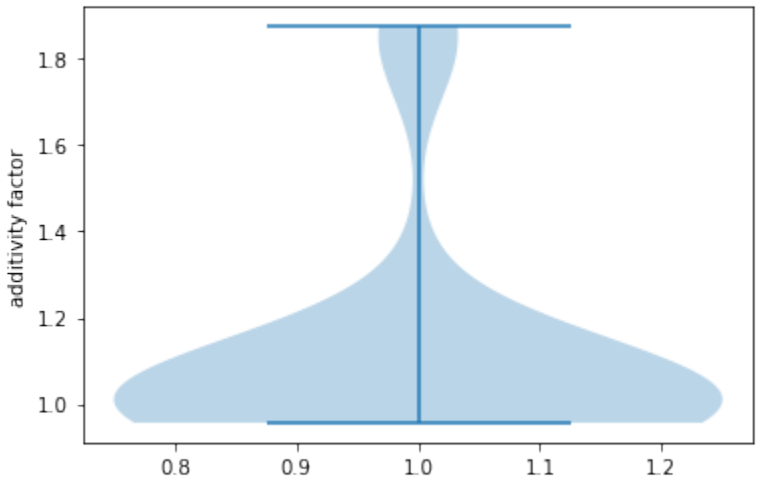
So if the additivity factor is close to 1, then the layer additivity assumption is approximately true.

The distribution of the additivity factor in our dataset looks as follows:



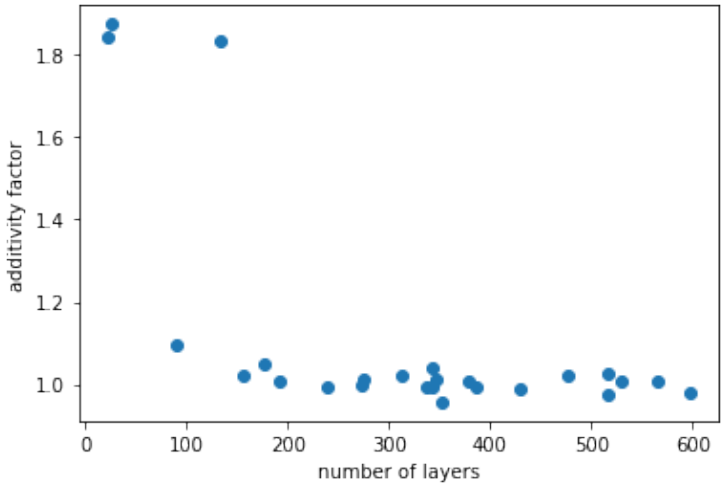
So the additivity factor ranges up to 4!

We repeated the experiment for a smaller dataset (same pretrained models but with batch size of 1) this time isolating the node and increasing the number of forward passes (over which the energy consumption is averaged):



One can see that the additivity factor (apart from a few outliers) ranges from 1 to 2. This means that the power consumption for a forward pass through the complete model is consistently higher than the sum of power consumptions for forward passes through the layers, individually. So, the whole is more than the sum of its parts (but only up to 2 times more).

Another interesting result is that the additivity factor approaches 1 with a higher model complexity:



Hence, we conclude that the additivity assumption is (approximately) true at least for complex models. The higher additivity factors in our dataset can probably explained by issues with our measuring setup (in our dataset we could not isolate the GPU nodes and only averaged over 500 forward passes). Unfortunately, because of computational resource limitation we could not repeat the measurements for our large dataset with more forward passes on isolated nodes. Hence, we just assumed that the layer additivity assumption is also true in our dataset and replaced the model-wise energy consumption by the sum of layer-wise power consumptions.

Conclusion and Future Outlook

The global climate change poses a huge threat for the sustainability of the environment of which will have more intense costs for the next generations. With the release of much large-scaled datasets and the higher computation capabilities the development of more complex machine learning models have been made possible which brings much larger energy consumption trends. In order to raise awareness in the research community and to incentive eco-friendly studies, we tried to derive a solution which would predict the power consumption of a neural network architecture before running it. We collected our own dataset using CodeCarbon as power measurement tool by running different input and batch size configurations on the previously trained models available in Keras. [10] We decided to tackle the problem of predicting the overall power consumption of a neural network architecture in the aspects of layer-wise fashion and model-wise fashion. The layer-wise approaches outperforms the model-wise approaches as they are trained on the power consumption of each layer forming a model separately. Finally, we observed that Random Forest, which is an ensemble-learning method, performs the best results with an RMSPE of 12.26%. It is an eco-friendly solution which is easy and fast to train, which generalizes well on the unseen test data as well. The possible future directions of the study might be that the proposed models can be trained by the computation graph itself on an operation-wise fashion rather than on the features of the layers. Additionally, the isolation of the GPU can provide more consistent consumption measurements, which would increase the performance of the models and would enable a better evaluation.

References

[1] Wu, Carole-Jean, et al. "Sustainable ai: Environmental implications, challenges and opportunities." Proceedings of Machine Learning and Systems 4 (2022): 795-813.

[2] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green AI. Commun. ACM 63, 12 (December 2020), 54–63. https://doi.org/10.1145/3381831

[3] Patterson, David, et al. "Carbon emissions and large neural network training." arXiv preprint arXiv:2104.10350 (2021).

[4] Strubell, Emma, Ananya Ganesh, and Andrew McCallum. "Energy and policy considerations for deep learning in NLP." *arXiv preprint arXiv:1906.02243* (2019).

[5] Da Li, Xinbo Chen, Michela Becchi, and Ziliang Zong. “Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs”, 2016, https://ieeexplore.ieee.org/document/7723730

[6] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, Thomas Dandres. “Quantifying the Carbon Emissions of Machine Learning”, 2019, https://arxiv.org/abs/1910.09700

[7] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, Joelle Pineau. “Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning”, 2020, https://arxiv.org/abs/2002.05651.

[8] CodeCarbon [https://codecarbon.io/]

[9] Ermao Cai, Da-Cheng Juan, Dimitrios Stamoulis, Diana Marculescu. "NeuralPower: Predict and Deploy Energy-Efficient Convolutional Neural Networks". 2017. https://arxiv.org/abs/1710.05420. (Accepted as a conference paper at ACML 2017)

[10] Keras [https://keras.io/api/applications/]