New is not always better - A comparison of the performance and efficiency of two CNNs on a classification task

Christian Burmester, Maximilian Kalcher, and Marlena Reil

Implementing ANNs with TensorFlow, University of Osnabrueck.

March 2022

Abstract

Convolutional Neural Networks (CNNs) are deep neural models that reach promising results in image classification tasks. However, training these networks requires a high amount of energy which poses environmental concerns. With this work, we attempt to show that less complex neural networks are to be favoured in simple image classification tasks. By training two comparable neural networks, ResNet-50 and ConvNeXt, on the same dataset, we found that the less complex architecture of ResNet-50 exhibits a smaller GPU workload while still achieving sufficient accuracy on the classification of the data. We conclude our report with an outlook and thoughts for the community in respect to environmental concerns that the training of modern neural networks may raise.

1 Introduction

Machine Learning (ML) models are powerful tools that perform Computer Vision (CV) tasks such as image classification and object detection on a human-like level. Larger and more powerful neural networks are being developed and tested at a constant rate to solve even more complex problems in CV and other domains like speech recognition or machine translation. By shifting computations to Graphics Processing Units (GPU), neural networks can be trained multiple times faster than before [1]. However, with increasing complexity, ML tools consume a lot of energy. Training Google AI's language model MegatronLM, for example, required almost three times the amount of energy that an average US household uses annually [2].

With more and more tech companies running AI models on a large scale, ML can pose a serious threat to climate change. Energy production in form of fossil fuels causes increasing emissions of carbon dioxide (CO2) into the atmosphere.

Being part of the broader ML community, we want to face the trend of building large ML tools in a more critical manner. We propose to evaluate ML methods not only with respect to performance but also with respect to their sustainability.

Among the CNNs for image classification, a recent one presenting major improvements was ConvNeXt, which was proposed by Facebook AI Research (FAIR). This network is based on the older ResNet architecture, and has shown to reach state-of-the-art performance on the ImageNet dataset. It is even able to compete with Transformers on tasks such as object detection, image segmentation, and classification.

Due to its fundamental changes however, even the smallest version of the ConvNeXt, the "ConvNeXt-T" with 28M parameters, needs a lot more computational power than its predecessor, ResNet. The question now becomes clear: is it worth training such a large network to get fractional improvements in accuracy, when only performing a simple classification task?

In our work, we compared the performance and energy consumption of ResNet-50 and ConvNeXt, when trained on real-world data. Our aim was to show significant and meaningful differences to contribute to the trend of more energy-efficient models.

2 Methodology

We first built a ResNet-50 model and trained it on a classification task. We then deployed a ConvNeXt network on the same dataset and compared the results with respect to performance and energy efficiency.

2.1 Dataset

For reasons of comparability, we looked for a dataset with high quality and a sufficient sample size. We chose the BIRDS 400 SPECIES - IMAGE CLASSIFICATION dataset that is publicly available on Kaggle ([3]). The dataset comprises 400 classes of bird species with 58,388 train, 2,000 validation and 2,000 test images. The number of images per class is not balanced. Nevertheless, there are a minimum of 120 training images per class. Therefore, we did not expect the imbalanced classes to have an influence on our models' performances. The bird images come in 224 x 224 x 3 jpg format. Each image contains a single bird which covers at least 50 percent of the pixels.

The author of the dataset was able to reach an accuracy of above 90 percent on the correct classification of the birds ([3]). This, to us, indicated that the dataset was suitable for our work. It allowed us to solely focus on the models' architectures and their impact on the energy consumption. Our goal was not to reproduce or improve on accuracy levels.

2.2 Model selection and architectures

2.2.1 ResNet-50

Our focus lied on training a ResNet-50 to evaluate its performance and energy consumption. Deep Residual Neural Networks (ResNets) for Image Recognition were first proposed in 2015 by He et al. [4]. The main idea of the authors was to develop an architecture that overcomes the problem of decreasing accuracy that arises in deeper neural networks. Although being more expressive than shallower models [5], it has been shown that with increasing depth, the accuracy saturates and

Figure 1: Sample images of the dataset.

decreases at some point [4]. ResNets are CNNs that use skip connections to overcome this problem. Skip connections add the input of a block consisting of several layers directly to its output. These direct connections basically serve as identity mapping as the input is passed on without being put to any other function before being added to the output of the convolutional layers. In this way, the networks gets less complex and easier to train.

ResNets have been shown to be especially useful for visual analysis tasks. They won prizes in the ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC, 2015) and Microsoft Common Objects in Context 2015 (MS COCO, 2015) [4]. He et al. developed several ResNet versions that vary in the amount of layers. All variants achieved top results when compared with State-of-the-art models at the time. Looking at the top-5 error rates (in percent), 5 different versions of the ResNet baseline model scored a rate of 5.71 percent or below on the ImageNet validation set. The best scoring competing model PReLU-net also had an error rate of 5.71 percent. VGG and GoogLeNet attained error rates around 7-8 percent [4]. We decided to implement a ResNet-50 which comprises 50 layers. Our model is built up 16 blocks of convolutional layers, alternating with normalization layers and ReLu activation functions. Fig. 2 shows the structure of ResNet-50.

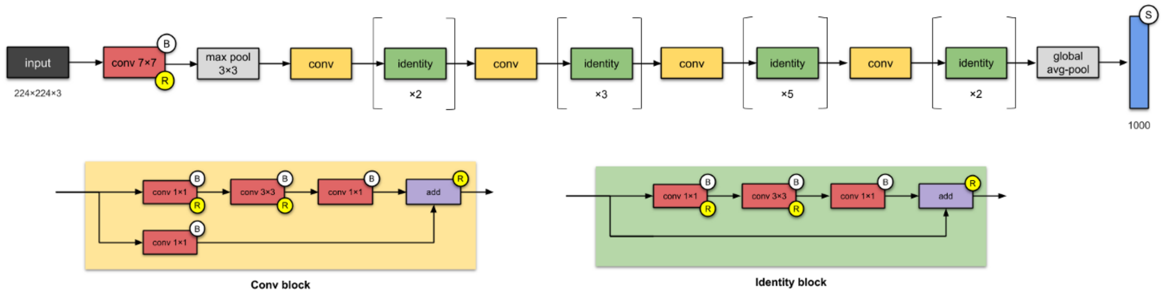

Figure 2: Building architecture of ResNet-50.
[6]

2.2.2 ConvNeXt

To be able to put the GPU workload of ResNet-50 during training into perspective, we deployed a second model on the same dataset. We chose the ConvNeXt model, which was introduced by FAIR in 2022. The main reason for choosing this model was that it inherits its main architecture from ResNet-50. This, we anticipated, would facilitate an objective justification of our assumptions and enable us to track potential sources of energy efficiency or inefficiency.

The developers of the ConvNeXt model have shown that its model is able to reach an accuracy of up to 86 percent on the the smallest variant of the ImageNet dataset (1K classes). Comparing this to the 79 percent ResNet-50 was able to reach, it shows a 7-8 percent increase in accuracy. In the larger dataset with 22K classes, the ConvNeXt model is compared to Vision Transformers (ViT) from 2020 and Swin Transformers (SwinT) from 2021. Between these three models, ConvNeXt achieves the highest accuracy score of around 90 percent with both ViT and SwinT recording slightly lower (82-87 percent) figures (see Fig. 3). The superiority in accuracy of the ConvNeXt model is rooted in its network architecture. FAIR advises that ConvNeXt is constructed entirely from standard ConvNet modules and claim that it is accurate, efficient, scalable and very simple in design [7]. ResNet-50 served as the basis model for the construction of ConvNeXt. A large portion of the design changes however, are inspired by the modern Transformer networks. Fig. 4 summarises all design changes that were implemented. It also indicates how much the individual changes distributed to the overall performance increase (in percent).

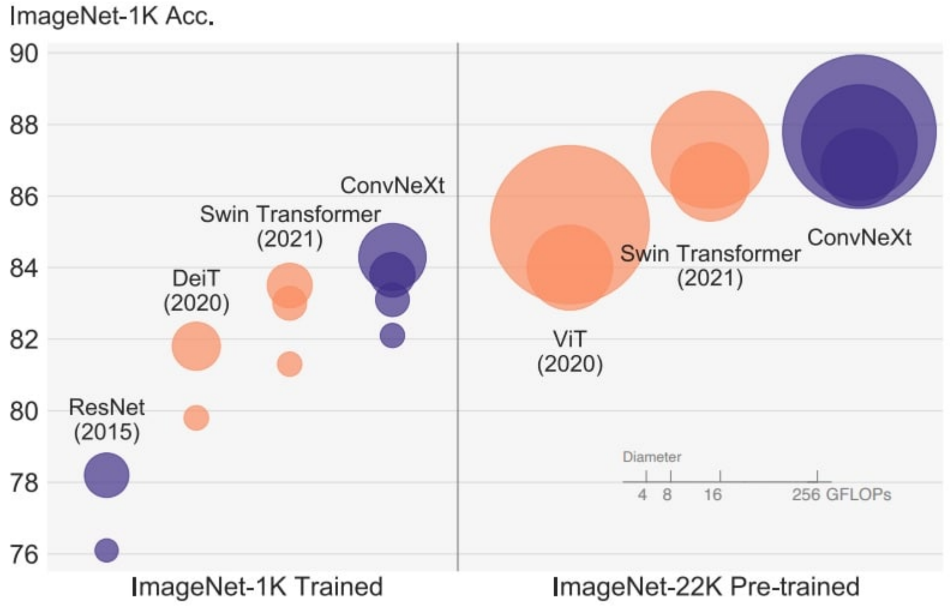

Figure 3: Comparison of classification results from the original paper.
[7]

The design changes can be separated into five different levels. The first changes appeared in the macro design of the model. The stage ratio was adjusted from (3,4,6,3) blocks in each stage

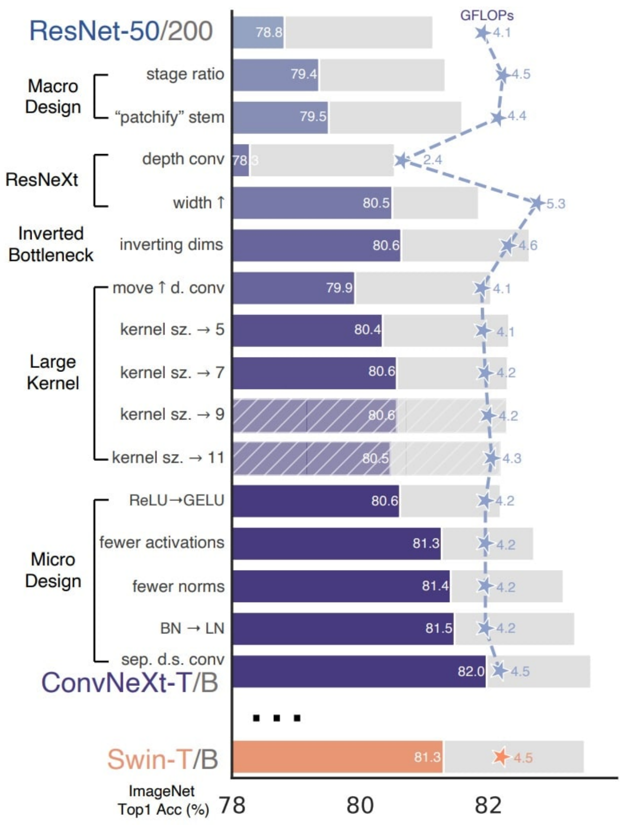

Figure 4: Design changes of ConvNeXt based on ResNet-50 from the original paper. [7]

in ResNet-50 to (3,3,9,3) in ConvNeXt. This change is inspired by the compute ratio of SwinT (1,1,3,1) and allows for a more optimal distribution of computation throughout the entire network. The second change in macro design involves the stem cell structure. A stem cell controls how input images are processed before being fed through the layers of CNNs. Downsampling the inputs using standard convolutional kernels is common practice which has the goal to reduce redundancy in natural images. ConvNeXt deploys an approach that resembles the techniques used in ResNet-50 and SwinT. ResNet-50 uses a 7 x 7 convolutional layer with a stride of 2. This is followed by max pooling and thus resulting in a 4 x 4 downsampling of the input images. SwinT extract similarly sized patches however, by increasing the stride they produce non-overlapping convolutions. ConvNeXt introduces a "patchify" layer with a 4 x 4 kernel and a stride of 4, thus also attaining non-overlapping patches.

In the next level of macro design changes, ConvNeXt developers worked off the improved ResNeXt-ify, as introduced by Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks" in CVPR, 2017. The goal is to improve the floating point operations (FLOPs) / accuracy ratio in each forward pass. This is achieved by grouped convolution where convolutional filters are separated into different groups [7]. Since the groups work in parallel, the network's width is often increased for the network to learn a variety of low and high level features in the images. ConvNeXt adopted the core principles of this idea however, changed the standard grouped convolution to a depthwise grouped convolution. As can be seen in the code, the number of groups equals the number of channels. Each channel is then convolved over separately which resulted in a less complex and faster network. The network's width is expanded to assure the overall performance increase.

The next design change concerns the hidden layer of the MLP block which, in the case of ConvNeXt, is four times wider (384) than the input dimension (96). This is referred to as "Inverted Bottleneck" and is used in every Transformer architecture. The position of the spatial depthwise convolution was moved up by one layer (see Fig. 5) to realise the next design change. As seen in Vision Transformers, global receptive fields support the idea of non-local self-attention. By using larger kernels, ConvNeXt adopted this idea. It is known that global receptive fields do not improve performance however, it is proven that the opposite i.e. having smaller kernels and thus limited and very local receptive fields, causes a decrease in performance. By moving the depthwise convolution to a position prior to the MLP block, it took care of the complex and rather inefficient modules in the MSA layer with large kernel convolutions. The "heavy lifting" in the MLP blocks is allocated to the efficient, dense 1 x 1 layers.

In our implementation, we tried to utilize a version of the ConvNeXt with a similar number of parameters like the ResNet-50 to obtain a fair comparison. The lightest and most similar version was the ConvNeXt-T with 28M parameters.

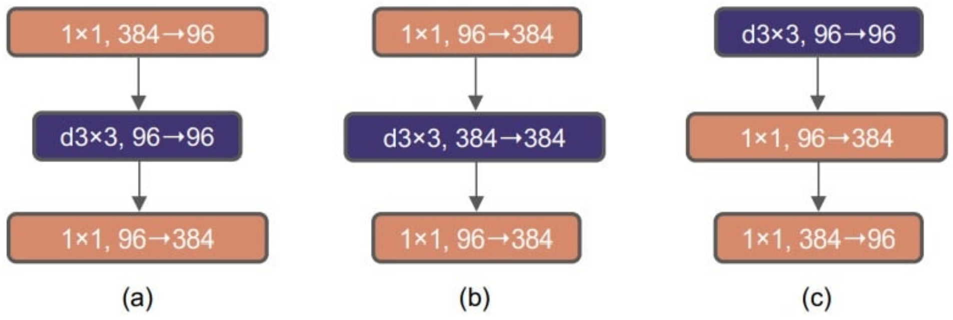

Figure 5: Block modifications from the original paper. (a) ResNeXt block; (b) Inverted bottleneck of ConvNeXt; (c) Depthwise convolutional layer (in purple) moved up.
[7]

On a micro level, the developers at FAIR substituted the ReLU activation functions of ResNet-50 by its smoother variant Gaussian Error Linear Unit (GELU), although this did not bring about any increase in performance. Overall, fewer activation functions are used. Replicating the style of a Transformer block, ConvNeXt drops all the GELU layers from the residual block except the one between two 1 x 1 convolutional layers. In the end, only one GELU activation was applied in each block. [8].

In ConvNeXt the number of normalization layers is limited. Further, most of the batch normalization (BN) was replaced by layer normalization except one BN layer just before the 1 x 1 convolutional layers. Traditional ConvNets as well as Resnet-50 make use of BN as it improves the convergence and reduces overfitting (reference original paper). As final step, FAIR added a separate downsampling layer of 2 x 2 convolutions with stride 2 between each stage to capture spatial resolution in the image patches.

3 Results

We used both the ResNet-50 and the lightest ConvNeXt version, the ConvNeXt-T, to classify 400 bird species. During the training, we also tracked the GPU workload.

3.1 Performance

3.1.1 ResNet-50

The ResNet-50 is trained, like the original model, on 224x224x3 images. Before feeding the data into the network, we normalized the images and shuffled them. The training data was also additionally randomly augmented. Augmentation steps included random rotations, horizontal and vertical offset, cropping and zooming, horizontal flips, ZCA whitening, random brightness change and nearest fill-modes, which segment parts of the image as a preprocessing step. The original model was trained for 90 epochs. But, by increasing the batch size to 32, our model could still reach a sufficient accuracy after 10 epochs. We used the Adam optimizer which combines an adaptive learning rate with advantages of momentum [9] and the categorical cross-entropy loss.

After 10 epochs, the model achieved a training accuracy of 0.78 with a loss of 0.83, a validation accuracy of 0.85 with a loss of 0.55 and finally a testing accuracy of 0.87 with a loss of 0.42.

3.1.2 ConvNeXt-T

In addition to design changes, training parameters were adapted to increase performance of ConvNeXt. Whereas ResNet-50 was originally trained for 90 epochs, the developers of ConvNeXt pushed this to 300 epochs fort their training. Due to mistakes in our own implementation of the ConvNeXt model or the nature of the training epochs, we were not able to train the model for longer than 6 epochs. Of the originally three data augmentation techniques, we were only able to recreate one: Mixup augmentation. Fig. 6 shows an example image created from the interpolation of two original images of the dataset. The two data augmentation techniques that are missing in our work are Cutmix and RandAug. ConvNeXt utilizes AdamW as optimizer during training, as opposed to Adam, used in the implementation of ResNet-50. AdamW is a variant of the optimizer Adam that has an improved implementation of weight decay [10]. Categorical cross-entropy loss was also used.

Due to the before mentioned circumstances, the model's training accuracy did not move significantly, staying at around 0.024 after 6 epochs. The loss only decreased minimally during that time.

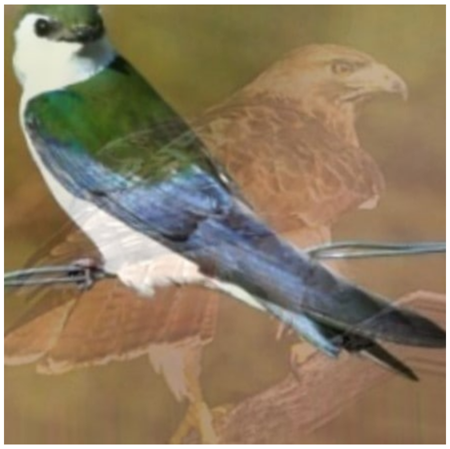

Figure 6: Mixup data augmentation of two sample images.

3.2 Power Consumption

The training of both models was performed on an NVIDIA GTX 1070, CUDA 11.6 and CuDNN 8.5. To measure the power consumption on the graphics card, we used the built-in command using **nvidia-smi**:

```
nvidia-smi --query-gpu=index,timestamp,power.draw,clocks.sm,clocks.mem,clocks.gr
--format=csv -l 10 -f ./GPU-stats.csv
```

Nvidia-smi is a command line utility that is based on the NVIDIA Management Library (NVML). Its purpose is to aid in the management and monitoring of NVIDIA GPU devices. This command records the current power consumption every 10 seconds using the -l argument, and stores it in the file *GPU-stats-*.csv* in the Data/GPU/ directory. For reference, the idle GPU power consumption is about 7-12W.

3.2.1 ResNet-50

Our ResNet-50 model trained for 10 epochs with about 810 seconds per epoch, totalling 2.3 hours of training. It consumed about 93474.32 W in total or **21.1 KWh**.

In Fig. 7 we can observe a snapshot of the power draw every 10 seconds. The power draw for the training steadily oscillated between 60W and 150W, never exceeding more than 160W.

3.2.2 ConvNext-T

Since the developers of the ConvNeXt themselves pushed the training to 300 epochs, it already suggests a higher energy consumption for the standard case. As we mentioned before, we stopped the training of our ConvNext-T model after 6 epochs and about 3.1 hours, since no progress in accuracy was being achieved. During this time, the network consumed about 113903.22 W in total or **29.68 KWh**.

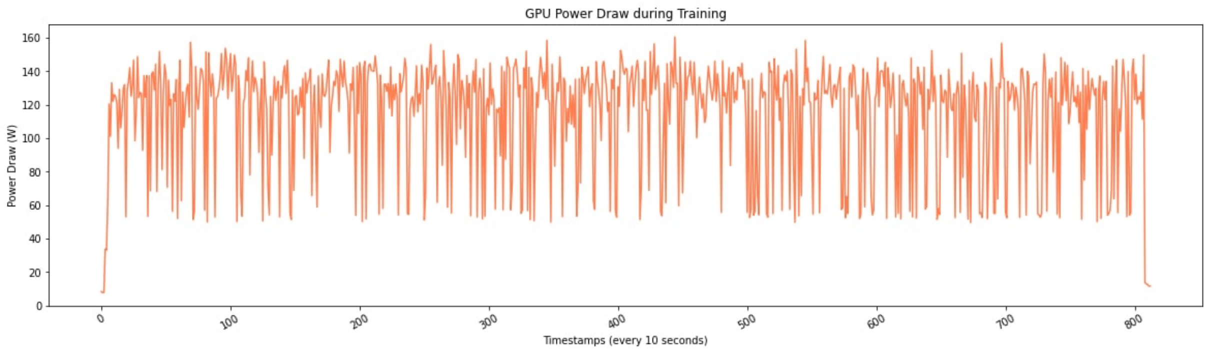

Figure 7: GPU Power Draw (W) every 10 seconds during training of ResNet-50.

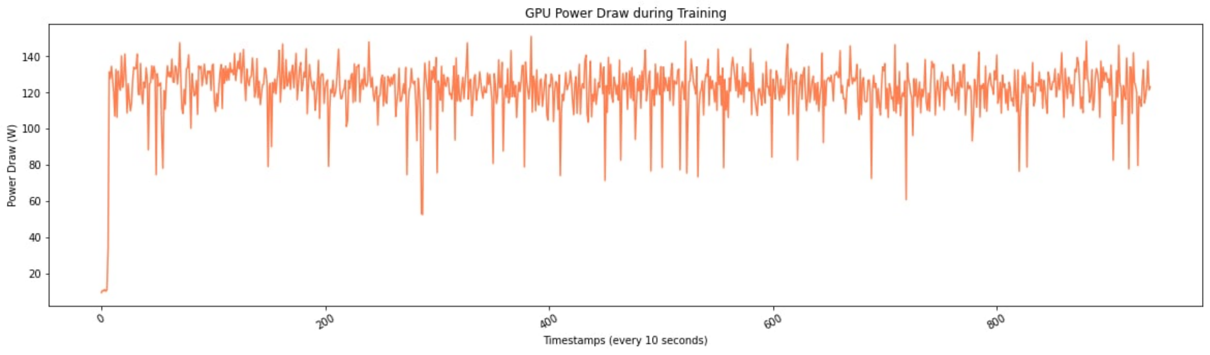

Figure 8: GPU Power Draw (W) every 10 seconds during training of ConvNeXt-T.

In Fig. 8 we can see the power draw every 10 seconds of our ConvNext-T. The power draw for the training here seems more steady, but also higher overall, staying around the 130W region.

3.3 Comparison

Both of our models, including performance and GPU-usage can be summarized within the following table:

Model	Parameters	Test Accuracy	Epochs	Average Time per Epoch	Energy Usage
ResNet-50	24,407,312	0.87	10	810s	93474.32W / 21.11KWh
ConvNeXt-T	28,127,728	DNF	6	1881s	113903.22W / 29.68KWh

Even though we chose the lightest version comparable to the ResNet-50, the ConvNeXt still had about 4 million more parameters. This, and the overall architectural choices, like advanced (depth-wise) convolutions and a more sophisticated optimizer, increased the average time per epoch significantly. At around 810 seconds per epoch, the ResNet-50 managed to be 2.3 times faster than the ConvNeXt.

We cannot really comment on an accuracy comparison, since only one model was trained successfully. But even after 10 epochs and 2.3 hours, the ResNet-50 reached a very desirable result, while the ConvNeXt probably needed many more epochs if implemented correctly, following the guide of the authors.

Even on a total energy consumption level, the ResNet-50 consumed about 30% less energy in its total training than the ConvNeXt did after only 6 epochs.

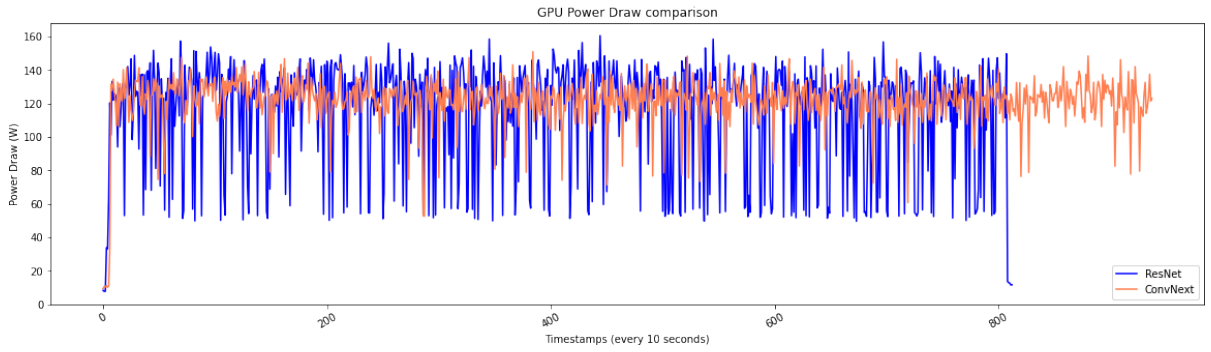

Figure 9: GPU Power Draw (W) of ResNet-50 and ConvNeXt during training

As for the raw GPU power draw comparison in figure 10, the ResNet-50 does not seem to consume as much power per epoch, due to the steady oscillations between 60W and 150W, while the ConvNeXt-T seems to need steady and high power at 130W most of the time.

4 Conclusion and discussion

As we can gather from the information above, the ConvNeXt does not only take significantly longer to train and use more power in the process, but also failed to reach a significant accuracy in our implementation. This was partly due to long training expectations but, also due to possible mistakes on our end. It still shows that an older model such as the ResNet-50 is very much usable and a lot more power friendly, despite being released 7 years earlier. The question arises, if more complex models are actually necessary to be deployed on selected tasks. Seeing the monetary and environmental costs the training of such models may have, we would argue that for simpler and very specific tasks, such as the one in our work, one should favour a less complex network architecture. While keeping the before mentioned costs low, such models are trained in much less time, underlying their better usability. In other, much more complex tasks, it seems that there will be no way around deeper and more complex models. As an example, GPT-3 is able do things that it was not even designed to do [11]. It can generalize past text generation and completion, solving math problems, translating text, describing objects, etc. but, just inferring with the model is computationally very expensive. In our project, a small ConvNet could perform relatively well on the task of predicting the class of a bird and the inference remains cheap and takes very little time. If we think ahead to use cases and physical objects with restraint memory and a limited amount of time, it is apparent that we should use smaller models instead of bigger ones to solve niche problems. We would like to reiterate that even though the training for the ConvNeXt was not conclusive, the power consumption and epoch lengths show a major difference impacting efficiency.

GPU workload however, may not be the only deciding factor of whether a network may be considered bad for the environment. Data is another relevant topic with respect to energy efficiency. Current research already explores ways to use smaller amounts of data (data distillation)[12], which would equally result in shorter training times.

Seeking cheaper training options will most likely be a more determining component in the training of models in the future, especially considering the political climate and rising energy costs. We believe it is important to steer towards a more efficient and environmentally friendly practice in the field in general.

Acknowledgments

Many thanks to the IANNwTF team for preparing us to be able to put this project into practice.

References

[1] D. Strigl, K. Kofler, and S. Podlipnig, "Performance and scalability of gpu-based convolutional neural networks," in *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, 2010, pp. 317–324. DOI: `10.1109/PDP.2010.43`.

[2] Labbe, "Energy consumption of ai poses environmental problems," T. Network, Ed., Aug. 2021, Online; posted 26-August-2021. [Online]. Available: `https://www.techtarget.com/searchenterpriseai/feature/Energy-consumption-of-AI-poses-environmental-problems#:~:text=AI%20energy%20consumption%20during%20training&text=A%20single%20V100%20GPU%20can,27%2C648%20kilowatt%20hours%20(kWh)`.

[3] Gerry, *Bird 400 - species image classification*, 2022. [Online]. Available: `https://www.kaggle.com/datasets/gpiosenka/100-bird-species/version/54`.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. arXiv: `1512.03385`. [Online]. Available: `http://arxiv.org/abs/1512.03385`.

[5] D. Rolnick and M. Tegmark, "The power of deeper networks for expressing natural functions," *CoRR*, vol. abs/1705.05502, 2017. arXiv: `1705.05502`. [Online]. Available: `http://arxiv.org/abs/1705.05502`.

[6] R. Karim, "Illustrated: 10 cnn architectures," Jul. 2019, Online; posted 29-July-2019. [Online]. Available: `https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d`.

[7] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[8] A. Singh, "Convnext: The return of convolution networks," Feb. 2022, Online; posted 10-February-2022. [Online]. Available: `https://medium.com/augmented-startups/convnext-the-return-of-convolution-networks-e70cbe8dabcc#:~:text=ConvNeXt%20uses%20depthwise%20convolution%2C%20a,information%20in%20the%20spatial%20dimension`.

[9] *Adam.* [Online]. Available: `https://peltarion.com/knowledge-center/documentation/modeling-view/run-a-model/optimizers/adam`.

[10] *Adamw.* [Online]. Available: `https://peltarion.com/knowledge-center/documentation/modeling-view/run-a-model/optimizers/adamw`.

[11] R. Dale, "Gpt-3: What's it good for?" *Natural Language Engineering*, vol. 27, no. 1, pp. 113–118, 2021. DOI: `10.1017/S1351324920000601`.

[12] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv preprint arXiv:1811.10959*, 2018.