Transfer Learning for Design-Space Exploration with High-Level Synthesis

Jihye Kwon
Department of Computer Science
Columbia University, New York, NY, USA
jihyekwon@cs.columbia.edu

Luca P. Carloni
Department of Computer Science
Columbia University, New York, NY, USA
luca@cs.columbia.edu

ABSTRACT

High-level synthesis (HLS) raises the level of design abstraction, expedites the process of hardware design, and enriches the set of final designs by automatically translating a behavioral specification into a hardware implementation. To obtain different implementations, HLS users can apply a variety of knobs, such as loop unrolling or function inlining, to particular code regions of the specification. The applied knob configuration significantly affects the synthesized design's performance and cost, e.g., application latency and area utilization. Hence, HLS users face the design-space exploration (DSE) problem, i.e. determine which knob configurations result in Pareto-optimal implementations in this multi-objective space. Whereas it can be costly in time and resources to run HLS flows with an enormous number of knob configurations, machine learning approaches can be employed to predict the performance and cost. Still, they require a sufficient number of sample HLS runs. To enhance the training performance and reduce the sample complexity, we propose a transfer learning approach that reuses the knowledge obtained from previously explored design spaces in exploring a new target design space. We develop a novel neural network model for mixed-sharing multi-domain transfer learning. Experimental results demonstrate that the proposed model outperforms both single-domain and hard-sharing models in predicting the performance and cost at early stages of HLS-driven DSE.

CCS CONCEPTS

• **Hardware** → **High-level and register-transfer level synthesis**; • **Computing methodologies** → **Transfer learning**.

KEYWORDS

high-level synthesis; design space exploration; neural networks; machine learning; transfer learning; multi-task learning

ACM Reference Format:
Jihye Kwon and Luca P. Carloni. 2020. Transfer Learning for Design-Space Exploration with High-Level Synthesis. In *2020 ACM/IEEE Workshop on Machine Learning for CAD (MLCAD '20), November 16–20, 2020, Virtual Event, Iceland*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3380446.3430636

MLCAD '20, November 16–20, 2020, Virtual Event, Iceland

ACM ISBN 978-1-4503-7519-1/20/11…$15.00
https://doi.org/10.1145/3380446.3430636

Sobel Filter	BFS	Merge Sort
• Block-dim X • Block-dim Y • Compute units • SIMD X; SIMD Y • Tool SIMD • Sub-dim X; Y	• Compute units 1 • Compute units 2 • Enable branch • Mask type • SIMD 2 • Unroll factor	• Compute units • Local sort size • Local pointers • Specialized code • Unroll factor • Work groups; items

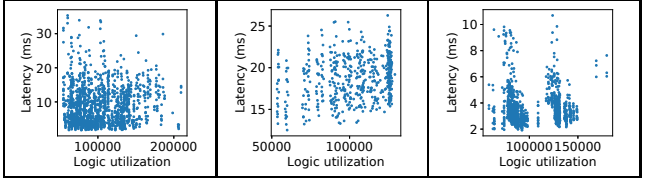


Figure 1: Configurable knobs (top) and design space (bottom) for three applications in Spector: Sobel Filter, BFS (breadth-first search), and Merge Sort [7].

1 INTRODUCTION

Computer-aided design (CAD) approaches for electronic circuits have evolved towards raising the level of abstraction. High-level synthesis (HLS) processes a high-level specification of a design and automatically generates a hardware implementation that realizes the specified behavior [14]. A specification can be written in a high-level language such as C, C++, SystemC [2], and OpenCL [1], and HLS tools automatically translate the given specification into a lower-level description such as a structural register-transfer level (RTL) one. The RTL description can be further transformed into an integrated circuit (IC) layout or a field-programmable gate array (FPGA) hardware configuration file via logic synthesis and physical implementation. HLS flows expedite the circuit design task of hardware engineers, and facilitate the FPGA acceleration of software applications written in high-level languages. Furthermore, one specification or application can be synthesized with different *knob configurations* that direct HLS flows to generate alternative but functionally equivalent implementations.

Most HLS-flow knobs define micro-architectural choices or design options for a specific part of the application's specification that eventually affect the performance and cost, also called quality-of-result (QoR), of the final implementation. For instance, applying the unrolling option to a loop in the specification code generally results in a reduction of computational latency in exchange for an area increase. This prompts the need for performing an extensive design-space exploration (DSE) by running the HLS flows with various knob configurations in order to obtain a final implementation with the desired QoR or a rich set of alternative Pareto-optimal implementations [20]. The top row in Fig. 1 lists the configurable knobs for three applications belonging to the Spector

OpenCL FPGA benchmark suite: Sobel Filter, BFS (breadth-first search), and Merge Sort. The bottom row shows the design space for each application in terms of the latency and logic resource utilization of final implementations obtained with different configurations of those knobs in the top row. This is a result of a comprehensive end-to-end DSE performed by Gautier et al., the authors of the benchmark suite [7].

For a given application, designers often complete a huge number of HLS runs with different knob configurations. They do so to explore the multi-objective design space in search of Pareto-optimal designs that offer interesting trade-off opportunities across QoR metrics such as latency, area (resource utilization), and power. The total number of knob configurations grows exponentially with increasing numbers of knobs. Hence, it can be costly, impractical, or infeasible to run HLS flows with all those configurations. For efficient DSE, a large number of approaches have been proposed [5, 13, 24, 25]. In particular, approaches based on machine learning take the HLS flow as a blackbox and train a QoR prediction model using sampled HLS results [11, 12, 16]. These approaches, however, still require a sufficient number of samples to train the model with high accuracy. This number depends on the application and the set of defined knobs. In the worst cases, however, 20% to 50% of the design space are sampled and synthesized [11, 12, 16].

Contributions. We propose a new approach to enhance the training performance and reduce the sample complexity. Our idea is to extract the knowledge obtained from the previously explored spaces of *source* designs and apply it to the early stage of DSE of a new *target* design. Taking the three applications in Fig. 1, for instance, we investigate the following problem: *Is it possible to learn an underlying pattern from DSE results of Sobel Filter that can be effectively applied to Merge Sort?*

The properties and constraints of HLS-driven DSE pose three critical challenges. First, the *domain* (input feature space) of the learning task varies across applications. A QoR prediction model takes as input a knob configuration and outputs predicted QoR values. However, distinct applications often have different numbers and types of knobs available. In Fig. 1, Sobel Filter has 8 knobs, while Merge Sort has 7, and many knobs are different between the two applications. Second, obtained knowledge for one application is expected to include application-specific information. Since Sobel Filter and Merge Sort have different levels and structures of parallelism, applying some common knobs may have different impacts on their QoR. Third, the target application may contain properties that have not been observed in other applications. Hence, the obtained knowledge from source applications may be insufficient to fully describe and predict HLS results of the target application.

In this paper, we address each of the above three challenges as follows. First, we adopt and extend the notion of *transfer learning*. Unlike traditional machine learning (ML), transfer learning aims to improve the performance of a *target* learning task by reusing the knowledge obtained for different but related *source* tasks or domains. For example, artificial neural network (NN) models trained for natural image classification have been transferred to aid disease classification in the medical image domain [21]. In that case, the final classifier layer of the pre-trained NN was freshly trained for the target task. Similarly, we transfer an NN model trained for Sobel Filter to the domain of Merge Sort. In our case, since the input layers are domain-specific, the first hidden layer is freshly trained for the target task.

Second, we propose *multi-domain* transfer learning where a common model is trained from multiple source domains. For instance, if DSE results for BFS are available together with those for Sobel Filter, we can pre-train a NN model that has two versions of the first hidden layer, one with 8 elements for Sobel Filter, and the other with 6 elements for BFS. This is in an attempt to extract effectively common knowledge between multiple source applications that is expected to be shared also with the target application.

Third, we propose a novel NN model for *mixed-sharing* multi-domain transfer learning. In multi-task learning and transfer learning, *hard parameter sharing* refers to the sharing of a common model by multiple tasks [19]. In *soft parameter sharing*, each task has its own model and parameters, where some parameter values are shared across all tasks [4]. Our transfer learning tasks intrinsically exploit hard sharing, and we connect the shared model with an auxiliary and independent sub-model for each task. We introduce this mixed-sharing approach to reflect the diversity of various source and target applications.

To the best of our knowledge, this is the first paper proposing transfer learning approaches for HLS QoR prediction where different applications have different knobs or synthesis options. We present and evaluate two single-domain models and two transfer learning models. The experimental results using the Spector dataset demonstrate that the proposed mixed-sharing multi-domain transfer learning model outperforms the single-domain and hard-sharing models in the early stages of DSE.

2 PROBLEM DESCRIPTION

Let X denote the set of all configurations x of n HLS knobs defined for a given application:

$$x = (x^1, \cdots, x^n) \in X \subset \mathbb{R}^n \tag{1}$$

Let QoR be a function that maps a knob configuration to the QoR values obtained by applying the target HLS flows:

$$QoR : X \rightarrow \mathbb{R}^m \tag{2}$$

$$QoR(x) = (QoR^1(x), \cdots, QoR^m(x)) \tag{3}$$

where m denotes the number of QoR metrics of interest. As shown in Fig. 1, the ranges of QoR values vary depending on the application. Let $Train$ denote the set of knob configurations $x \in X$ with known QoR values, and $Test$ the set of those with unknown QoR values. A normalized QoR function $\widetilde{QoR}_{Train}$ returns the QoR values for $x \in Train$ normalized to a fixed interval, e.g., $[0, 1]^m \subset \mathbb{R}^m$, where the normalization is performed separately for each QoR metric. When this function is applied to $x \in Test$, some of the resulting $\widetilde{QoR}_{Train}(x)$ values may be out of the original normalization interval. When $Train$ is obvious, we omit it from the notation $\widetilde{QoR}_{Train}$. The target problem for a given application is to learn its $\widetilde{QoR}$ function, given a $Train$ set labeled with the observed QoR values:

Problem 1. *Given $Train \subset X$, find a prediction function $F : X \rightarrow \mathbb{R}^m$ that approximates $\widetilde{QoR}$:*

$$\arg\min_F \sum_{x \in X} ||F(x) - \widetilde{QoR}(x)||^2 \tag{4}$$

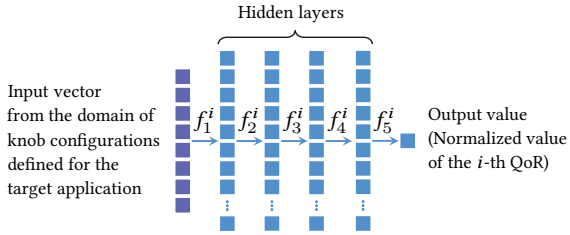


Figure 2: A fully-connected NN F^i as a predictive model for $\widetilde{QoR}^i$ $(1 \leq i \leq m)$ in single-task learning. The input and output layers represent the knob configuration and the predicted QoR value, respectively.

3 PROPOSED QOR PREDICTION MODELS

The goal of classification and regression ML tasks is to learn a label-prediction function, given an input *domain* (a set of knob configurations), output label space (the range of normalized QoR values), and training data (sampled DSE results). Recently, NN models have been extensively used in ML tasks to represent the prediction functions. Inspired by the human brain, a NN consists of processing layers that contain computation units called nodes or neurons [15]. By the universal approximation theorem, any continuous function on a compact set can be approximated by a fully-connected NN [3, 9]. Below, we present a series of fully-connected NN models for HLS QoR prediction in the order of evolution towards the proposed mixed-sharing multi-domain transfer learning.

Single-Task Learning. Fig. 2 illustrates a fully-connected NN model $F^i : X \rightarrow \mathbb{R}$ for learning the unknown target function $\widetilde{QoR}^i$. A NN consists of layers, and each layer contains nodes, depicted as small solid squares in the figure. Each node represents a single value, and the width of a layer is the number of nodes contained in it. The input layer represents a knob configuration x from the domain X. The j-th layer ℓ_j^i is computed with a function f_j^i that takes the previous layer as input and outputs a vector of dimension equal to the width of ℓ_j^i. Layer functions f_j^i are defined as follows:

$$f_j^i(y) = Activation(A_j^i y + b_j^i) \tag{5}$$

where A_j^i is a weight matrix, b_j^i is a bias vector (of appropriate dimensions), and $Activation$ denotes a nonlinear activation function such as hyperbolic tangent, sigmoid, rectified linear unit, or softmax. A network function F^i with 5 layers (as shown in Fig. 2) is a composition of the 5 layer functions f_j^i $(1 \leq j \leq 5)$:

$$F^i(x) = (f_5^i \circ f_4^i \circ f_3^i \circ f_2^i \circ f_1^i)(x) = f_5^i(f_4^i(f_3^i(f_2^i(f_1^i(x))))) \tag{6}$$

The objective of the learning is to minimize the difference between the predicted QoR $F^i(x)$ and the actual QoR value $\widetilde{QoR}^i(x)$ for all knob settings x. Given a configuration set $Train$, Problem 1 can be re-written as follows:

$$\underset{F^i=(A_1^i,b_1^i,\cdots,A_5^i,b_5^i)}{\arg\min} \sum_{x \in Train} \| F^i(x) - \widetilde{QoR}^i(x) \|^2 \tag{7}$$

Stochastic gradient descent and its variants with optimization techniques have been demonstrated to solve effectively the above problem [8]. The L1 norm and squared L2 norm of the weights A_j^i multiplied by a small coefficient are often added to the objective function in Eq. (7) as regularization terms.

Multi-Task Learning. Fig. 3 (a) shows one larger NN representing a multi-task model for all QoR metrics of interest. Instead of learning one model for each metric as in Fig. 2, multi-task learning attempts to improve the performance of multiple learning tasks by using some common knowledge contained in those tasks [23]. The function G computed by the proposed multi-task NN for m QoR metrics is defined as follows:

$$G(x) = (G^1(x), \cdots, G^m(x)) \tag{8}$$

$$G^i(x) = (f_5^i \circ f_4^i \circ g_3 \circ g_2 \circ g_1)(x) = f_5^i(f_4^i(g_3(g_2(g_1(x))))) \tag{9}$$

where g_1, g_2, and g_3 are nonlinear functions for the first three hidden layers. The input for g_1 is a knob setting $x \in X$ for the target application. The output of g_3 represents the common feature that is used as an input to $f_5^i \circ f_4^i$ for each task i.

From our experiments, when the size of $Train$ is large enough, single-task models can approximate the target function with higher accuracy on $Test$ than multi-task ones, but when $Train$ is relatively small, multi-task models outperform single-task ones. We mainly explore multi-task models in the next subsections.

Cross-Domain Transfer Learning. The QoR prediction model in Fig. 3 (a) learned for one application can be transferred to a different target domain, as shown in Fig. 3 (b). More precisely, *a part of the NN learned for the source task is transferred to be a part of another NN for the target task.* The shaded box in Fig. 3 (b) corresponds to the transferred part of the NN from Fig. 3 (a). Given a pre-trained multi-task model G, the proposed transfer learning model H is defined as follows:

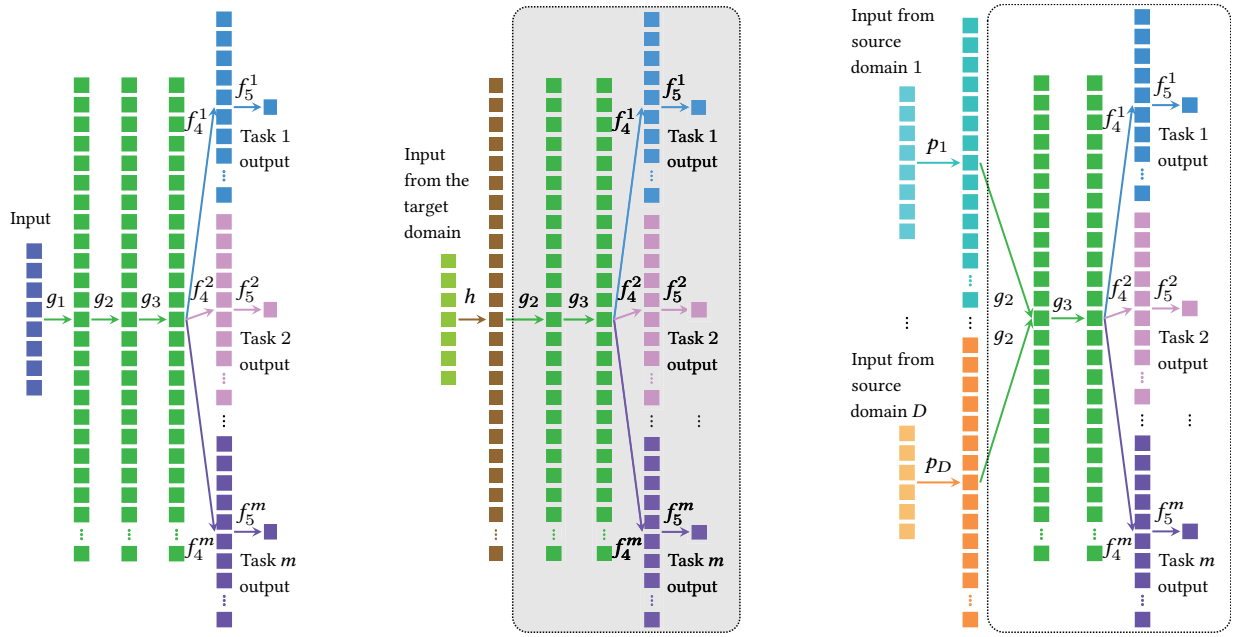
$$H(x) = (H^1(x), \cdots, H^m(x)) \tag{10}$$

$$H^i(x) = (f_5^i \circ f_4^i \circ g_3 \circ g_2 \circ h)(x) = f_5^i(f_4^i(g_3(g_2(h(x))))) \tag{11}$$

where h denotes the new nonlinear function in the first hidden layer (the brown layer in Fig. 3 (b)). Pre-trained functions g_2, g_3, f_4^i, and f_5^i (in the shaded box in Fig. 3 (b)) are transferred from G. The leftover part in G is its first hidden layer g_1. The domain of g_1 is the knob configuration space of the synthesized application that G has been trained for. The codomain of g_1 is a vector space of a fixed dimension, which is the domain of g_2. The new function h is defined with the domain of the knob configuration space for the new target application and its codomain is equal to that of g_1. Since this codomain coincides with the domain of g_2, the function composition of $g_2 \circ h$ is well-defined in the new model H. The transferred part determines how to compute the predicted QoR values from the intermediate feature and it remains fixed during training for the new target task. Only the new function h from the target domain to the intermediate feature space is learned, possibly with much smaller samples. The optimization problem can be expressed as follows:

$$\underset{h=(A_h,b_h)}{\arg\min} \sum_{x \in Train} \| H(x) - \widetilde{QoR}(x) \|^2 \tag{12}$$

Multi-Domain Transfer Learning. Transfer learning aims to improve the performance of the target learning task by discovering and transferring latent or hidden knowledge from the source domain and task [22]. One of the main research topics in transfer learning is "what to transfer." Some knowledge is specific for individual domains or tasks, whereas some other common knowledge can be transferred to help improve the performance of the target



(a) A single-domain multi-task model. A common sub-network (green layers) is connected to the task-specific ones (sky-blue, pink, and purple layers).

(b) A cross-domain transfer learning model using the sub-network (in the shaded box). For the target task, only the first hidden layer (the brown layer) is freshly trained.

(c) A multi-domain model for transfer learning. This model is trained on multiple source domains. Then, the trained sub-network (in the dotted box) is transferred (to the shaded box in (b)).

Figure 3: A single-domain and two multi-domain multi-task models for transfer learning.

task [18]. Thus, it is critical to identify such common knowledge, which is often latent. In an attempt to more effectively disentangle common latent knowledge from what is specific for the source domain and task, we propose a multi-domain transfer learning model that interacts with multiple source domains.

Fig. 3 (c) illustrates the proposed multi-domain multi-task model for transfer learning. In this model, we assume that HLS results for multiple source applications are available for pre-training. For D source applications with heterogeneous knob configuration spaces X_d ($1 \le d \le D$), the proposed multi-domain QoR prediction model P is defined as follows:

$$P(d, x_d) = (P^1(d, x_d), \cdots, P^m(d, x_d)),\ 1 \le d \le D,\ x_d \in X_d \quad (13)$$

$$\begin{aligned} P^i(d, x_d) &= (f_5^i \circ f_4^i \circ g_3 \circ g_2 \circ p_d)(x_d) \\ &= f_5^i(f_4^i(g_3(g_2(p_d(x_d))))),\ x_d \in X_d \end{aligned} \quad (14)$$

where p_d represents the individual feature function for application d. The domain of each p_d is the knob configuration space for application d, but their codomain is the common feature vector space with a fixed dimension, which is also the domain of the next layer function g_2. In this model, p_d functions for $d \neq 1$ have some similarities with h in the cross-domain transfer learning model H (in Fig. 3 (b)). However, whereas h and its inputs are not used for the pre-training of the rest of H (which has been transferred from G), all p_d functions and their inputs are used for training the entire model P. By exploring multiple source domains and individual feature functions simultaneously, the shared part of the model $f_5^i \circ f_4^i \circ g_3 \circ g_2$ (in the dotted box in Fig. 3 (c)) is expected to extract common latent knowledge while individual p_d functions are expected to interpret knowledge specific to each application. Then, the shared part can be transferred to a new model for the target application, as in Fig. 3 (b).

Mixed-Sharing Multi-Domain Transfer Learning. To further separate common knowledge from possibly application-specific and domain-specific knowledge, we present another multi-domain multi-task model, illustrated in Fig. 4, that reconciles two types of parameter sharing: hard and soft. The models in Fig. 3 employ hard parameter sharing, where individual active networks for different applications share a common sub-network (the dotted box in (c) and the shaded box in (b)). In soft parameter sharing, each task has its own network and parameters where some parameter values are shared (the values are either identical or very similar) across those tasks [4, 19]. The proposed model Q incorporates both hard parameter sharing and a variant of soft parameter sharing:

$$Q(d, x_d) = (Q^1(d, x_d), \cdots, Q^m(d, x_d)),\ x_d \in X_d \quad (15)$$

$$\begin{aligned} Q^i(d, x_d) &= \alpha_d (f_5^i \circ f_4^i \circ g_3 \circ g_2 \circ p_d)(x_d) + q_d(x_d) \\ &= \alpha_d \cdot f_5^i(f_4^i(g_3(g_2(p_d(x_d))))) + q_d(x_d) \\ &(1 \le d \le D,\ x_d \in X_d,\ 1 \le i \le m, \alpha_d \in \mathbb{R}) \end{aligned} \quad (16)$$

Here, q_d represents an auxiliary function for application d (the light purple sub-network on the top for application 1 and the light orange sub-network at the bottom for application D in Fig. 4). These auxiliary functions are expected to capture the application-specific knowledge in a stronger sense than p_d functions and to separate it from the shared sub-network (in the dotted box in Fig. 4). A learnable parameter α_d determines the ratio of the contribution of the shared function to the final output (predicted QoR). After the pre-training using multiple source applications, only the shared sub-network $f_5^i \circ f_4^i \circ g_3 \circ g_2$ is transferred to a new model for

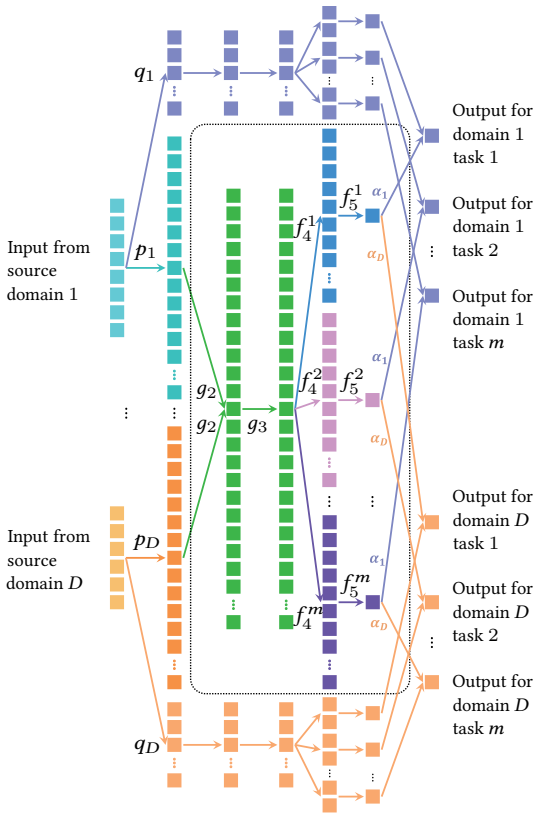


Figure 4: A mixed-sharing multi-domain model for transfer learning with both hard and soft parameter sharing. The network in the dotted box is transferred to the target domain.

the target application. This new model also has its application-specific auxiliary part, in addition to its feature function, that will be freshly trained for the target application while the transferred sub-network parameters are fixed.

4 EXPERIMENTAL RESULTS

To evaluate the performance of the proposed models, we used the set of comprehensive DSE results from the Spector benchmark suite repository [7] containing the nine applications reported in Table 1. The authors of the benchmark suite synthesized the applications using the Altera OpenCL SDK v14.1, and executed successfully synthesized designs on a Terasic DE5 board with a Stratix V FPGA. The QoR metrics of interest are the latency and utilization of the four types of resources: logic, RAM, on-chip memory, and DSP. For each of **`bfs`** and **`spmv`**, there are two sets of QoR obtained with different input datasets. We trained the following predictive models.

Single-task single-domain: model F^i in Fig. 2 for $i = 1, \cdots, 5$. Each F^i has 4 hidden layers with 25 nodes per hidden-layer.

Multi-task single-domain: model G in Fig. 3 (a) with 4 hidden layers. Each hidden layer contains 125 nodes.

Multi-domain transfer: model P in Fig. 3 (c) for the pre-training in the source domains, and model H in 3 (b) for the transfer learning in the target domain. Model P has 1 domain-specific hidden layer with 10 nodes per domain, and 3 shared hidden layers each 125 nodes each. Model H consists of 1 domain-specific hidden layer with 10 nodes, and the transferred sub-network from P.

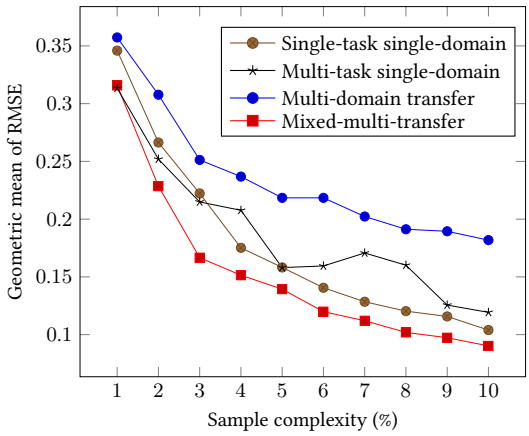


Figure 5: Geometric mean of RMSE across the 11 sets of HLS QoR, achieved by each of the 4 predictive models.

Mixed-multi-transfer: mixed sharing multi-domain model Q in Fig. 4 for the pre-training, and a variation of model H with an auxiliary network for the transfer learning. Q has 1 domain-specific hidden layer containing 10 nodes, one shared sub-network with 3 hidden layers each containing 100 nodes, and a domain-specific auxiliary network that has 4 hidden layers with 25 nodes per hidden-layer for each source domain.

The structure of the single-task model was determined empirically based on the observation that as the numbers of layers and nodes increase, the model's prediction accuracy first increases and then decreases; the structure with the peak accuracy was selected. Other models were defined to have the same number of hidden layers and nodes with the single-task ones. For two transfer learning models, all applications except the target one were used as the source applications. The models were trained using the PyTorch library with the iterations of 100,000 and the learning rate of 0.001 for P and Q, and of 0.01 for all other NNs. For each application, pre-training with P or Q took between 100 and 160 minutes using Intel i7-6700K CPU running Ubuntu 16.04.6 LTS. The target predictive models took between 10 and 30 minutes, depending on the sample complexity. Fig. 5 shows the geometric mean of the *RMSE (root mean squared error)* of the four predictive models across all target applications for each of the sample size $1\%, 2\%, \cdots, 10\%$ of the design space [8]. Those RMSE values were computed after normalizing the entire DSE results X, whereas for training, $Train$ sets were independently normalized. With small sample complexity from 1% to 3%, the multi-task single-domain model achieves lower error rates, and thus higher accuracy, than the single-task model. For all sample complexities tested, the mixed-sharing multi-domain transfer learning model achieves the lowest error rates.

Using the Spector dataset, we simulated the DSE by reading the reported QoR values instead of running HLS flows to train and test the single-task, multi-task, and mixed-multi-transfer learning models. As DSE objectives, we used the latency and logic utilization; it is also possible to perform a multi-dimensional DSE with more metrics. We used the models trained with 3% of the target design space, and selected 3% Pareto-optimal and near-optimal configurations from the predicted QoR. Fig. 6 shows the *ADRS (average distance from reference set)* of the proposed configurations by each model with respect to the golden Pareto-optimal set [20].

Table 1: Design spaces of the 9 applications in the Spector dataset. Each of bfs and spmv has 2 versions of the QoR results.

Application	bfs_dense	bfs_sparse	dct	fir	hist	mergesort	mm	normals	sobel	spmv_5000	spmv_500000
# Knobs	6	6	9	8	7	7	9	7	8	4	4
# Designs	507	507	211	1173	896	1532	1180	696	1381	740	740

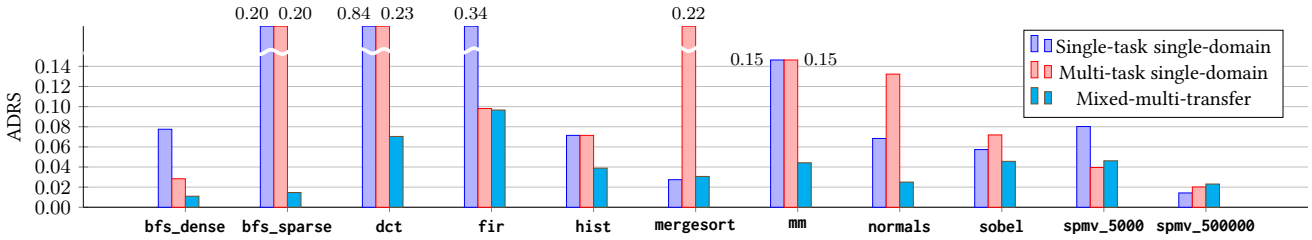


Figure 6: Simulated DSE results for 9 Spector applications with 3%-sample and 3%-proposed configurations.

The mixed-multi-transfer model achieves the lowest ADRS in most cases. The geometric mean of its ADRS for all applications is 0.034.

5 RELATED WORK

Design Space Exploration. A recent work by Schafer et al. classifies the HLS-driven DSE techniques into three categories: 1) synthesis based, 2) model based, and 3) supervised learning, which is a mix of the first two [20]. The synthesis-based approaches observe the QoR of synthesized designs and select knob configurations for the next iterations of HLS flows. To generate configurations, meta-heuristic methods such as genetic algorithms have been exploited [5, 10]. Mahapatra et al. perform simulated annealing and use a decision tree to reduce the search space [13]. The analytical model-based approaches construct a predictive model, instead of invoking HLS flows, based on the information about applications, HLS tools, and hardware platforms [24, 25]. The supervised-learning approaches sample the design space (synthesis-based) and train predictive models (model-based). Liu et al. perform active learning for sampling, and train a random forest to predict the QoR [12]. Mahapatra et al. execute simulated annealing, and use a decision tree to reduce the search space [13]. Meng et al. eliminate the design points that are likely to be Pareto-dominated [16]. Ferretti et al. propose a lattice-traversing based exploration approach [6]. Lin et al. exploit four classes of machine learning methods to estimate the power, and NN models has shown the best performance [11]. O'Neal et al. predict the QoR on FPGA given the profiled information of the target application running on CPUs [17]. With all those approaches, the results from previous applications are not exploited during the DSE of a new application.

Transfer Learning. Transfer learning allows the domains, tasks, and distributions used in pre-training and testing to be different but related. The transfer-learning settings and methods presented in this paper can be considered as examples of inductive transfer learning [18] as well as network-based deep transfer learning [22].

6 CONCLUSION

We propose a transfer learning approach to aid HLS-driven DSE when the target application's design space is huge and DSE results from other source applications are available. Successfully trained predictive models can approximate QoR without invoking HLS tools for an exhaustive DSE. As demonstrated using the Spector dataset, the mixed-sharing multi-domain model outperforms the other models in QoR prediction and in the simulated DSE.

Acknowledgments. This work is partially supported by the NSF (A#: 1527821 and 1764000).

REFERENCES

[1] [n.d.]. OpenCL, The Khronos Group. https://www.khronos.org/opencl/.
[2] [n.d.]. SystemC, Accellera Systems Initiative. https://accellera.org/community/systemc.
[3] G. Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Math. of control, signals and systems* 2, 4 (1989), 303–314.
[4] L. Duong et al. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *ACL-IJCNLP*. 845–850.
[5] F. Ferrandi et al. 2008. A multi-objective genetic algorithm for design space exploration in high-level synthesis. In *Symposium on VLSI*. 417–422.
[6] L. Ferretti et al. 2018. Lattice-traversing design space exploration for high level synthesis. In *International Conf. on Computer Design*. 210–217.
[7] Q. Gautier et al. 2016. Spector: An OpenCL FPGA benchmark suite. In *Intl. Conf. on Field-Programmable Technology*. 141–148.
[8] I. Goodfellow et al. 2016. *Deep Learning*. MIT Press.
[9] K. Hornik et al. 1989. Multilayer feedforward networks are universal approximators. *Neural Networks* 2, 5 (1989), 359–366.
[10] V. Krishnan and S. Katkoori. 2006. A genetic algorithm for the design space exploration of datapaths during high-level synthesis. *IEEE Transactions on Evolutionary Computation* 10, 3 (2006), 213–229.
[11] Z. Lin et al. 2020. HL-Pow: A learning-based power modeling framework for high-level synthesis. In *Asia and South Pacific Design Automation Conf.*
[12] H. Liu and L. Carloni. 2013. On learning-based methods for design-space exploration with high-level synthesis. In *Design Automation Conf.* 50.
[13] A. Mahapatra and B. Schafer. 2014. Machine-learning based simulated annealer method for high level synthesis design space exploration. In *Electronic System Level Synthesis Conf.*
[14] G. Martin and G. Smith. 2009. High-level synthesis: Past, present, and future. *IEEE Design Test of Computers* 26, 4 (July 2009), 18–25.
[15] W. S McCulloch and W. Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics* 5, 4 (1943), 115–133.
[16] P. Meng et al. 2016. Adaptive threshold non-Pareto elimination: Re-thinking machine learning for system level design space exploration on FPGAs. In *Design, Automation & Test in Europe*. 918–923.
[17] K. O'Neal et al. 2018. HLSPredict: Cross platform performance prediction for FPGA high-level synthesis. In *International Conf. on Computer-Aided Design*.
[18] S. Pan et al. 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2010), 1345–1359.
[19] S. Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv:1706.05098* (2017).
[20] B. Schafer and Z. Wang. 2019. High-level synthesis design space exploration: Past, present and future. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2019).
[21] H. Shin et al. 2016. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging* 35, 5 (2016), 1285–1298.
[22] C. Tan et al. 2018. A survey on deep transfer learning. In *International Conf. on Artificial Neural Networks*. 270–279.
[23] Y. Zhang et al. 2017. A survey on multi-task learning. *arXiv:1707.08114* (2017).
[24] J. Zhao et al. 2017. COMBA: A comprehensive model-based analysis framework for high level synthesis of real applications. In *International Conf. on Computer-Aided Design*. 430–437.
[25] G. Zhong et al. 2016. Lin-analyzer: A high-level performance analysis tool for FPGA-based accelerators. In *Design Automation Conf.*