Optimal Policy for Deployment of Machine Learning Models on Energy-Bounded Systems

Seyed Iman Mirzadeh , **Hassan Ghasemzadeh** ,
Washington State University, USA
{seyediman.mirzadeh, hassan.ghasemzadeh}@wsu.edu

Abstract

With the recent advances in both machine learning and embedded systems research, the demand to deploy computational models for real-time execution on edge devices has increased substantially. Without deploying computational models on edge devices, the frequent transmission of sensor data to the cloud results in rapid battery draining due to the energy consumption of wireless data transmission. This rapid power dissipation leads to a considerable reduction in the battery lifetime of the system, therefore jeopardizing the real-world utility of smart devices. It is well-established that for difficult machine learning tasks, models with higher performance often require more computation power and thus are not power-efficient choices for deployment on edge devices. However, the trade-offs between performance and power consumption are not well studied. While numerous methods (e.g., model compression) have been developed to obtain an optimal model, these methods focus on improving the efficiency of a "single" model. In an entirely new direction, we introduce an effective method to find a combination of "multiple" models that are optimal in terms of power-efficiency and performance by solving an optimization problem in which both performance and power consumption are taken into account. Experimental results demonstrate that on the ImageNet dataset, we can achieve a 20% energy reduction with only 0.3% accuracy drop compared to Squeeze-and-Excitation Networks. Compared to a pruned convolutional neural network for human activity recognition, while consuming 1.7% less energy, our proposed policy achieves 1.3% higher accuracy.

1 Introduction

With the world witnessing an unprecedented growth in both Internet of Things (IoT) and Artificial Intelligence (AI), new research avenues at the confluence of these two areas are emerging. On one hand, the number of connected IoT devices is expected to reach 41.6 billion by 2025 Shirer and MacGillivray [2019]. On the other hand, the worldwide spending on AI systems is projected to reach \$77.6 billion in 2022 D'Aquila and Shirer [2019]. These trends suggest that the intersection of IoT and AI, *intelligent edge*, is one of the most promising and demanding future research directions. It is already expected that the AI edge processors will reach a unit shipment of 1.5 billion by 2023 Shirer and Palma [2019]. There are several motivations behind performing machine learning on the edge systems. These motivations include latency, connectivity, energy consumption, security, and privacy. For example, transmitting data over wireless network raises security concerns in mission-critical systems such as security cameras and health monitoring systems. Moreover, for privacy-preserving reasons, the demand to push the computation from the cloud to the edge has increased as users prefer not to share their private data.

However, in a burgeoning era that promises integration of machine learning models and IoT devices, various challenges emerge in realizing the vision of intelligent edge. One prominent challenge arises from stringent-constrained resources (e.g., compute power, battery capacity, memory) that are available on the edge devices. For instance, current high-performance deep learning models for image recognition include millions of parameters and billions of operations per inference He *et al.* [2015]; Huang *et al.* [2016]. While this amount of computation is within the computing budget of a powerful cloud-based server, it is beyond the capability of an embedded computer. In addition to the limited compute power of an edge device, we note that many of these devices (i.e., wearable sensors, health trackers, security cameras) are battery-powered necessitating days or even weeks of continuous operation prior to a battery recharge.

To address the aforementioned challenges, much effort has been made. Several special-purpose processor chips are designed to perform training or inference on edge devices with improved performance Jouppi *et al.* [2017]; Franklin [2019]. Furthermore, techniques such as weight pruning Zhu and Gupta [2017], quantization Jacob *et al.* [2017], and knowledge distillation Hinton *et al.* [2015] are developed to compress the models. Additionally, advances in neural architecture search introduce models with much fewer parameters while delivering a performance comparable to gigantic handcrafted models Pham *et al.* [2018].

The methods for model compression and architecture search, however, focus on obtaining a single model, optimized according to a particular objective such as accuracy or power-efficiency, for deployment on the device. We argue that an efficient deployment of multiple models can outperform the performance of the single model approach. This area of research has remained virtually unexplored to date.

Our contributions can be summarized as follows:

1. We define and formulate the optimal deployment problem as an optimization problem.
2. We provide a rigorous theoretical analysis of the deployment problem and its practical extensions.
3. By performing comprehensive experiments on different machine learning tasks, we show the performance gain of our proposed solution.

Many practical applications can benefit from our proposed framework. For instance, our approach allows for easy deployment of object detection models on security cameras with the guarantee of operating for a week and still be sure that the chosen models will provide the highest accuracy among all other model combinations.

2 Background

In this section, we review some of the most important research ideas, developed to overcome the challenge of deploying machine learning models on edge devices and their relation to our work.

Model Compression

There are various methods developed to compress machine learning models. Generally, these methods are based on two ideas. The first idea is to reduce the number of parameters. Optimal Brain Surgeon Hassibi *et al.* [1993] uses the Hessian of the loss function to measure importance of the parameters.Han *et al.* [2015] prunes weights whose magnitude is lower than a threshold. The method in Denton *et al.* [2014] finds a low-rank approximation of weights with minimal accuracy drop. The same idea of reducing the number of parameters is used in Keerthi *et al.* [2006] to reduce the number of support vectors for support vector machine models and thus reduce the complexity. The other idea is to work with lower precision (fewer bits per weight). For instance, BinaryNet Courbariaux and Bengio [2016] limits the parameters to have 1-bit representations. Recently, Jayakodi *et al.* [2020] proposed to design classifiers of increasing complexity using pre-trained Convolutional Neural Networks to perform input-specific adaptive inference.

Another related research direction is to use the teacher-student learning paradigm for compression. Instead of reducing a large model into a smaller model, these methods start with the smaller model as the student who tries to learn from the large model as his teacher. Knowledge Distillation methods Hinton *et al.* [2015]; Mirzadeh *et al.* [2019] generalized this idea where the student learns from both the teacher and the data instead of learning only from the teacher. Knowledge Distillation methods share some similar ideas with Transfer Learning methods Alinia *et al.* [2020].

Artificial Intelligence Chips

The undeniable and critical need for performing AI on edge started a race on the design and development of AI chips. Companies are developing custom builds for general- or specific-purpose AI applications. From Nvidia's Jetson family Franklin [2019], to Intel's Nervana Neural Network Processor Intel [2019], all serve the same goal: fast and efficient performance on the edge. These chips are already being used in the production level.

One key takeaway from the presented history of the edge AI, is that the majority of the mentioned methods focused on one question: "How to obtain a better model?". Here, the word "better" might means having a smaller model and thus less energy consumption, and sometimes having better performance with the same computation complexity. Besides, a majority of the mentioned methods are focused on a single family of machine learning models. However, in this work, we present a general framework that works with any machine learning model, regardless of whether that model is a neural network or a support vector machine. More importantly, our framework provides bounded guarantees on energy consumption or performance of its proposed model deployment, an important requirement that has been overlooked by other researchers.

3 Optimal Deployment Policy

In this section, we first provide a statement of the optimal deployment problem followed by a formal definition. Then, we formulate and analyze the problem and its extensions from several aspects such as time complexity and approximation bounds.

3.1 Problem Statement

Given a supervised learning problem, let $\mathcal{M} = \{m_1, m_2, \ldots, m_n\}$ be a set of n trained models. We use the term *"candidate models"* or *"model pool"* throughout this paper to refer to $\mathcal{M}$. Let $\mathcal{P} = \{p_1, p_2, \ldots, p_n\}$ be the corresponding real-valued performance measures for the models in $\mathcal{M}$. Furthermore, let $\mathcal{C} = \{c_1, c_2, ..., c_n\}$ represent the set of cost values where the c_i represents the cost due to making one inference by executing model m_i. Here, performance can be any numeric metric (e.g., classification accuracy) used to assess the performance of the models. Note that cost can refer to any constraint (e.g., power consumption, inference time) for the deployment of the model on an edge device. Our aim is to deploy a combination of the models for an expected duration. To represent this duration, we can use the number of inferences we expect our system to make in each time unit and then multiple it by the number of time units to obtain the total expected inferences, K. Finally, we want to maximize our "objective" while we satisfy our problem "constraints". For deployment, there could be multiple "objectives" such as maximizing performance or minimizing energy consumption. Moreover, "constraints" represent our limitations, such as available energy (e.g., battery capacity) or a minimum level of desired performance.

We are interested in solving the problem with the discussed constraints. Otherwise, for example, having no limit on en-

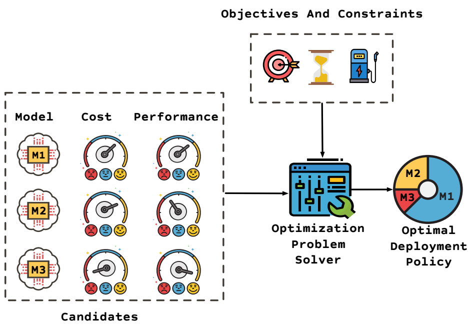
Objectives And Constraints
Model
Cost
Performance
M1
M2
M3
Candidates
Optimization Problem Solver
Optimal Deployment Policy

Figure 1: An illustration of the optimal deployment problem.

ergy consumption, we can choose the model with the highest performance regardless of other limitations, an approach which is unrealistic for embedded devices.

3.2 Problem Definition

In this section, we present a formal definition of the deployment problem prior to discussing the mathematical formulation.

Definition 1. Optimal Deployment Problem *Given n models, each with a performance p_i and an inference cost c_i, and a maximum budget cost B to perform K inferences, the optimal deployment problem aims to find inference count x_i for each model m_i such that the overall performance is maximized without exceeding the budget B.*

We refer to the optimal values of the inference counts x_i as *Optimal Policy*. Note that prior research uses only one model for all the K inferences, thus rendering a solution that contains a *single* model m_i where x_i is set to the total number of inferences K. In contrast, by viewing "Optimal Deployment" as an optimization problem, we find a *combination of the models* whose overall performance outperforms the performance of the single-model solution.

3.3 Problem Formulation

With the introduced notations in Section 3.1 and the problem description, the natural objective would be to maximize the expected performance. Despite being correct, this objective neglects the fact that switching between models will also incur model loading costs. To take such costs into account, we propose to add a *penalty term* to the objective function. To this end, we add an auxiliary binary variable y_i, which will be 0 whenever a model is chosen and thus the objective will be penalized by the loading cost $u_i > 0$ for model m_i. The penalty term can be controlled by a parameter, λ, depending on the application. By adding this penalty term to the objective function, whenever a model is selected (i.e., $x_i > 0$), y_i will be forced to be 0 to satisfy Constraint (4). Another interpretation is that we reward the solutions that contain a fewer number of different models (i.e., $y_i = 1$).

$$\text{maximize} \quad \underbrace{\sum_{i=1}^{n} p_i x_i}_{\text{expected performance}} + \lambda \underbrace{\sum_{i=1}^{n} u_i y_i}_{\text{penalty term}} \tag{1}$$

subject to :

$$\sum_{i=1}^{n} c_i x_i \leq B \tag{2}$$

$$\sum_{i=1}^{n} x_i = K \tag{3}$$

$$x_i \leq K(1 - y_i) \tag{4}$$

$$x_i \in \mathbb{Z}_{\geq 0} \tag{5}$$

$$y_i \in \{0, 1\} \tag{6}$$

Note that (1) and (2) are in fact the expected performance and expected energy consumption. The term $\sum_{i=1}^{n} x_i$ is removed from the denominator because this value is set to be constant in the constraint (3).

It is straightforward to derive a new formulation from (1) to reach a policy where we obtain a lower-bound guarantee on the performance (P) while maximizing the battery lifetime of the system by minimizing the energy consumption. To this end, we can change (1) to "minimize $\sum_{i=1}^{n} c_i x_i$" where Constraint (2) will be replaced by "$\sum_{i=1}^{n} p_i x_i \geq P$". This formulation will also be applicable in practice because there are several power-bounded systems in security and healthcare domain that need to operate for as long as possible and the performance will be the secondary concern as long as it meets a minimum threshold.

3.4 Theoretical Analysis of the Problem

The objective function introduced in (1), is in the form of 2-dimensional Knapsack problem where (2) represents the first dimention, and (3) refers to the second dimension. Moreover, this very specific formulation is known as *cardinality constrained knapsack* or *Exact K-item Knapsack Problem(E-KKP)*. This problem is proved to be NP-Complete Kellerer *et al.* [2004].

However, there are two critical factors that we need to consider. First, from a practical point of view, our problem size is not large and the exact solution can be computed in a reasonable time. Specifically, it is shown that the E-KKP problem with n items and U as the upper bound on the optimal solution value, can be solved to optimality by dynamic programming in $\mathcal{O}(nKU)$ time and $\mathcal{O}(n + KU)$ space Kellerer *et al.* [2004]. Second, it is possible to apply linear programming relaxation techniques to achieve a near-optimal solution, which will be the topic of the next subsection.

Linear Relaxation and Approximation Algorithms

In the *exact k-item knapsack (E-KKP)*, we can change the Constraint (5) to $x \in \mathbb{R}_{\geq 0}$ and obtain a linear programming relaxation form. However, changing Constraint (6) to $y \in \mathbb{R}_{[0,1]}$ will not affect the problem because y is dependent on x.

Let the vector $x^{\text{LP}} = (x_1^{\text{LP}}, \dots, x_n^{\text{LP}}) \in \mathbb{R}^n$ and the scalar $z^{LP} \in \mathbb{R}$ denote the solution of the linear relaxation problem and the optimal value, respectively. Similarly, let the vector x^* and the scalar z^* be the optimal solution and its value for the Integer Program in (1). Then we have the following lemma:

Lemma 3.1. *There exists an optimal solution vector x^{LP} with at most two fractional values.* Kellerer *et al.* [2004]

Using this Lemma, we can derive the following theorem on an error bound for the linear relaxation approximation of the optimal deployment problem.

Theorem 3.2. *Let p_i, p_j, and p_k $i, j, k \in \{1, 2, \dots, n\}$ represent performance of the most-accurate model, second most-accurate model and the model with the minimum energy consumption, respectively. The approximation error of the linear relaxation algorithm, $\epsilon = z^* - z^{LP}$, is upper-bounded as stated by $\epsilon \leq \frac{(p_i + p_j - 2 \times p_k)}{K}$*

Proof. From Lemma 3.1, we know that the vectors x^* and x^{LP} are different at most two places where x^{LP} at those places are fractional. The maximum error will occur when these two indices are i and j (top two high-performance models). Then, by rounding x^{LP} at these indices and adding the fractional values to the index k, we obtain the maximum difference of $\frac{(p_i - p_k) + (p_j - p_k)}{K}$ and this completes the proof. □

One important result of Theorem 3.2 is that ϵ will decrease as K grows as we will observe this in practice in Section 5.3. Furthermore, using this relaxation technique, followed by a greedy algorithm for E-KKP, the run-time complexity will be $\mathcal{O}(n)$ Kellerer *et al.* [2004]. These results are promising since the optimal deployment problem can be solved quickly and accurately in practice.

There are several improved relaxation and rounding algorithms developed to find fully polynomial time approximation scheme (FPTAS) for the E-KKP problem. The earliest idea to construct a FPTAS for this problem was based on dynamic programming and profit scaling techniques, which run in $\mathcal{O}(\frac{nK^2}{\epsilon})$ time and $\mathcal{O}(n + \frac{K^3}{\epsilon})$ space Caprara *et al.* [2000]. The downside of such an algorithm is the dependence on the cardinality K. This has been improved recently to $\mathcal{O}(n + \frac{z^2}{\epsilon^2})$ time and $\mathcal{O}(n + \frac{z^2}{\epsilon})$ space where $z = \min\{K, \epsilon^{-1}\}$ Li *et al.* [2019].

4 Experiment Setup

4.1 Power Modeling

To measure the power consumption of different machine learning models with different implementations, we utilized Intel's Running Average Power Limit (RAPL) Weaver *et al.* [2012] implemented in the Likwid library Center [2019]. RAPL allows us to monitor energy consumption on the CPU chip and the attached DRAM. For a fair comparison, we used only a single core and fixed the clock frequency at 1.5GHz for all our experiments.

Moreover, to reduce the effect of loading models and the dataset at the beginning of inference, and to achieve more accurate measurements, we ran each model ten times and averaged the energy consumption and run-time values. For each model, we then subtracted the idle energy consumption of CPU during the run-time of the inference. Finally, for each experiment, we scaled the energy consumption of the models such that the energy consumption of the model with the highest accuracy be a hundred.

4.2 Datasets

UCI Human Activity Recognition

We used the "Human Activity Recognition Using Smartphones" (UCI-HAR) dataset Anguita *et al.* [2013] which contains sensor data such as accelerometer and gyroscope data for 30 patients, each doing six different activities. The sensor signals (accelerometer and gyroscope) were pre-processed to minimize the effect of high-frequency noise and then sampled in fixed-width sliding windows of 2.56 sec. and 50% overlap (128 readings/window).

We trained Decision Tree, Support Vector Machine, and Gradient Boosting classifier with ten decision tree estimators each with a maximum depth of 3. We adopted the scikit-learn Pedregosa *et al.* [2011] library to train and evaluate the classification algorithms. In addition, we trained two 1-D Convolutional Neural Networks(CNN) with two layers, both with 64 filters with a kernel size of 3. The first CNN was trained normally, while the second neural network was trained using low-magnitude pruning. Both neural networks were trained using the Adam Optimizer Kingma and Ba [2014] with Tensorflow library for 50 epochs with early stopping. For the classification task on this dataset, we used the objective introduced in (1) where K is set to 1000 inferences, $\lambda = 0.1 \times K = 100$, and u_i is set to the constant value of 1 to penalize selecting many model. To solve the integer and linear programming formulations, we used CVXPY library Diamond and Boyd [2016]; Akshay Agrawal and Boyd [2018] with the ECOS solver Domahidi *et al.* [2013].

ImageNet

The ImageNet classification dataset Russakovsky *et al.* [2014] has 1.28 million training images and 50,000 validation images that includes of 1000 classes. We used the official pre-trained models implemented in the Pytorch framework Paszke *et al.* [2017]. We also used VGG Simonyan and Zisserman [2015] with and without batch normalization Ioffe and Szegedy [2015], Resnet He *et al.* [2015], SE-ResNet Hu *et al.* [2018], MobileNet-V2 Sandler *et al.* [2018], and ShuffleNet-V2 Ma *et al.* [2018].

To benchmark the energy consumption of the models, we used 1000 random images of the validation images (one per each class) and reported the scaled energy consumption in the results and repeated each experiment five times and reported the average, as described in the power modelling section. Similar to the activity recognition dataset, we used (1) as our objective with $K = 1000$ and $\lambda = 100$. Here, we also used the CVXPY library to solve the integer and linear programming problems.

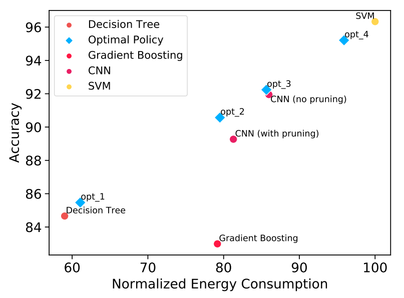

Figure 2: Benchmark of different models on UCI-HAR

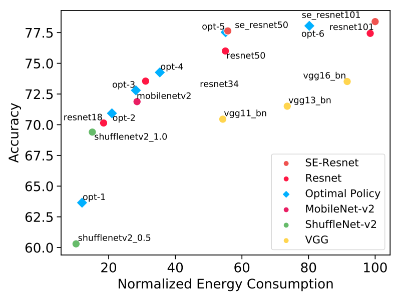

Figure 3: Benchmark of different models on ImageNet

model	**cost (energy consumption)**	**score (accuracy)**
Decision Tree	59.01	84.66
Optimal Policy 1	61.07	85.47
Gradient Boosting	79.18	82.99
CNN (with pruning)	81.29	89.27
Optimal Policy 2	79.51	90.57
CNN (no pruning)	85.98	91.95
Optimal Policy 3	85.65	92.24
SVM	100	96.33
Optimal Policy 4	95.9	95.21

Table 1: Comparison of our proposed policy with other models on UCI-HAR

model	**cost (energy consumption)**	**score (accuracy)**
ShuffleNet V2 - 0.5×	10.2	60.36
Optimal Policy 1	12.0	63.64
MobileNet V2	28.51	71.88
Optimal Policy 3	28.16	72.8
Resnet 50	55.05	76.0
Optimal Policy 5	55.1	77.53
SE-ResNet 101	100	78.39
Optimal Policy 6	80.24	78.05

Table 2: ImageNet Comparison Results

5 Experiments and Results

In this section, we study several experiments that we conducted and report the results. In the first part, we compare the optimal policy with the scenario where only one type of model is used. We report the results on both human activity recognition dataset and ImageNet dataset. In the second part, we will discuss the effect of the penalty term introduced in (1) on the performance and number of models that are selected in the optimal policy. Finally, in the third section, we will compare the accuracy of the integer programming formulation with the discussed linear programming relaxation. To have an easier comparison of the energy consumption of different models, we scaled the energy consumption between 0 and 100 where the model with the highest energy consumption is set to 100 and other models are scaled accordingly. Finally, all the accuracy numbers that are reported, represent the accuracy on validation data.

5.1 Effectiveness of the Optimal Policy

In this section, we compare the optimal policy with several methods, for two different classification tasks.

Activity Recognition

Figure 2 shows the energy consumption and accuracy of different models for the activity recognition task. Moreover, in Table 1, we have reported the results for several cases where we solved the optimization problem (1) with a power budget (B) close to the power consumption of available models so we can have a fair comparison.

We were able to achieve comparable performance and energy consumption with many standard methods that are currently used for this dataset. In some cases, we could even achieve better accuracy with a lower power budget. For instance, *"Optimal Policy 2"* gives 1.3% higher accuracy than a CNN with low-magnitude pruning of the weights with 1.7% less energy. The models used for this policy are mostly SVM, Decision Tree, and then CNN with pruning (also available on Figure 5, under the normalized energy level of 79%).

Image Classification

Similar to the previous experiment, for ImageNet classification task, we also compared the solution of optimal policy with several efficient methods developed in recent years. An overview of the energy consumption and performance of several models are plotted on Figure 3. The exact numbers of several important points in the mentioned figure, grouped by similar energy consumption, are shown in Table 2.

Several impressive results can be observed in this experiment. First, *"Optimal Policy 3"* consumes 0.35% less energy while achieving 0.92% accuracy gain, compared to a very efficient neural network such as MobileNetV2. More importantly, this is achieved by using only Resnet34 and ShuffleNetV2-1× models. Second, *"Optimal Policy 6"*, which uses both SE-ResNet50 and SE-ResNet101, can save nearly 20% energy with only 0.34% accuracy drop.

5.2 Effects of Applying the Penalty Term

In this section, we study the effects of applying the penalty term introduced in (1) with the same parameters we reported in the experimental setup section.

Figure 4 visualizes the optimal deployment distribution over different models in our model pool for the activity recognition dataset. In other words, each row i and column j of this figure shows what would the normalized value of x_i be if the normalized budget (B) is set to be j. Although there is a general trend between two models with lower energy and higher accuracy, we can observe that by adding the penalty term, the number of selected models is reduced. We set $\lambda = 0.1K$ in (1) to enforce choosing as few models as possible as visualized in Figure 5.

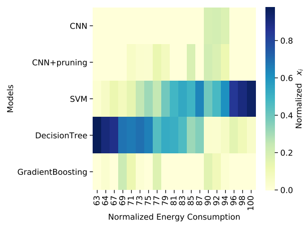

Figure 4: $\lambda = 0$

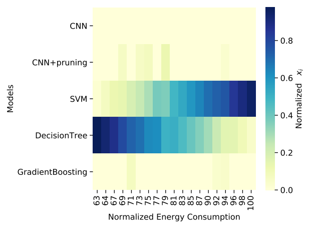

Figure 5: $\lambda = 0.1K$

Another important aspect to discuss is the effect of applying the penalty term on accuracy. Table 3 compares the accuracy of models with and without the penalty term. As we can see, if we do not apply the penalty term, we can get higher accuracy due to the fact that we are allowed to include as many different models as we need in our solution to increase the expected performance. Although the reported energy on the first column only takes into account the energy consumption of inference and not the energy due to loading the models, the accuracy gain is visible if we do not penalize inclusion of multiple models. However, if the number of expected inferences is much less than the number of models, the penalty term can be ignored. The reason is, we load at most n different models which is negligible compared to the energy we need to spend for inference if $n << K$. Nevertheless, for the sake of completeness, we reported the results with the penalty term in the previous section.

Energy Consumption	**Accuracy** ($\lambda = 0.1K$)	**Accuracy** ($\lambda = 0$)
61.07	85.47	85.71
79.51	90.57	90.75
85.65	92.24	92.42
95.90	95.21	95.21

Table 3: Effect of applying penalty term on accuracy

5.3 Empirical Approximation Errors of LP-Relaxation

As discussed in the Linear Relaxation and Approximation Algorithms section, in practice, the solution to the linear programming (LP) relaxation of the E-KKP problem is in an acceptable approximation range. In Table 4 and 5, we compare the accuracy of the solutions to the optimal deployment policy problem for the two methods: integer linear programming (ILP) formulation of (1) (with $\lambda = 0$); and linear programming(LP) relaxation solution with the floor function as the rounding mechanism. We compare these methods for different values of K. The final column shows the results when we apply the LP-relaxation.

We can observe that as the expected number of inferences (K) increases, the LP-relaxation solution converges to the optimal value. Note that the rounding mechanism is unsophisticated and the value of K does not have to be very large as we found even with 1000 of expected inference. Yet, the approximation error of the linear relaxation solution is acceptable. However, in practice, the expected number of inferences we expect a deployed model to perform is significantly higher.

Energy Consumption	**Accuracy**					
	K=10 (ILP)	**K=10 (LP)**	**K = 100 (ILP)**	**K = 100 (LP)**	**K= 1000 (ILP)**	**K = 1000 (LP)**
65	86.68	78.01	86.78	85.92	86.8	86.7
70	88.05	79.1	88.13	87.22	88.15	88.07
75	89.38	61.13	89.49	88.64	89.52	89.43
80	90.82	81.27	90.87	89.94	90.88	90.79
85	92.11	83.44	92.2	91.35	92.24	92.15
90	93.48	84.53	93.6	92.65	93.61	93.52
95	94.68	85.61	94.96	94.06	94.97	94.88
100	96.33	86.7	96.33	95.37	96.33	96.23

Table 4: Empirical Approximation Errors for UCI-HAR

Energy Consumption	**Accuracy**					
	K=10 (ILP)	**K=10 (LP)**	**K=100 (ILP)**	**K=100 (LP)**	**K=1000 (ILP)**	**K=1000 (LP)**
20	70.65	63.71	70.67	69.95	70.68	70.61
30	73.21	66.2	73.27	72.57	73.27	73.2
40	74.85	59.58	75.02	74.29	75.03	74.95
50	76.48	69.06	76.66	75.92	76.68	76.61
60	77.64	69.87	77.7	76.93	77.71	77.63
70	77.86	70.1	77.88	77.1	77.88	77.8
80	78.02	70.25	78.05	77.27	78.05	77.97
90	78.17	70.4	78.22	77.44	78.22	78.15
100	78.39	70.56	78.39	77.61	78.39	78.32

Table 5: Empirical Approximation Errors for Imagenet

6 Discussion and Future Work

We introduced a new research problem at the intersection of machine learning and edge devices, namely how to find an optimal allocation of machine learning models subject to a given energy budget. We showed that this optimization problem is NP-Complete and designed an approach to obtain near-optimal solutions. Our extensive analyses using real-data on two emerging applications, including image and activity recognition tasks, demonstrated the superiority of our approach over the state-of-the-art models.

There are several promising future directions that we plan to pursue. In particular, "Power Modeling" deserves further investigation. We measured the power consumption using an accurate interface, which may initially seem impractical. However, as shown in Table 6, the number of required FLOPS gives a close estimation of the actual energy consumption for several models. Note that the number of required arithmetic operations for other ML models can easily be calculated and is not limited to neural networks. Furthermore, modelling system load and throughput of the system, which can affect the power consumption, can be incorporated into the optimization problem in special applications.

We believe that by showing the effectiveness of the optimal deployment, we can engage others to work on this challenging and unexplored research problem.

Model	**FLOPS**	**Actual Energy Consumption**
resnet18	23.8	18.4
resnet50	54.0	55.0
resnet101	100.8	98.5
se_resnet50	51.1	55.7
se_resnet101	100	100

Table 6: FLOPS of a model gives a close estimation of actual energy consumption.

Acknowledgement

This work was supported in part by the National Science Foundation under grant CNS-1750679. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding organizations.

References

Steven Diamond Akshay Agrawal, Robin Verschueren and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

Parastoo Alinia, Seyed-Iman Mirzadeh, and Hassan Ghasemzadeh. Actilabel: A combinatorial transfer learning framework for activity recognition. *ArXiv*, abs/2003.07415, 2020.

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *ESANN*, 2013.

Alberto Caprara, Hans Kellerer, Ulrich Pferschy, and David Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, 123:333–345, 2000.

Erlangen Regional Computing Center. Likwid: Performance monitoring and benchmarking suite. https://github.com/RRZE-HPC/likwid, 2019. [Online].

Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *ArXiv*, abs/1602.02830, 2016.

Marianne D'Aquila and Michael Shirer. Worldwide Spending on Cognitive and Artificial Intelligence Systems Forecast to Reach $77.6 Billion in 2022, According to New IDC Spending Guide. https://www.idc.com/getdoc.jsp?containerId=prUS44291818, 2019. [Online].

Emily L. Denton, Wojciech Zaremba, Joan Bruna, and et. al. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Alexander Domahidi, Eric Chu, and Stephen P. Boyd. ECOS: an SOCP solver for embedded systems. In *European Control Conference, ECC 2013, Zurich, Switzerland, July 17-19, 2013*, pages 3071–3076, 2013.

Dustin Franklin. Jetson Nano Brings AI Computing to Everyone. https://devblogs.nvidia.com/jetson-nano-ai-computing/, 2019. [Online].

Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *NIPS*, 2015.

Babak Hassibi, David G. Stork, and Gregory J. Wolff. Optimal brain surgeon and general network pruning. *IEEE International Conference on Neural Networks*, pages 293–299 vol.1, 1993.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. 2018.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2016.

Intel. Intel Nervana Neural Network Processors. https://www.intel.ai/nervana-nnp, 2019. [Online].

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ArXiv*, abs/1502.03167, 2015.

Benoit Jacob, Skirmantas Kligys, Bo Chen, and et. al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2017.

Nitthilan Kanappan Jayakodi, Syrine Belakaria, Aryan Deshwal, and Janardhan Rao Doppa. Design and optimization of energy-accuracy tradeoff networks for mobile platforms via pretrained deep models. *ACM Transactions on Embedded Computing Systems (TECS)*, 19:1 – 24, 2020.

Norman P. Jouppi, Cliff Young, Nishant Patil, and et. al. In-datacenter performance analysis of a tensor processing unit. *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, pages 1–12, 2017.

S. Sathiya Keerthi, Olivier Chapelle, and Dennis DeCoste. Building support vector machines with reduced classifier complexity. *J. Mach. Learn. Res.*, 7:1493–1515, 2006.

Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Wenxin Li, Joohyun Lee, and Ness B. Shroff. A faster fptas for knapsack problem with cardinality constraint. *ArXiv*, abs/1902.00919, 2019.

Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: practical guidelines for efficient CNN architecture design. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, pages 122–138, 2018.

Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant: Bridging the gap between student and teacher. *ArXiv*, abs/1902.03393, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, and et. al. Automatic differentiation in pytorch. 2017.

F. Pedregosa, G. Varoquaux, Gramfort, and et. al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *ArXiv*, abs/1802.03268, 2018.

Olga Russakovsky, Jun Deng, Hao Su, Jonathan Krause, and et. al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2014.

Mark Sandler, Andrew G. Howard, Menglong Zhu, and et. al. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4510–4520, 2018.

Michael Shirer and Carrie MacGillivray. The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025, According to a New IDC Forecast. https://www.idc.com/getdoc.jsp?containerId=prUS45213219, 2019. [Online].

Michael Shirer and Carrie Michael Palma. Annual Edge AI Processor Shipments Forecast to Reach 1.5 Billion Units by 2023, According to IDC. https://www.idc.com/getdoc.jsp?containerId=prUS45124019, 2019. [Online].

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, and et. al. Measuring energy and power with papi. In *2012 41st International Conference on Parallel Processing Workshops*, pages 262–268, Sep. 2012.

Michael Zhu and Suyog Gupta. To prune, or not to prune: Exploring the efficacy of pruning for model compression. *ArXiv*, abs/1710.01878, 2017.