

Hyperparameter Power Impact in Transformer Language Model Training

Lucas Høyberg Puvis de Chavannes 

IT University of Copenhagen
M47 Labs
lucas.puvis@m47labs.com

Mads Kongsbak 

IT University of Copenhagen
mkon@itu.dk

Timmie Mikkel Rantzau Lagermann 

IT University of Copenhagen
timm@itu.dk

Leon Derczynski

IT University of Copenhagen
ld@itu.dk

Abstract

Training large language models can consume a large amount of energy. We hypothesize that the language model’s configuration impacts its energy consumption, and that there is room for power consumption optimisation in modern large language models. To investigate these claims, we introduce a power consumption factor to the objective function, and explore the range of models and hyperparameter configurations that affect power. We identify multiple configuration factors that can reduce power consumption during language model training while retaining model quality.

1 Introduction

Large language models have pushed the boundaries of accuracy and performance in various NLP tasks, at the cost of energy efficiency. This is due to the increasing amount of compute time and power needed to train these models (Amodei, 2018), thus increasing the amount of energy the computers training the models need to consume.

The Robustly Optimized BERT approach (RoBERTa) (Liu et al., 2019) achieved this by improving the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) in multiple ways, such as increasing the training time through more epochs and a larger amount of data, with BERT already requiring 1507 kWh of electricity and emitting 652 kg of CO_2 . Other strategies, such as NAS, an English to German machine translation model, consumed 656,347 kWh of electricity, corresponding to 248,019 kg of CO_2 (Strubell et al., 2019). While these models show great potential, it comes at the cost of high CO_2 emissions. The core issue is that the electricity consumed is not guaranteed to be environmentally friendly, and often comes from sources such as coal or gas. According

to Strubell et al. (2019), we must cut CO_2 emissions by half to slow natural disasters. However, much research in the field ignores the perspective of energy efficiency. When looking at papers from three top AI conferences, namely ACL, NeurIPS, and CVPR, work tends to focus on accuracy rather than efficiency, or a mixture (Schwartz et al., 2019).

An added benefit to developing more energy-efficient models is a reduced barrier of entry to NLP research. Researchers with good ideas may not be able to execute those ideas, given that state-of-the-art results are locked behind large-scale compute (Strubell et al., 2019; Bender et al., 2021).


This study investigates how to reduce power consumption in training transformer language models. We seek to address the issue of high-power models by analysing the resulting models’ hyperparameters, energy consumption, and perplexity and providing initial parameter guidelines for low-power, high-performance transformers, and an opening into the research of low-power transformers.

Our research question is: How can we reduce the energy consumption of models to both lower the barrier of entry and reduce CO_2 emissions, while still keeping an effective model?

Following Strubell et al. (2019), a possible approach to this problem could be the use of *Bayesian hyperparameter search*. Throughout our work, we managed to identify hyperparameter configurations that provide strong entries for both perplexity and energy consumption. These configurations were found through our methodology utilising Bayesian optimisation, by combining the libraries Hyperopt and PyTorch. The optimal configurations collectively spanned the identified Pareto frontier.

2 Related Work

There is a body of work on making language models and transformers efficient, for varying definitions of efficiency, but we’re not aware of any that algo-

: These authors contributed to the paper equally.

rhythmically integrate power consumption into the loss function or architecture search.

The closest related work is that on task-specific network reduction and on low-power language processing. The former can be achieved through distillation, pruning, quantisation, or all of the above. For example, [Wasserblat et al. \(2020\)](#) reduce trained BERT models in size by orders of magnitude while retaining task performance. Similarly, [Kim et al. \(2019\)](#) present highly efficient networks that as a result can process translations very quickly. However, all these techniques require the training of a large network first, thus only offering power savings at inference time. Furthermore, [Kaplan et al. \(2020\)](#) presents scaling laws for neural language models, which can assist in more efficient training when applied.

3 Method

Our general approach is as follows. We specify a dataset, task, objective function, and hyperparameter space. We then explore hyperparameter space, repeatedly training models over the same data and evaluating them in terms of task performance and power consumption. This exploration optimises for good task performance and low power consumption, but is limited to a certain volume of model configurations. Once complete, we analyse these model configurations further, investigating per-epoch performance, and common factors in high and low power consumption and task performance.

3.1 Data

The dataset chosen is the CC-News dataset ([Mackenzie et al., 2020](#)). This is a subset of the English-language portion of the entire CC-News dataset. This specific set of data was chosen due to it being partially what RoBERTa was trained on, and having the longer document length typical to the news genre. Only the first 100 000 examples are used as train, primarily chosen in an attempt to keep the energy consumption of the trained models to a minimum. Each document comes with multiple data fields, with only the `text` field being used for training. The reduced amount of data introduces an assumption that these results will scale, which we address later in the paper when investigating common factors in efficient and inefficient models.

3.2 Task: Language modelling

The task used for this paper is *Masked Language Modelling*, also referred to as Masked LM or MLM. The procedure is very simple: mask some words in the input sentences with the token (`[MASK]`), and then attempt to predict what these words are. An example of such a sentence is “The borders of Paris are `[MASK]`”, where `[MASK]` is the word to be predicted. Perplexity is a widely used metric for the evaluation of language models. Low perplexity means better performance, which makes it a useful metric to evaluate language models in general. We use perplexity as defined by [huggingface](#), as the exponentiated average negative log-likelihood of a sequence ([HuggingFace, 2021](#)).

3.3 Perplexity-Energy Product

We used a simple multiplication of the total perplexity and energy consumption of a model, $m_{perplexity}, m_{energy}$ to act as the return value, Perplexity-Energy Product (PEP), to be reduced:

$$PEP = m_{perplexity} \cdot m_{energy} \quad (1)$$

Where $m_{perplexity}$ is the perplexity of the trained LM, and m_{energy} is the energy consumption for training the LM measured by Carbontracker ([Anthony et al., 2020](#)). We chose to call this return value, PEP, as shown in the equation. The reason for using a composite expression of both perplexity and energy is to make the optimization focus on parameters that affect both of these. We chose to multiply the two values as we hypothesised it would punish high values a lot more since multiplication is a lot more violent than addition. An issue with this is that as soon as either value is below 1, the loss is simply a linearly scaled-down expression of the other number. A worst-case scenario would be that Hyperopt would prioritise optimising one value and neglect the other, but the resulting loss would still be acceptable for the optimiser. A lower PEP value is better, since the aim is to minimize the energy cost and the perplexity.

There are underlying issues with assessing both the quality and efficiency of models in a single metric. One of the issues is that when efficiency is below one kilowatt-hour, the resulting PEP value will essentially be a scaled-down perplexity. So potentially, a model with a good balance between efficiency and quality would have a higher PEP value than a model that is very efficient, but lacking in quality. Additionally, using a single metric of the

Parameter	Interval
vocab_size	[1,30522]
hidden_size_multiplier	[1,100]
num_hidden_layers	[1,12]
num_attention_heads	[1,18]
intermediate_size	[1,3072]
hidden_act	(relu, silu, gelu, gelu_new)
hidden_dropout_prob	[0.1,1]
attention_probs_dropout_prob	[0.1,1]
max_position_embeddings	[512,512]
type_vocab_size	[1,1]
initializer_range	[0.02,0.02]
layer_norm_eps	[1.00E-12,1.00E-12]
position_embedding_type	(absolute, relative_key, relative_key_query)

Table 1: Hyperopt parameter search space

product of cost and quality makes an assumption - that changes in either component have a linear impact. This could be regulated by adding a weight to both the metrics in order to regulate the difference of change within the respective metric.

3.4 Hardware

The experiment platform was V100 GPUs on i7 CPUs running over a SLURM service. As stated by (Lasse, 2021), Carbontracker also works slurm, only measuring GPU devices available for the given job, so the reported consumption figures are consistent but could be slight underestimates.

3.5 Hyperparameters

To investigate hyperparameter impact on Transformer model power use, we chose to specifically look at the parameters related to its size, such as the number of its hidden layers or number of attention heads, alongside a few key parameters such as the type of positional encoding, activation functions and dropout probabilities. The parameters picked were chosen on the assumption that they were the ones most likely to affect both model perplexity, and model energy consumption during training.

Table 1 shows our final search space and parameter value intervals. Please note that the *hidden_size* of the model is given as:

$$h_{size} = h_{size_mult} \cdot \#A_{heads} \quad (2)$$

Where h_{size} is hidden size, h_{size_mult} is hidden size multiplier and $\#A_{heads}$ is number of attention heads. This is due to the constraint that the hidden size of the model has to be a multiple of the number attention heads in the model (Vaswani et al., 2017). This leaves a parameter space of size 1026 between

[1, 1800], most of which are centred around the lower end of the interval.

3.6 Search algorithm

We use Bayesian optimisation to traverse hyperparameter space in search of low-power transformer training configurations. Bayesian optimisation is suited to cases where one wishes to find optimal parameter configurations, for some definition of optimal, but individual trials can be expensive. This makes it a good match for transformer hyperparameter tuning. Search is executed in parallel over multiple GPUs and GPU hosts. As noted in (Bergstra et al., 2011), “*The consequence of parallelization is that each proposal x^* is based on less feedback. This makes search less efficient, though faster in terms of wall time*”. Research indicates Bayesian hyperparameter search techniques are more efficient than brute force techniques (Strubell et al., 2019), such as grid search; this family is often the least efficient in terms of time-to-viable-solutions. Further, our general focus on energy efficiency motivates choosing efficient search algorithms.

Samples follow a uniform distribution. This was a deliberate choice, as we have no knowledge over which parameters would perform best. We hypothesised giving a uniform distribution for all parameters would yield the best results. The specific algorithm used to minimise the loss is the *Tree of Parzen Estimators* (TPE) (Bergstra et al., 2013).

3.7 Model Selection and Per-epoch Measurements

Hyperopt was left to run over the data and hyperparameters for a fixed number of days. Models were trained with a batch size of 2, over 3 epochs. The batch size was chosen due to stability, as the relatively low number of epochs could make the results unstable. According to Masters and Luschi (2018), the most stable and best generalization results have been obtained with a batch size of 32 or smaller, but the best results were with batch sizes as low as 2. We also logged all of the parameter configurations alongside the energy consumption and perplexity of each model. The result was 154 different hyperparameter configurations.

We then retrained the 154 hyperparameter configurations we found over 10 epochs to see a further evaluation of how each model would evolve per epoch in terms of power consumption and performance. For each model, a callback implementing Carbontracker was used to gather data about the

energy consumption after each epoch, and the same callback was used to log the perplexity of each model. Another callback was created to save a model for each epoch, resulting in 1540 different models being saved. Each model was trained on a single Tesla V100 32GB graphics card.

4 Results

4.1 Correlation Matrices

The majority of the result section is going to utilise correlation matrices to analyse the data. We have two categorical data entries, namely the activation function and the position embedding type. For this, we use categorical correlation to showcase both the activation function and the position embedding type alongside the rest of the data. The primary reason for using correlation matrices is its ability to quickly visualise and pinpoint patterns in the data. We have three different ways of evaluating the different hyperparameters, namely with (i) energy consumption, (ii) perplexity, and (iii) PEP value, and therefore correlation matrices make it easy to visualise the hyperparameters. In general, the primary concern is to see correlations between the evaluation methods and hyperparameters.

4.2 PEP

Table 2 contain overviews of the best 15% of models and worst 15% of models, for PEP value, with regard to average parameter size. A count of what position embedding type and activation function was used can be found in table 3 The models were sorted by lowest PEP value, and the best 15% and worst 15% were chosen. Table 2 presents an overview of average energy consumption and perplexity of the best 15% of models, to compare with the worst 15% of models in terms of PEP value.

The data of the best 15% of models is introduced to analyse tendencies that provide the PEP values. The data of the worst 15% of models is introduced to analyse what *not* to do when choosing parameters for a new model. Three different correlation matrices, for all models, for the best and worst 15% of models are given in Appendix A.

5 Energy

The data here is presented in the same format as the previous section. The table 4 show findings of the best 15% models in terms of low energy consumption, and the worst 15%, alongside several correlation matrices in appendix B. Furthermore,

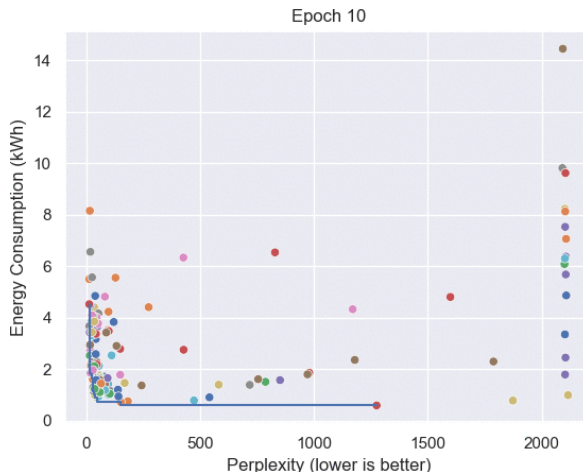


Figure 1: The Pareto curve for epoch 10. An animated visualisation of all 10 epochs can be found at [this link](#).

As these results are extracted from models that have been trained through hyperopt with a specific loss function, the resulting parameters are not chosen to achieve the lowest energy consumption possible, but rather the lowest *PEP*-value. These results can then possibly indicate which parameters can be tweaked to reduce energy consumption specifically, while retaining some performance.

5.1 Identifying optimal models

While PEP is a suitable metric, we also want to be able to identify models which can't be optimised further with respect to perplexity or energy consumption without a penalty in the other. Therefore we identified the Pareto-optimal models.

The next graphs, Figure 1 showcases energy consumption related to perplexity for each model evolving over the 10 epochs. The colour of each dot represents a specific model throughout all the models, hence it is possible to see how each model progresses throughout the graphs. Besides a visualisation of how each model evolves, the graph also highlights the Pareto curve for each epoch, which is an indicator of the best-performing models for each epoch. While some of these best models might not have a particularly good PEP value, they are still a part of the Pareto curve, and thus can't improve either perplexity or energy consumption without a decrease in the other.

Furthermore, as can be seen in figure 1, there are a few models which permanently reside at 2000 perplexity. These are all models which have low vocab size, high dropout probabilities, low hidden size, or a combination of the three. This, combined with a relatively high number of hidden layers and

	Best 15% Mean	Best 15% Std. Dev	Worst 15% Mean	Worst 15% Std. Dev
vocab_size	19458.52	5741.79	16072.78	9451.63
actual_hidden_size	273.17	101.14	670.52	468.80
num_hidden_layers	1.91	0.92	6.52	3.24
num_attention_heads	8.17	4.13	10.47	4.70
intermediate_size	716.17	531.30	1237.08	581.32
hidden_dropout_prob	0.18	0.06	0.52	0.24
attention_probs_dropout_prob	0.25	0.10	0.46	0.26
energy consumption	1.70	0.51	5.70	3.11
perplexity	27.23	7.80	1809.32	501.77

Table 2: Mean and standard deviation of hyperparameters for the best 15% and worst 15% of models wrt: PEP value.

	Best 15% Count	Worst 15% Count
relative_key_query	20	7
relative_key	0	8
absolute	3	8
GELU	14	8
GELU_new	2	7
ReLU	1	5
SiLU	6	3

Table 3: Count of activation functions and position embedding types wrt: PEP value.

attention heads result in models which have a hard time learning.

6 Analysis

For the analysis, we start with the most trivial optimisation steps and progress towards less trivial ones.

6.1 Parameter correlations

When looking at the best 15% models in terms of low PEP value, analysing the resulting correlation matrix given in Figure 3 in appendix A can give us insight into whether our approach has resulted in a good balance between energy consumption and perplexity. It can also give clues as to what parameter values can be chosen to reduce a models energy consumption without affecting perplexity. Additionally, we define a good balance for a hyperparameter between energy consumption and perplexity as the absolute value of the correlation one parameter to perplexity roughly equalling the absolute value of the correlation between the same parameter and energy consumption. Specifically:

$$|corr_{param_n, PPL}| \approx |corr_{param_n, energy}|$$

The correlation between perplexity and energy consumption is at -0.88, which indicates that the

use of hyperopt and our PEP loss function has given a good balance between the quality and energy use during optimisation. The three key parameters appear to be the number of hidden layers, the hidden activation function and the position embedding type. Hidden layer count has a correlation of 0.71 and -0.64 between energy consumption and perplexity respectively, this being the hyperparameter with the greatest impact on both of these values. Activation function also has a high impact, again with -0.63 and 0.68 with energy consumption and perplexity respectively (Derczynski, 2020). Lastly, the positional embedding is at 0.49 and -0.65 respectively. As in Table 2 and 3, the optimal number of hidden layers averaged out at 1.91. The most used activation functions and positional encodings are GELU and relative_key_query and 14 and 20 appearances. It is important to note that as the positional encoding defines the way attention is calculated, there may be an underlying link between the number of attention heads and encoding choice.

It is important to note that correlation is a useful tool to find linear tendencies. While a correlation close to negative and positive indicates a correlation between the two values, a value of zero doesn't guarantee a lack of correlation since there could still exist a non-linear correlation.

6.2 What predicts a good or bad model?

Looking at Table 2, the overall tendency is that the models, on average, are much smaller than the config of *RoBERTa_{BASE}* (Liu et al., 2019). The number of hidden layers in our best models in terms of PEP value is on average 2, down from 12. With hidden layers having a correlation of 0.71 with energy consumption and -0.64 with perplexity, as seen in Figure 3, it is by far one of the most volatile parameters to adjust. Increasing the number of hid-

	Best 15% Mean	Best 15% Std. Dev	Worst 15% Mean	Worst 15% Std. Dev
vocab_size	21187.73	6426.71	18545.04	9141.88
actual_hidden_size	116.43	84.20	727.78	27.33
num_hidden_layers	1.52	0.77	7.82	3.26
num_attention_heads	8.08	5.27	12.78	4.48
intermediate_size	890.47	554.19	1149.65	739.19
hidden_dropout_prob	0.37	0.23	0.40	0.15
attention_probs_dropout_prob	0.28	0.14	0.47	0.28
energy consumption	0.99	0.17	6.18	1.93
perplexity	338.51	576.74	1236.04	962.74

Table 4: Mean and standard deviation of hyperparameters for the best 15% and worst 15% of models wrt: energy consumption.

	Best 15% Count	Worst 15% Count
relative_key_query	15	14
relative_key	4	6
absolute	4	3
GELU	3	11
GELU_new	3	5
ReLU	6	3
SiLU	11	4

Table 5: Count of activation functions and position embedding types wrt: energy consumption.

den layers will increase energy consumption, but decrease perplexity. Interestingly, with it having a correlation of 0.056 to PEP value, it could suggest that hyperopt has found a good compromise in the number of hidden layers that gives a good balance between low energy consumption and perplexity.

The number of attention heads is at 8, which is higher than what our initial assumption was, but with a deviation of 4.13, it varies a lot from model to model. It is important to note that this parameter is dependent on the type of positional embedding used, as the way attention is calculated is heavily dependent on it.

While these models have good performance, with the energy consumption averaging at 1.70 and perplexity at 27.23, it is important to note that these two metrics have a negative correlation with each other of -0.88. If one is reduced, the other one increases. This could suggest that we have hit a point where we can no longer make our models smaller without severely affecting our end performance. RoBERTa reported perplexities between 3.68 and 3.99 in table 3 in (Liu et al., 2019). While our perplexities on average are roughly 7.4 times higher, both our amount of training data and model size are vastly smaller by a factor of 10, thus probably leading to a shorter downstream task fine-tuning time, and as

a result, lower energy consumption.

On the opposite end of the spectrum are the worst 15% of the models in terms of PEP value. We assume that these models would be bigger, both in terms of hidden layers, their hidden size, and intermediate size, as these would most likely result in longer training times than smaller ones, thus consuming more energy. Comparing Table 2 to the best 15% supports this analysis. The number of hidden layers has increased from an average of 1.91 to 6.52, hidden size from 273 to 670, and intermediate size from 716 to 1237. The energy consumption is indeed also higher, as it has gone from 1.7 kWh to 5.7 kWh. The perplexity is also very bad, being at an average of 1809.

6.3 Reducing LLM training power consumption without reducing quality

There is also a slight correlation between hidden dropout probability and perplexity, but there is almost no correlation between that probability and energy consumption. These correlations suggest that a low hidden dropout probability results in a lower perplexity, without impacting the energy consumption of the models.

The top 15% of models at $\eta = 10$ (Figure 3) indicate interesting correlations. This matrix suggests a stronger negative correlation between energy consumption and perplexity. These results indicate that it is hard to reduce both perplexity and energy consumption at the same time, since lowering one tends to increase the other - a trade-off. This could be because the models are already performing decently well and have reached the point where subsequent iterations bring diminishing returns in perplexity, while still incurring a linear increase in energy consumption. Looking at the models, 10 of the 23 models in the best 15% appear on the Pareto-

optimal curve as seen in Figure 1. This suggests that the remaining 13 models are very close to being optimal, whereas the 10 models which appear on the Pareto curve already are. The same trend happens in the number of hidden layers, where the correlations indicate that increasing the number of hidden layers will also increase energy consumption, but will decrease perplexity. Interestingly, the correlation previously seen with the hidden dropout probability has disappeared. This indicates that the best-performing models already have a low enough hidden dropout probability for this correlation to no longer be the case. When looking at the data, the entire dataset features an average hidden dropout probability of 0.385, where the best 15% of models have an average of 0.189, which supports the previous indication.

6.4 Different parameters have different effects in different realms

Most of our well-performing models – no matter if looking only at energy, perplexity, or PEP value – use the relative key query positional embedding. While this is a slightly more complex calculation for self-attention, this suggests that it doesn’t have a huge impact on energy consumption, while it has a noticeable impact on perplexity. Relative key query introduces another 143K parameters but is relatively insignificant when compared to the 103M BERT already has (Huang et al., 2020). While our dataset contains 107 models with “relative key query” as the positional embedding type, when looking at the worst-performing models, the distribution looks close to uniform, at least when looking at perplexity and PEP. Furthermore, Table 13 shows the distribution in the first cluster, which features 94 out of the 107 models with relative key query, further strengthening the claim.

A similar trend appears for the choice of activation functions, with GELU being the best-performing activation function for our best models in terms of perplexity and energy loss. If one wants an activation function purely for energy efficiency, SiLU is the most prevalent activation function among the models with the lowest power consumption. When looking at Table 13, it’s seen that GELU and SiLU appear the most frequent, but ReLU is still close behind, in line with previous results (Derczynski, 2020).

Our findings were focused on hyperparameter optimization, and no compression of the LMs at all. Jacobsen et al. (2021) provide methods for measur-

	Energy Consumption
Hyperopt (1 Epoch)	50.53 kWh
Hyperopt (3 Epochs)	187.57 kWh
10 Epochs	393.56 kWh
Ad-hoc tests	62.42 kWh
Total	694.10 kWh

Table 6: Energy consumption for this work

ing model size. Li et al. (2020) show that training larger models for longer times and then compressing them leads to more efficient training. Though the previous analysis on layers and attention heads is less relevant for this approach, our findings regarding loss functions and embedding types should hold if choosing to train a larger model and then compress.

6.5 Our energy consumption

This paper investigates alternatives to high-performance, high-energy consumption top-of-the-line models by showing that other approaches can provide acceptable performance while cutting the size and training time. As this topic is relevant due to the environmental impact it can have, such as accelerating rising sea levels (Veng and Andersen, 2020), and as training NLP models has an actual impact on the environment due to high energy usage, reducing their consumption is, as a topic, both important and very unexplored. This also means that we, as the authors, have to stress the importance of noting that this is not a complete guide on how to create low-power, low-perplexity transformer models. As an example, reducing the number of hidden layers might have a positive effect in certain models, but not necessarily in others - one has to look at all the model parameters together.

Table 6 shows the total amount of energy we’ve spent training our various models. As the price of 1 kWh in Denmark is roughly €0.28 (Eurostat, 2021), the total amount of money we spent training all of our models totals roughly €195 in electricity. While not a lot in comparison to the models presented in table 3 in Strubell et al. (2019), our models are also trained on a smaller dataset, and some of them have a configuration that slims down their size. In terms of watt-hour usage, the number of models we trained consumed half the amount of energy that the BERT base did. As a comparison, 700 kWh can power a Tesla Model 3 for a whole 4337 kilometres (standard range of 16kWh per 100 kilometres (Wikipedia, 2021)), which is roughly

from Copenhagen to Barcelona and back.

6.6 Limitations

In this subsection, we discuss possible limitations of our study and indicate where future work could explore further.

Given the fact that this work was done on a small subset of the CC-News dataset, the best performing models may be smaller models where the small amount of data is not a downside. If the size of the corpus is increased, it would give space for the larger models to learn more, and maybe perform better than the smaller models deemed as the most efficient. Even though other efficient models might be uncovered, this will not change the correlation between the number of hidden layers and efficiency. There is also a small possibility that perplexity scores are inflated, as CC-News was used to pre-train RoBERTa (Liu et al., 2019), and this could occur non-linearly across hyperparameter configurations – but we expect the relations between network architecture components, power consumption, and performance to hold relative to each other overall.

We also note that this work was monolingual and focused on English, to the exclusion of other languages (Bender, 2019). While transformer-based LLMs seem to exhibit some properties in common across natural languages, there are also many language-specific considerations (e.g. for Finnish, Virtanen et al. (2019)). Our results cannot be guaranteed to generalise across other natural languages.

Given the fact that a heterogeneous cluster was utilized for training all of the models, there is no guarantee that the precise hardware configuration was used to train all 154 models. All compute nodes have the same type of graphics card but contain different server-grade CPUs. Since CPU and DRAM contribute to power consumption (Anthony et al., 2020), the variations in clock speed and core count could potentially have a small effect on the final energy consumption.

We used a relatively low epoch count for the search of hyperparameter space. This might not reflect high-epoch hyperparameter optima, and so the relations between perplexity scores are at best advisory. However, power consumption tends to be stable per epoch, and so this component of the function performs well.

We evaluate using perplexity, which has its own issues and is unlikely to approximate many NLU metrics. However, it does lend itself well to our

exploratory compound loss function, retains agnosticism regarding the many possible NLU tasks (such as those in SuperGLUE). The correlations between evaluated language model performance and their final performance are not of maximum certainty; however, these results can guide hyperparameter search/tuning in terms of good balances, and we hope to see much more research in this area of transformer efficiency and quality/energy use trade-off.

We fixed the epoch count for studies. An alternative would be to fix the FLOPS available. Gordon et al. (2018) present a regulariser that optimises FLOPS usage. Because perplexity tends to be asymptotic with zero for useful models, making absolute differences between model scores tend to be smaller as time goes on. This makes the metric a little noisy and differences hard to note. Fixed FLOPS allowances will lead to high epoch counts for smaller models, increasing the risk of model performances being hard to tease apart. Thus, fixing epochs instead of FLOPS during exploration gives finer granularity for very small, or very efficient models, though FLOPS is a closer approximation to power consumption.

As NVIDIA (2021) highlights, some size multipliers are more efficient when it comes to matrix-matrix multiplication. Allowing the search space as denoted on table 1 to take non-efficient values can affect reproducibility because of hardware idiosyncrasies. While general trends can be extracted, precise figures might not generalise well beyond the test setup.

7 Conclusion

We investigated how hyperparameters affected the power consumption and model quality of transformer-based language models. This paper has presented a method for low-power investigation of hyperparameter tuning, integrating power consumption measurement. We identified factors that increase power without giving quality and identified factors that increase quality without taking power. There are many possible extensions to this work and it is our hope that power consumption measurement will be improved and will be integrated with more architecture searches during model training.

References

- Dario Amodei. 2018. AI and Compute. <https://openai.com/blog/ai-and-compute/>.
- Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. **Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models**. *arXiv:2007.03051 [cs, eess, stat]*. ArXiv: 2007.03051.
- Emily M Bender. 2019. The #benderrule: On naming the languages we study and why it matters. *The Gradient*, 14.
- Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. **On the dangers of stochastic parrots: Can language models be too big?** . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21*, page 610–623, New York, NY, USA. Association for Computing Machinery.
- James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS'11*, page 2546–2554, Red Hook, NY, USA. Curran Associates Inc.
- James Bergstra, Daniel Yamins, and David Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR.
- Leon Derczynski. 2020. Power consumption variation over activation functions. *arXiv preprint arXiv:2006.07237*.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. *arXiv:1810.04805 [cs]*. ArXiv: 1810.04805.
- Eurostat. 2021. **Electricity price statistics**.
- Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. 2018. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1586–1595.
- Zhiheng Huang, Davis Liang, Peng Xu, and Bing Xiang. 2020. **Improve Transformer Models with Better Relative Position Embeddings**. *arXiv:2009.13658 [cs]*. ArXiv: 2009.13658.
- HuggingFace. 2021. **Perplexity of fixed-length models**.
- Magnus Jacobsen, Mikkel H Sørensen, and Leon Derczynski. 2021. Optimal size-performance trade-offs: Weighing PoS tagger models. *arXiv preprint arXiv:2104.07951*.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. **Scaling Laws for Neural Language Models**. *arXiv:2001.08361 [cs, stat]*. ArXiv: 2001.08361.
- Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. 2019. From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 280–288.
- Lasse. 2021. **lfwa/carbontracker**. Original-date: 2020-04-21T12:01:38Z.
- Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, and Joseph E. Gonzalez. 2020. **Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers**. *arXiv:2002.11794 [cs]*. ArXiv: 2002.11794.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. **RoBERTa: A Robustly Optimized BERT Pretraining Approach**. *arXiv:1907.11692 [cs]*. ArXiv: 1907.11692.
- J. Sander M. Ester, H. P. Kriegel and X. Xu. 1996. **A density-based algorithm for discovering clusters in large spatial databases with noise**.
- Joel Mackenzie, Rodger Benham, Matthias Petri, Johanne R Trippas, J Shane Culpepper, and Alistair Moffat. 2020. CC-News-En: A large English news corpus. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3077–3084.
- Dominic Masters and Carlo Luschi. 2018. **Revisiting Small Batch Training for Deep Neural Networks**. *arXiv:1804.07612 [cs, stat]*. ArXiv: 1804.07612.
- NVIDIA. 2021. **DL performance matrix multiplication**.
- Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. **Green AI**. *arXiv:1907.10597 [cs, stat]*. ArXiv: 1907.10597.
- Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. **Energy and Policy Considerations for Deep Learning in NLP**. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. **Attention Is All You Need**. *arXiv:1706.03762 [cs]*. ArXiv: 1706.03762.

Tadea Veng and Ole Andersen. 2020. [Consolidating sea level acceleration estimates from satellite altimetry](#). *Advances in Space Research*.

Antti Virtanen, Jenna Kanerva, Rami Ilo, Jouni Luoma, Juhani Luotolahti, Tapio Salakoski, Filip Ginter, and Sampo Pyysalo. 2019. Multilingual is not enough: Bert for finnish. *arXiv preprint arXiv:1912.07076*.

Moshe Wasserblat, Oren Pereg, and Peter Izsak. 2020. Exploring the boundaries of low-resource bert distillation. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 35–40.

Wikipedia. 2021. [Tesla Model 3](#). Page Version ID: 1024502085.

A PEP

Figure 2, 3, and 4 showcase the correlation matrices of the 154 trained models, the best 15% of models with regard to PEP value, and the worst 15% of models with regard to PEP value respectively. These correlation matrices are used to find relationships between different hyperparameters and PEP, in order to figure out which hyperparameters have a huge effect on PEP, and which do not have an effect.

B Energy

Figure 5, and 6 showcase the correlation matrices for the best 15% of models with regard to energy consumption, and the worst 15% of models with regard to energy consumption respectively. Having a correlation matrix over all 154 models would produce an identical result to Figure 2, so that's not included here. These correlation matrices are primarily used to find hyperparameters which greatly affect energy consumption, or which hyperparameters barely affect energy consumption. Both of these can be used to find hyperparameters which need to be tuned, and to find hyperparameters which can be tuned with regard to perplexity, given they do not affect energy consumption.

C Early optimisation results

In this section, we will present the preliminary result after the hyperparameter tuning done with hyperopt. This includes some of the first observable tendencies and was used to start the analytical process.

The loss saved from each model produced by hyperopt is the average overall three epoch, instead of the actual loss at the end of the third epoch. This results in difficulty calculating perplexity, as the loss is now skewed heavily towards the mean, and in return also results in energy loss not being comparable to other parts of the project. Energy consumption has been logged after the end of the third epoch, and can therefore still be compared to the rest of the project. Furthermore, it's not possible to try and calculate what the actual perplexity is, given that there is no real relation between perplexity and the number of epochs. The data will still be presented, but given the fact that the data collection has been different from the rest of the project, it will mostly not be compared or analysed further.

After hyperopt training all 154 models on 3 epochs was done, an analysis of the different model

configurations was performed in order to spot early trends in the data. A correlation matrix was constructed for all variable parameters, and compared to energy consumption, perplexity, and energy loss, as can be seen in Figure 7. When speaking about energy consumption, hyperparameters such as the number of hidden layers and the size of said hidden layers plays a big role. As either the number of hidden layers or their size increases, so does the energy consumption needed to train the model. Furthermore, when looking at perplexity, there are no correlations that are as strong as there were for energy consumption. The biggest thing to note here is that there is a positive correlation between hidden layer dropout probability and perplexity. The same goes for hidden layers and hidden size, although this correlation is not as strong as with energy consumption.

The last interesting thing to note here is that there is a positive correlation between energy consumption and perplexity. If one looks at the correlation between energy consumption and energy loss, or perplexity and energy loss, you see a closer correlation between those two parameters, which indicates that the ratio between energy consumption and perplexity, albeit a positive one, is a weaker correlation than the others. There are multiple explanations for this. It could be that this correlation shows that models can be improved in either department without sacrificing the other metric when looking at all models as a whole. It could also be that the aforementioned outliers are having an effect on the data. Following this, the correlation matrix for the best 15% of our models was constructed, as can be seen in Figure 8. When filtering for the best performing models, the strongest correlation is between energy consumption and perplexity, with a correlation of -0.86 . This indicates that as energy consumption goes higher perplexity will go lower, and the other way around.

D Perplexity

As with the previous section where we went through the models sorted by the best and worst 15% in terms of energy consumption, this section will do the same but in terms of perplexity instead. By analysing the parameters, we might be able to find a relation between certain of these that can cause both low energy consumption and perplexity, thus reducing the model size without affecting performance.

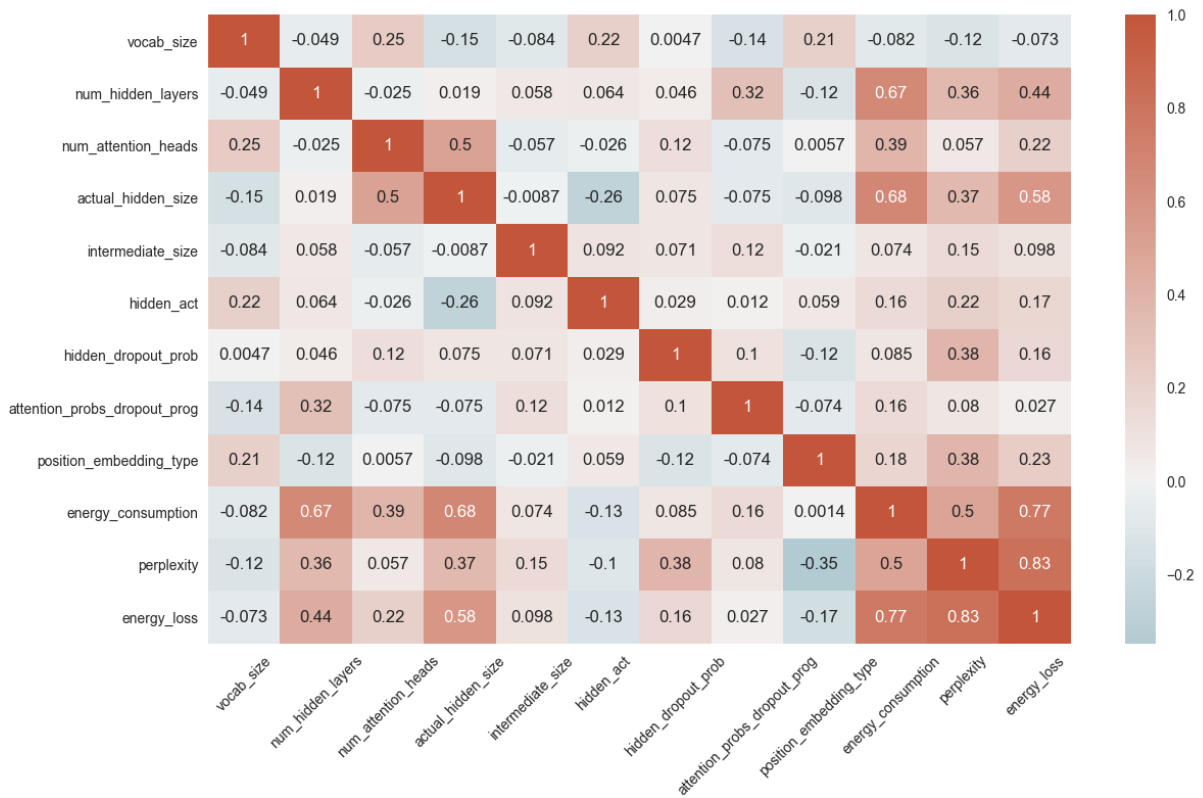


Figure 2: Correlation matrix over hyperparameters for 10 epochs.

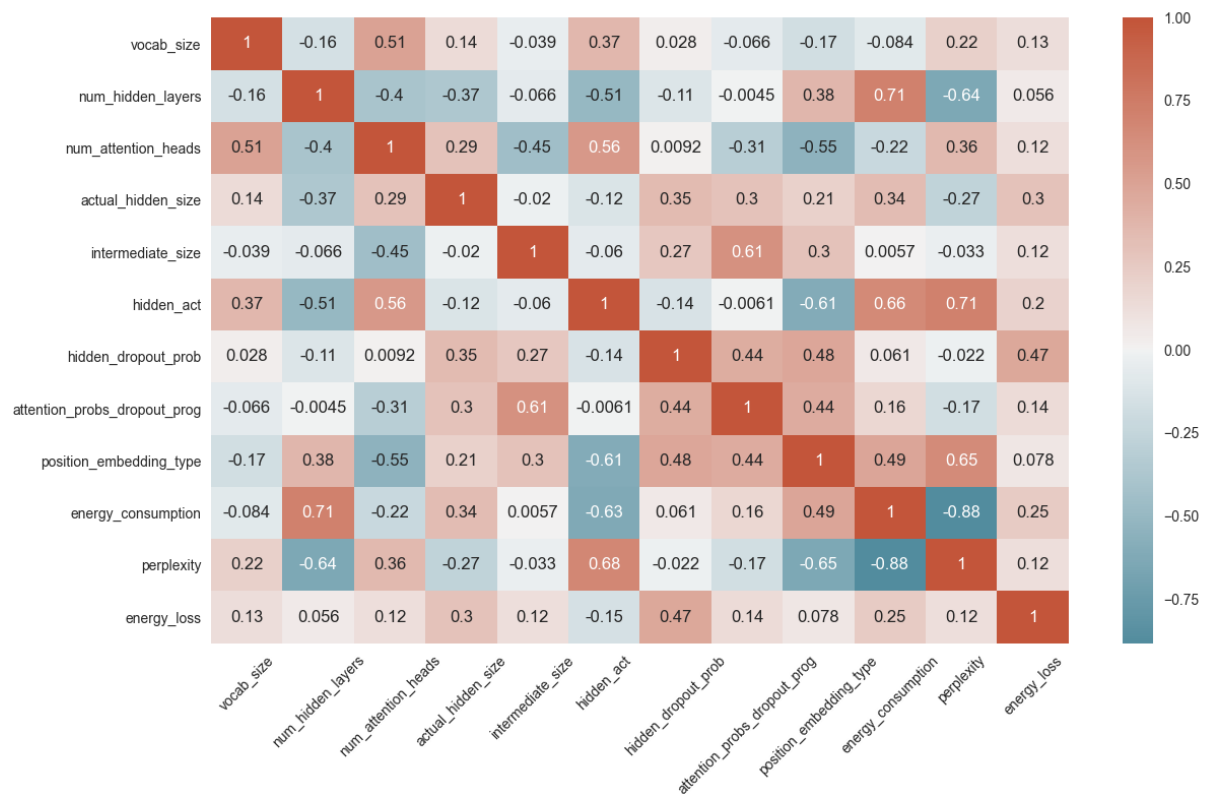


Figure 3: Correlation matrix over hyperparameters for the best 15% of models for 10 epochs.

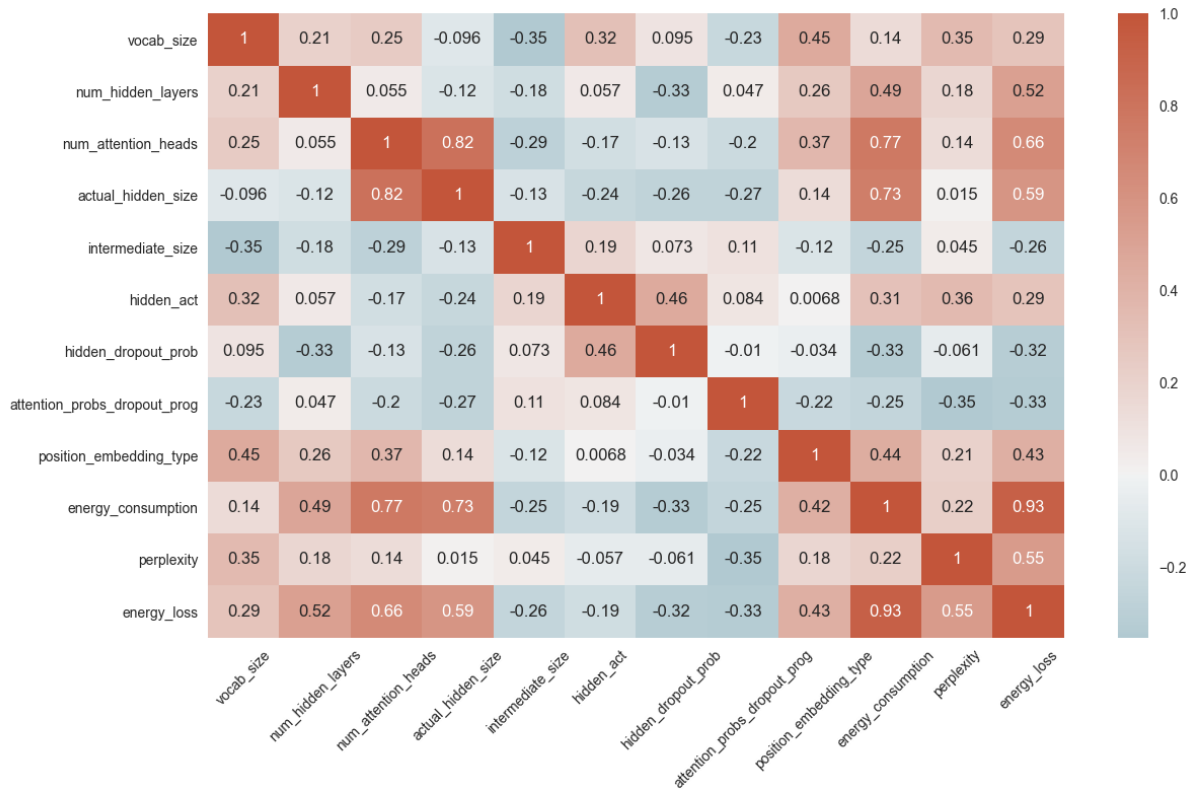


Figure 4: Correlation matrix over hyperparameters for the worst 15% of models for 10 epochs.

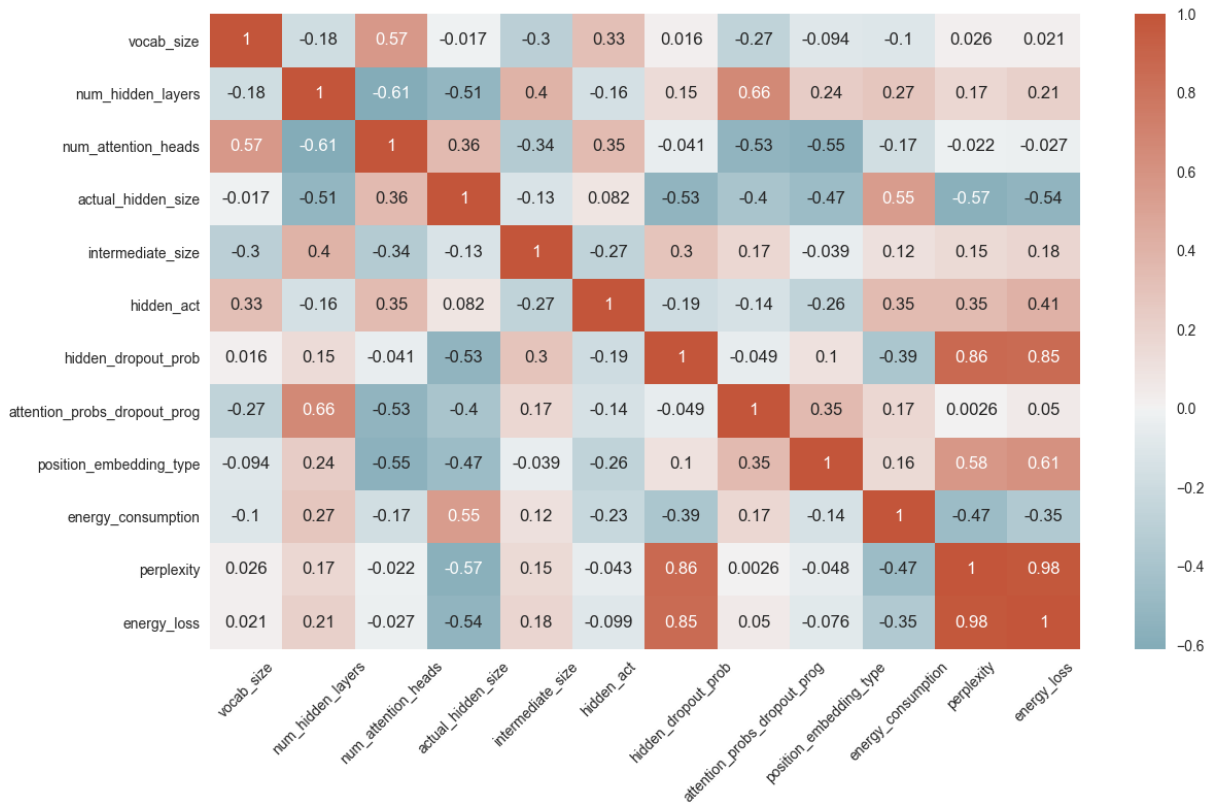


Figure 5: Correlation matrix of the best 15% of models wrt. power consumption.

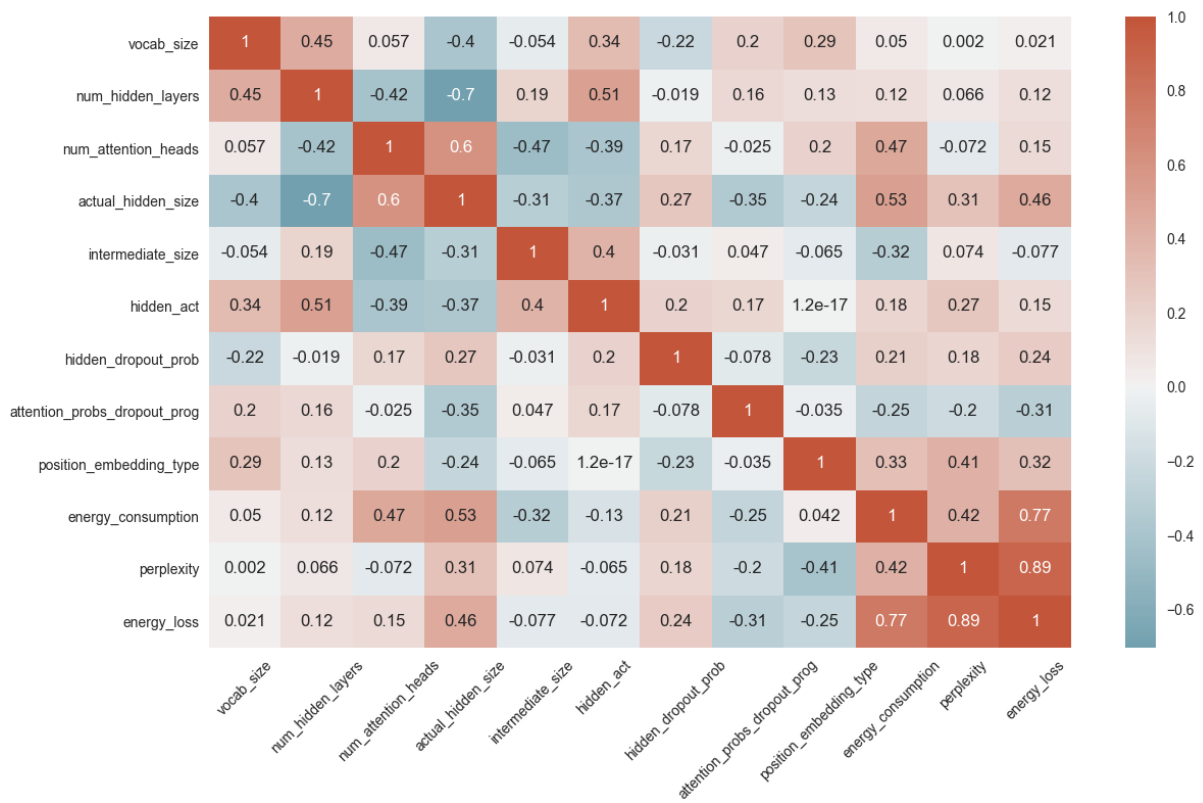


Figure 6: Correlation matrix of the most power-hungry 15% of models.

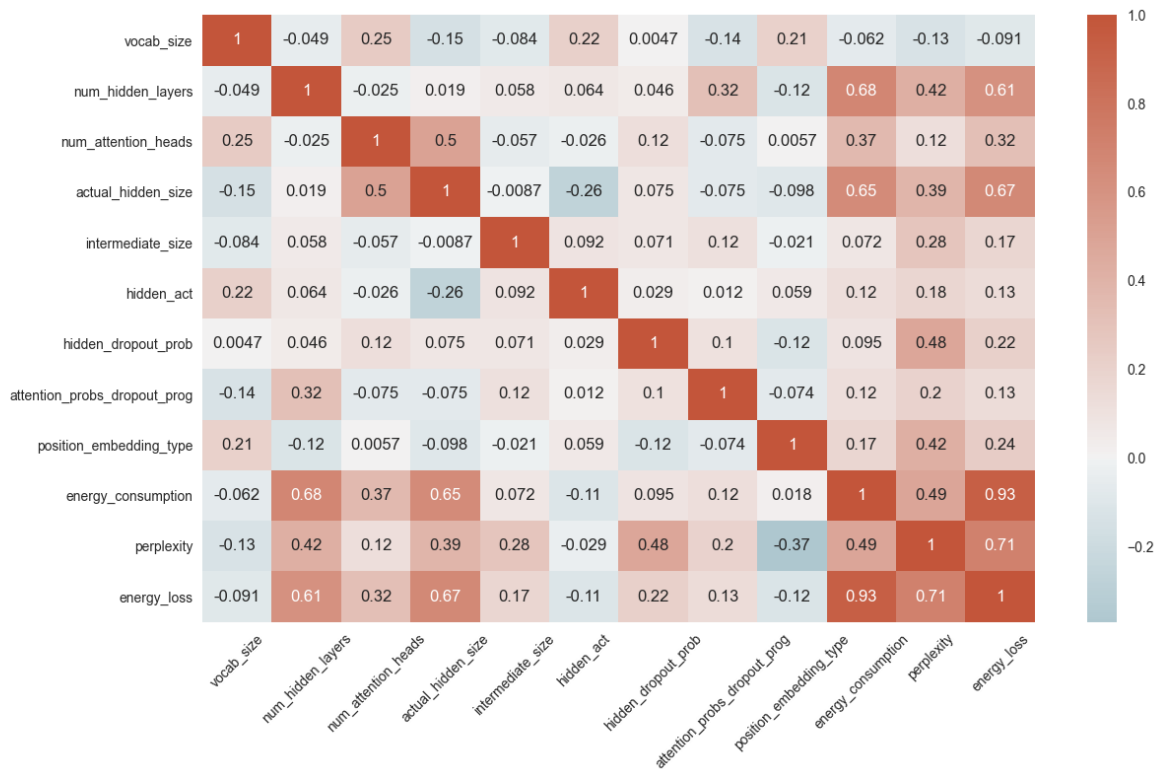


Figure 7: Correlation matrix over hyperparameters for 3 epochs of training.

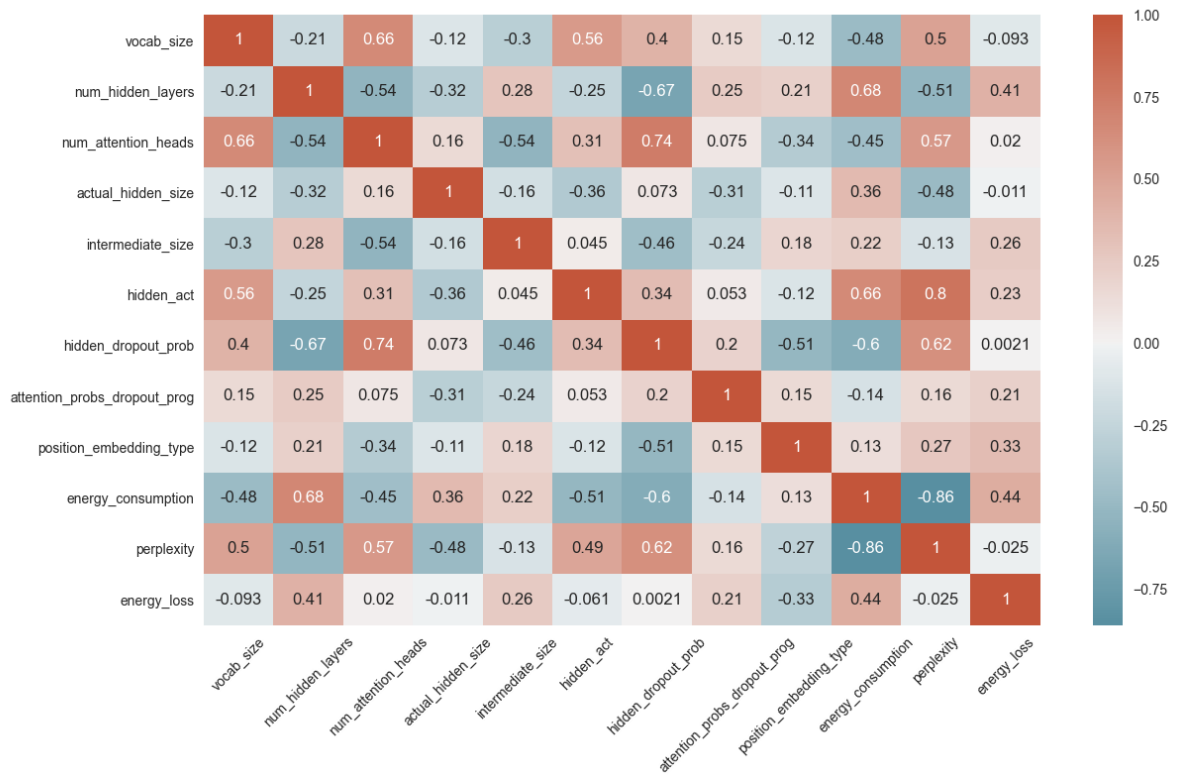


Figure 8: Correlation matrix over hyperparameters for the best 15% of models for 3 epochs of training.

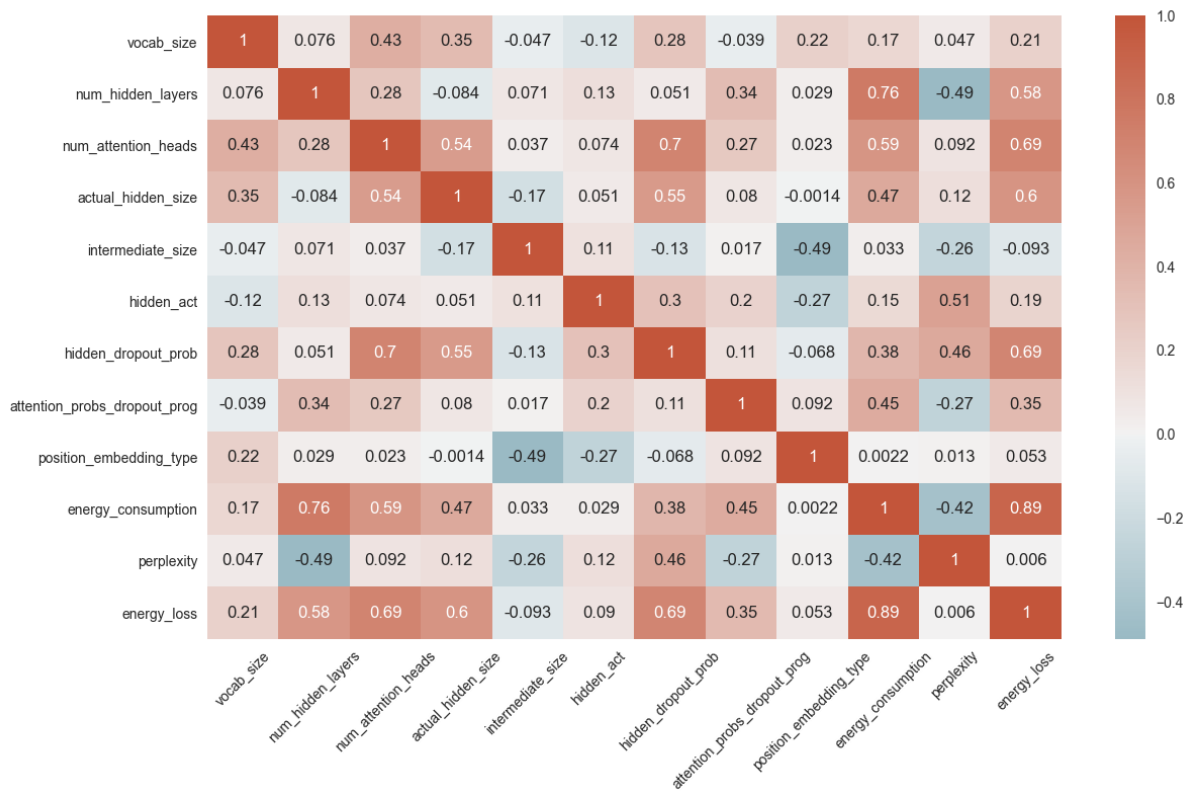


Figure 9: A correlation matrix of the best 15% of models with regard to perplexity.

	Average	Std. Deviation
vocab_size	18002.17	7986.21
actual_hidden_size	404.91	231.93
num_hidden_layers	3.69	2.09
num_attention_heads	10.34	4.62
intermediate_size	851.47	531.30
hidden_dropout_prob	0.21	0.08
attention_dropout_prob	0.37	0.24
energy consumption	3.40	1.58
perplexity	18.79	3.82

Table 7: Table with average hyperparameters of the best 15% of models wrt: perplexity.

The general point with the best-performing models in terms of perplexity is that they are slightly larger than both the best performing energy loss, and energy only models, as can be seen on Table 7. More hidden layers, more attention heads, bigger feed forward neural networks. A lot of these parameters do have a high standard deviation compared to their averages, such as the intermediate size and the number of hidden layers, meaning that there is room for reduction to reduce energy consumption through lower training times. What is interesting to point out is that while the models are slightly bigger, both their average perplexity and energy consumption are vastly better than the worst models in regard to energy consumption. It is possible that our best-performing perplexity models can, due to a slimmer size, reach good performance more easily than compared with the bigger models of the worst energy consumption. This energy consumption is still on average 2.4 kWh higher than our best performing models in terms of energy consumption, being at 1 kWh, but with a vastly better perplexity. It could suggest that energy loss, as a value to minimise for hyperopt, has been effective in finding compromises between performance and energy consumption. As can be seen on the corresponding correlation matrix for the best perplexity models, Figure 9, there is a negative correlation between energy consumption and perplexity of -0.42. This means that reducing one increases the other, which also suggests that hyperopt has found a compromise between the two. This matrix also shows a very high correlation between both energy consumption, perplexity, and the number of hidden layers. For energy consumption, it is 0.76, and for perplexity, it is -0.49. Increasing the hidden layers will drastically increase energy consumption, but it will also lower perplexity a lot. As mentioned earlier, since this adds extra training time to the model, it will automatically increase energy consumption.

This could possibly suggest that the number of hidden layers has a direct effect on how well a model performs.

	Appearances
relative_key_query	21
relative_key	2
absolute	0
GELU	11
GELU_new	5
ReLU	3
SiLU	4

Table 8: Count of activation functions and position embedding types in the best 15% of models with regard to perplexity.

Interestingly, as can be observed in Table 8, the distribution over positional embedding and activation function resembles the distribution of the ones sorted by energy loss a lot more than the ones sorted by energy consumption only, with some slight variations. They almost exclusively use relative key query positional embedding and rely more on a GELU activation function.

	Average	Std. Deviation
vocab_size	17920.60	8478.86
actual_hidden_size	569.26	441.47
num_hidden_layers	6.21	3.32
num_attention_heads	10	4.45
intermediate_size	1190.04	587.58
hidden_dropout_prob	0.54	0.26
attention_dropout_prob	0.43	0.26
energy consumption	5.20	3.40
perplexity	1891.61	364.25

Table 9: Table with average hyperparameters of the worst 15% of models wrt: perplexity.

The primary assumption with the models that perform terribly in terms of perplexity is that they are big, and thus have not had enough training time to fully develop. When comparing Figure 1 to Table 9, there are a couple of models that follow this trend, with one taking a significant leap down in perplexity between epoch 8 to 9. As that model follows the trend of badly performing models that are right on top of each other in terms of perplexity, it could be assumed that more epochs are what is needed for perplexity to drop. When looking at the general trend of the parameters, both the hidden size, number of hidden layers, attention heads and intermediate size are higher compared to the mod-

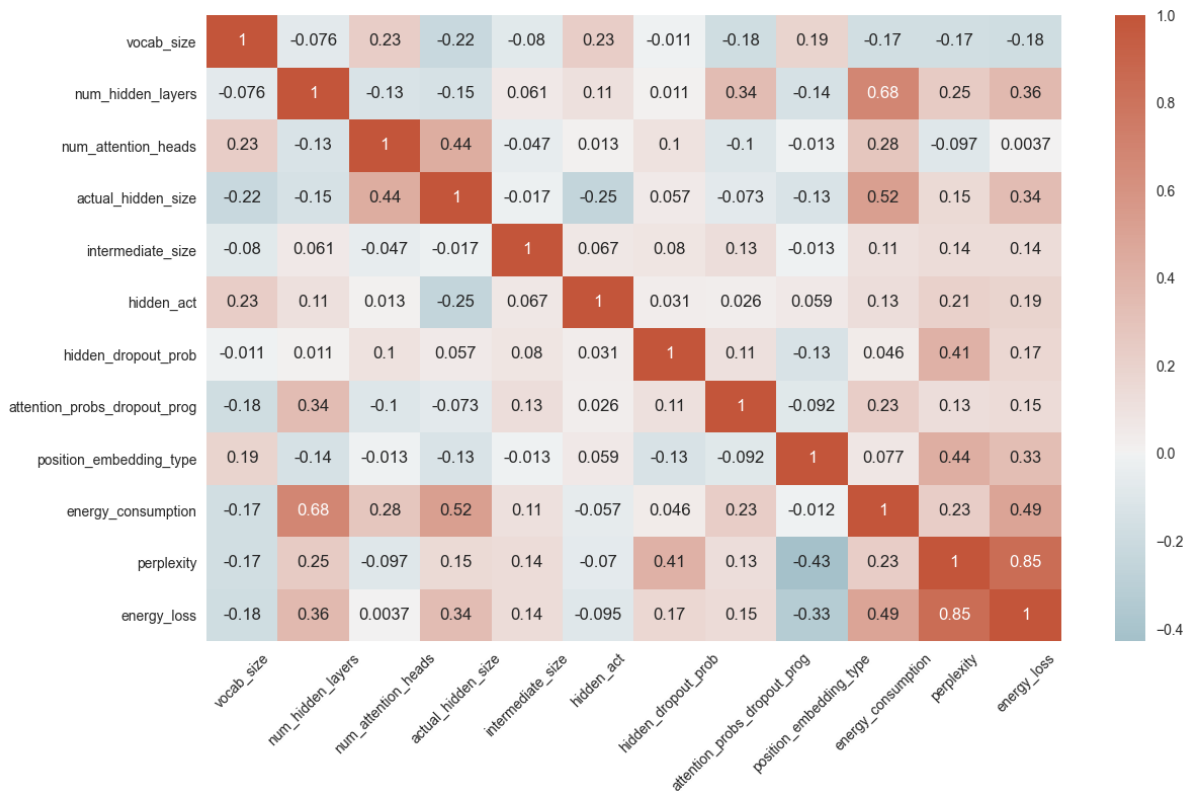


Figure 10: A correlation matrix of the worst 15% of models with regard to perplexity.

els that perform well in terms of perplexity. When looking at Table 9, the parameters for the models in the top 15% worst percentage are definitely larger than those in the top 15% best percentage for perplexity in most aspects. Hidden size and layers are much increased, the same with the intermediate size. All dropout probabilities are also higher, which could be a reason as to why some of these models keep performing terribly - whatever they learn, they end up forgetting, thus making it harder to train a bigger model.

	Appearances
relative_key_query	8
relative_key	7
absolute	8
GELU	7
GELU_new	7
ReLU	5
SiLU	4

Table 10: Count of activation functions and position embedding types in the worst 15% of models wrt: perplexity.

When looking at Figure 10, the activation func-

tion has a negligible effect on both perplexity and power consumption in terms of its correlation, but the positional embedding type has a significant impact on perplexity, with a correlation of -0.43. Looking at the difference between the choice of these between the best and worst models in terms of perplexity from Table 10 and Table 8, the better models all tend to use relative key query, whereas the distribution for the worse performing models is more uniform over the three choices. As most of the best models in terms of energy consumption and energy loss primarily use relative key query, the results suggest that there is little to no reason to use another type.

E Clustering

In this section, the data will be clustered in order to find commonalities among the different models, and group them into the three found clusters. Those clusters will then be described in detail, pointing out interesting characteristics in each one.

The data were clustered using DBSCAN algorithm, which is a density-based clustering algorithm designed to do spatial clustering with handling of noise (M. Ester and Xu, 1996). The density-based clustering method that is DBSCAN

does not handle varying axis ranges well, and given that energy consumption ranges from [0.529478; 12.986432] and perplexity ranges from [15.14847702; 2021.91427], normalization has to be done. The intent for this normalization is to have both axes have a mean of 0 and a standard deviation of 1. One of the consequences of doing this is that the distance used to specify clusters loses some of its intuitive understanding of what distance is now that both axes have been normalised. While this is true, the distance can now be used to properly be used to identify clusters for both metrics, rather than constraining the perplexity to the range of energy consumption. This enables us to find more than just vertical clusters.

For the actual clustering, a distance of 0.4 was chosen with the requirement of needing 5 samples to form a cluster. The clustering showcases the aforementioned outliers which ended at around 2000 perplexity but also finds another cluster close to our primary cluster, as well as 20 different outliers who were unable to be clustered, marked as black dots on Figure 11.

The exact distribution of the clusters can be seen here on Table 11. It’s interesting to note that had the minimum samples been a little higher, cluster 3 would not have existed.

	Number of Models
Cluster 1	118
Cluster 2	10
Cluster 3	6
Outliers	20

Table 11: Table showcasing the three cluster distributions and outliers for the DBSCAN clustering

Given that these three clusters were found by our algorithm, it’s only right to analyse those clusters further by constructing correlation matrices for each cluster. The first cluster, which can be seen in Figure 12, is the cluster featuring all the best performing models, and also subsequently the biggest cluster out of the three. The trends for this cluster are similar to what has been previously seen - the correlation matrix indicates that the number of hidden layers and energy consumption have a strong correlation, and the same goes for hidden size to a much lesser degree. Furthermore, the hidden size has a negative correlation with perplexity, and the hidden dropout probability has a correlation with

perplexity. This indicates the higher the hidden size, the lower the perplexity, and the lower the hidden dropout probability, the lower the perplexity. Interestingly, energy consumption and perplexity has a negative correlation, but it is incredibly weak, and therefore not indicative of anything. Most of our Pareto entries are in this cluster, which contributes to the negative correlation, but given that there are 118 entries here, as seen on Table 11, it indicates that a predominant amount of the models can still be optimised on both parameters.

When looking at the average parameters for the first cluster, it’s decently similar to that of the best 15% of models, as seen in Table 2 in section 4.2. Mostly all parameters follow an increasing trend compared to the other table, except for the actual hidden size, which remains slightly lower than when looking at the best 15% of models. As can visually be seen in Figure 11, this cluster features predominantly low perplexity, at an average of 56.28, whereas the energy consumption is slightly higher, sitting at an average of 2.34.

	Average	Std. Deviation
vocab_size	20491.32	7580.56
actual_hidden_size	263.55	208.74
num_hidden_layers	3.34	2.68
num_attention_heads	9.16	4.87
intermediate_size	891.15	730.67
hidden_dropout_prob	0.33	0.16
attention_dropout_prob	0.36	0.20

Table 12: Table with average hyperparameters of the top 15% of models in cluster 1.

Furthermore, it’s also seen that there is a heavy bias towards the position embedding type `relative_key_query`, which was also the case among our best 15% of models. The activation function remains slightly more spread out, with GELU and SiLU being the predominant activation functions, followed closely by ReLU. Among the best 15% of models, GELU was the predominant function, with SiLU at half of the occurrences, as can be seen on Table 2

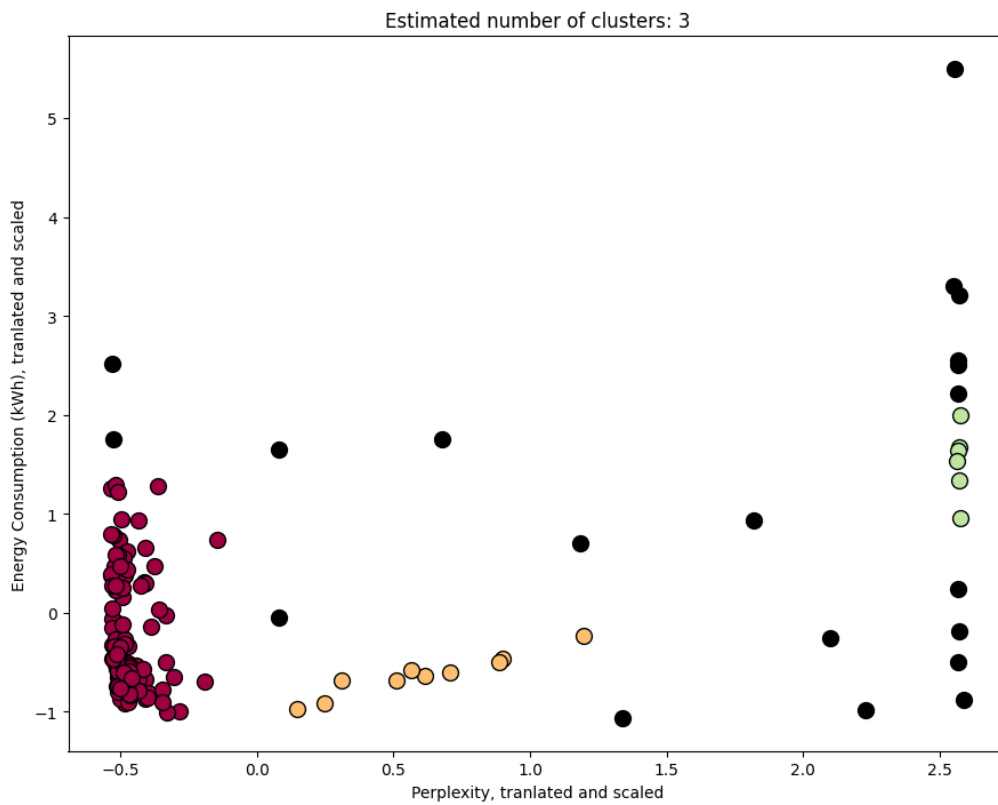


Figure 11: The data clustered using DBSCAN with axes scaled and translated for a mean of 0 and a standard deviation of 1.

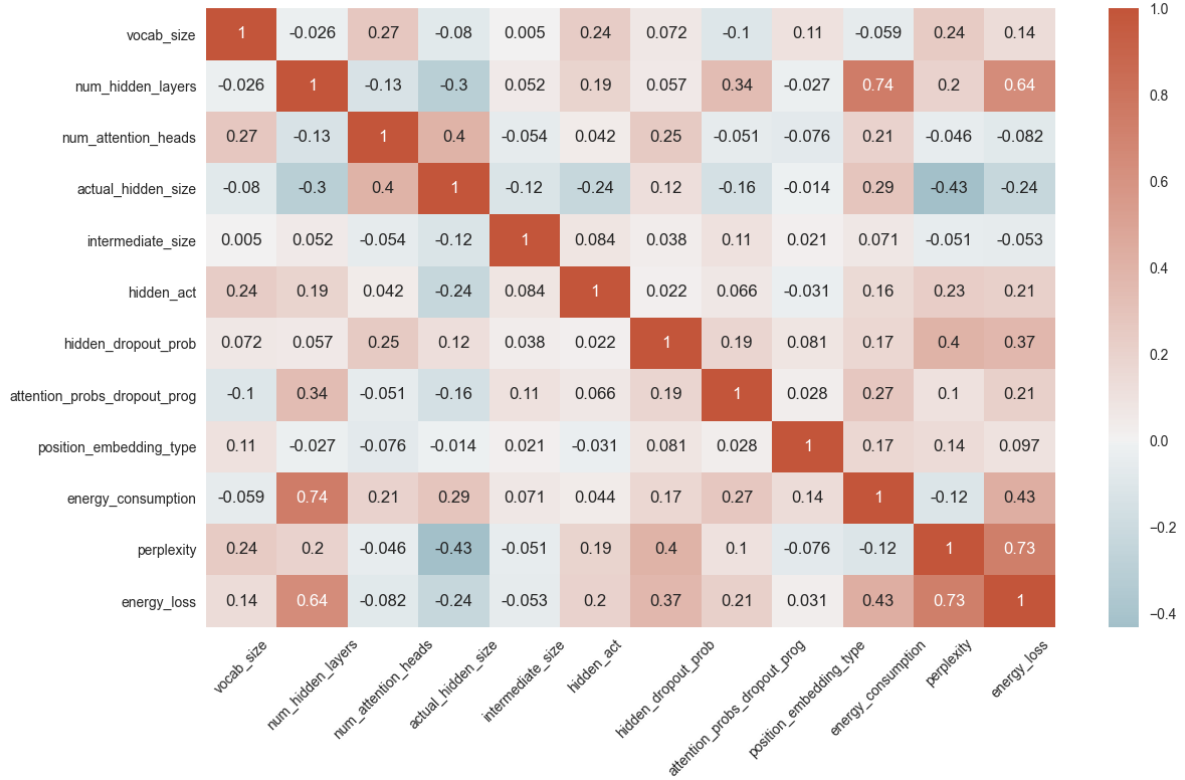


Figure 12: The correlation matrix for the first cluster

	Appearances
relative_key_query	94
relative_key	12
absolute	12
GELU	37
GELU_new	15
ReLU	29
SiLU	37

Table 13: Count of activation functions and position embedding types for the first cluster.

When looking at the second cluster on Figure 11, it can already be seen from this figure that there might be some linear tendencies in the models from this cluster. And as can be seen in Figure 13, energy consumption, perplexity, and energy loss all have a correlation close to 1, which indicates this to be the case. The correlation matrix also indicates linearity in the parameters, given that all parameters have very close correlation ratios between energy consumption, perplexity, and energy loss.

Because of the previously explained linear tendency, a linear regression was done on the models in cluster 2 as can be seen in Figure 14. If one extends beyond the scope of the cluster, it's pos-

sible that optimal models which decrease in both energy consumption and perplexity, which also lay on this line. Given that our hyperopt optimization was done on a limited scope, it's not possible to see if this is the case, although this linear regression strongly indicates that there are more models along this line that have not been explored.

The third cluster, being the cluster with all of the detected high perplexity models, can be seen in Figure 15. It features two strong correlations between the number of attention heads and energy consumption, as well as the hidden layer size and energy consumption. This indicates that the high amount of attention heads and hidden layer size have a strong influence on the increase of energy consumption. If one looks at the specific models of this cluster, it can be seen that they on average have 9.5 attention heads and a hidden size of 620.5. If one compares it to the results seen in Table 2, the average number of attention heads increases by approximately 1.5 and the average hidden size by 350. There is also a strong correlation between position embedding type and perplexity with 0.98, which strongly indicates that the position embedding type is tied to an increase in perplexity. Looking at the models in the cluster, it's seen that there's a uniform

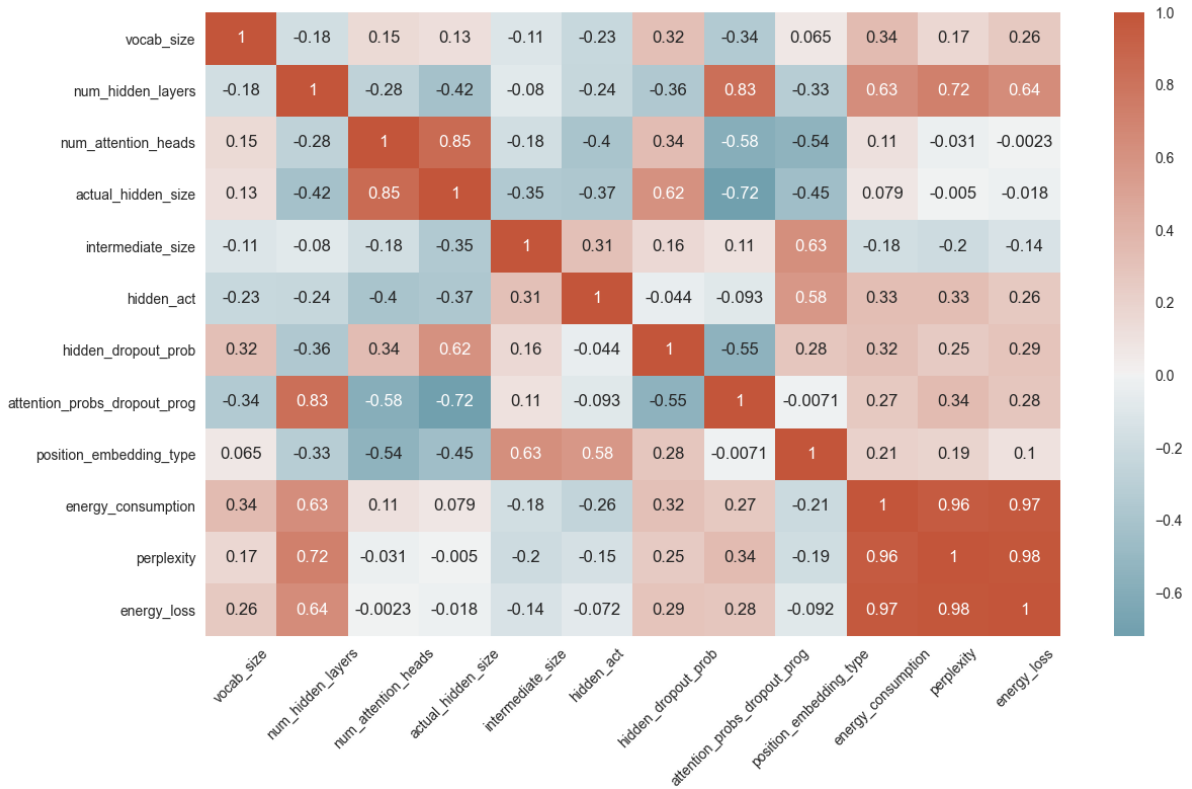


Figure 13: The correlation matrix for the second cluster

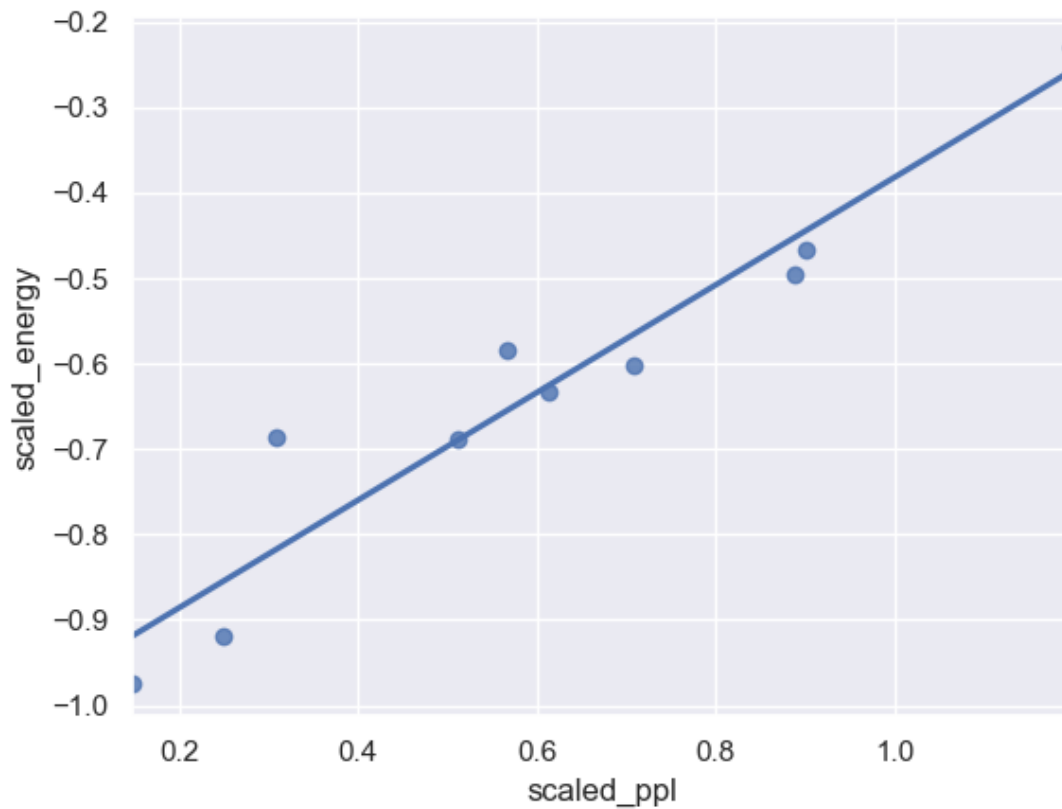


Figure 14: Regression done on the models of cluster 2

distribution between the three-position embedding types in the 6 models in the cluster. When referring back to Figure 10 it is seen that when looking at the 15% worst models with regard to perplexity, the correlation is still there, but not nearly as strong. Furthermore, the same uniform distribution of position embedding types can be seen on Table 10, which indicates that there are too few models to say anything about this correlation.

In Table 14 the 19 models situated on the Pareto curves through all 10 epochs are summarised along with the frequencies of occurrences, and for which epochs they occurred on the Pareto curve. Furthermore, the results with regard to energy consumption and perplexity are also showcased for the models after the 10th epoch, regardless of whether the model was a part of the Pareto curve at this point. Continuing on this point it is also important to note that towards the end of the Pareto curve, that the models most likely won't be the most effective models. This is for example the case with model number 103, which is a part of the Pareto curve for the 10th epoch, but with a perplexity of 1274.87. Model number 103 will probably not be a model you would want to focus on (especially since it was trained over 10 epochs) but can be used as a boundary to attempt to narrow down the search space, and thus still provide valuable information. Similarly, a model such as number 25, has a really strong score with regard to perplexity, but there is a number of models with similar perplexity scores while still having a much lower energy consumption. An important note here is to mention the possibility of training some of the low-cost models for even more epochs, to fairly compare their energy consumption vs perplexity with those of similar models. The reasoning here is if a low-cost model, such as model 103, would be comparable in perplexity to some of the other models after say 20 or 30 epochs, its energy consumption might still be in a relatively comfortable spot compared to another model with an energy consumption similar to that of model 25. The reasoning behind this argument comes from the linear cost of energy consumption for each epoch, such that even after 30 epochs of training, assuming a continuously linear tendency for the following epochs, model number 103 would have an approximate energy consumption of 0.58 (the energy consumption for model 103 after 10 epochs) times 3 (to go from 10 to 30 epochs) would be $0.58 \cdot 3 \approx 1.76$.

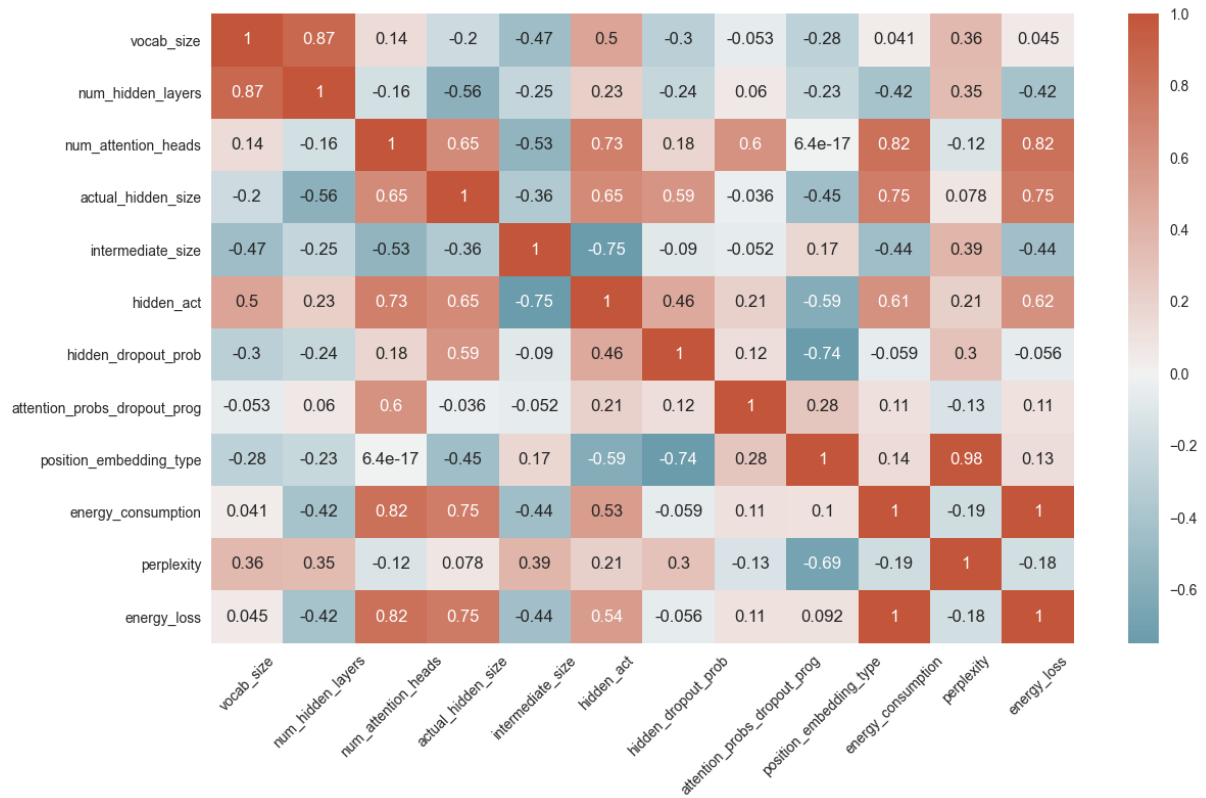


Figure 15: The correlation matrix for the third cluster

id	energy_epoch_10	PPL_epoch_10	occurrence	Epoch Occurrence
10	1.73 kWh	23.66	10	1,2,3,4,5,6,7,8,9,10
25	3.63 kWh	14.31	10	1,2,3,4,5,6,7,8,9,10
29	1.86 kWh	17.61	10	1,2,3,4,5,6,7,8,9,10
81	0.72 kWh	151.56	10	1,2,3,4,5,6,7,8,9,10
97	1.15 kWh	31.77	10	1,2,3,4,5,6,7,8,9,10
103	0.58 kWh	1274.87	10	1,2,3,4,5,6,7,8,9,10
48	0.90 kWh	46.47	9	2,3,4,5,6,7,8,9,10
62	2.52 kWh	15.71	9	2,3,4,5,6,7,8,9,10
88	1 kWh	37.25	9	2,3,4,5,6,7,8,9,10
106	1.26 kWh	29	7	4,5,6,7,8,9,10
111	1.59 kWh	29.27	7	1,2,3,4,5,6,7
58	1.63 kWh	27.19	6	4,6,7,8,9,10
136	1.78 kWh	19.38	6	1,2,3,6,9,10
87	3.66 kWh	13.65	5	6,7,8,9,10
133	4.52 kWh	13.12	5	6,7,8,9,10
147	1.28 kWh	30.43	4	1,2,3,4
114	3.42 kWh	15.51	3	8,9,10
24	1.80 kWh	23.48	2	9,10
99	0.94 kWh	54.87	2	1,2

Table 14: Table over pareto entries, featuring energy consumption and perplexity at 10 epochs, the number of occurrences in the pareto curves of each epoch, and the specific epochs at which it occurs on the pareto curve.