M^3ViT: Mixture-of-Experts Vision Transformer for Efficient Multi-task Learning with Model-Accelerator Co-design

Anonymous Author(s)
Affiliation
Address
email

Abstract

Multi-task learning (MTL) encapsulates multiple learned tasks in a single model and often lets those tasks learn better jointly. Multi-tasking models have become successful and often essential for many sophisticated systems such as autonomous driving and indoor robots. However, when deploying MTL onto those real-world systems that are often resource-constrained or latency-sensitive, two prominent challenges arise: (i) during **training**, simultaneously optimizing all tasks is often difficult due to gradient conflicts across tasks, and the challenge is amplified when a growing number of tasks have to be squeezed into one compact model; (ii) at **inference**, current MTL regimes have to activate nearly the entire model even to just execute a single task. Yet most real systems demand only one or two tasks at each moment, while flexibly switching between tasks per need: therefore such "all tasks activated" inference is also highly inefficient and non-scalable in practice.

In this paper, we present a model-accelerator **co-design** framework to enable efficient on-device MTL, that tackles **both** training and inference bottlenecks. Our framework, dubbed M^3ViT, customizes mixture-of-experts (MoE) layers into a vision transformer (ViT) backbone for MTL, and sparsely activates task-specific experts during training, which effectively disentangles the parameter spaces to avoid different tasks' training conflicts. Then at inference with any task of interest, the same design allows for activating only the task-corresponding sparse "expert" pathway, instead of the full model. Our new model design is further enhanced by hardware-level innovations, in particular, a novel computation reordering scheme tailored for memory-constrained MTL that achieves zero-overhead switching between tasks and can scale to any number of experts. Extensive experiments on PASCAL-Context [1] and NYUD-v2 [2] datasets at both software and hardware levels are conducted to demonstrate the effectiveness of the proposed design. When executing the practical scenario of single-task inference, M^3ViT achieves higher accuracies than encoder-focused MTL methods, while significantly reducing **88%** inference FLOPs. When implemented on a hardware platform of one Xilinx ZCU104 FPGA, our co-design framework reduces the memory requirement by **2.40**$\times$, while achieving energy efficiency (as the product of latency and power) up to **9.23**$\times$ higher than a comparable FPGA baseline. Code will be released.

1 Introduction

Vision Transformers (ViTs) [3, 4, 5, 6], as the latest performant deep models, have achieved impressive performance on various computer vision tasks [7, 8, 9]. These models are specially trained or tested for only one or a few tasks; however, many real-world applications require one compact system

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

that can *handle many different tasks* efficiently, and often need to swiftly *switch between tasks* per demand. For example, an autonomous driving system [10] needs to perform and switch between many tasks such as drivable area estimation, lane detection, pedestrian detection, and scene classification: apparently both single task inference and cross-task switching need to happen at ultra-low latency. As another example, smart-home indoor robots [11] are expected to address semantic segmentation, navigation, tracking or other tasks in varying contexts, with very limited on-board resources. Multi-task learning (MTL) [12, 13, 14] solves multiple tasks simultaneously within a single model and learns improved feature representations [15] shared by related tasks [16, 17]. Therefore, accomplishing **realistic efficient MTL** is becoming a key knob for building real-time sophisticated AI systems.
Despite the promise, challenges persist to build an efficient MTL model suitable for real world applications: ❶ during **training**, prior works [18, 19, 20] indicate the competition of different tasks in training may degrade MTL, since the same weights might receive and be confused by conflicting update directions. Specifically, [19] reveals that negative cosine similarities between different tasks' gradients are detrimental. [21, 22] confirm that conflicting gradients not only slow down convergence but also bias the learned representations against some tasks. That is only getting worse on compact models owing to their limited modeling capacity. To tackle the cross-task conflicts, solutions have been proposed by varying learning rate speeds of different tasks [20], using "cross-stitch" sharing [23], or re-balancing task gradients [19, 24, 20, 25]. However, they either require task-specific design or significantly increase the model complexity which contradicts our efficiency goal. ❷ at **inference**, existing MTL regimes typically activate the entire backbone model unconditionally. However, **many real systems only need to call upon one or a few tasks at each moment**, hence the "all activated" inference is heavily inefficient and non-scalable. For example, current regimes [14, 23, 26, 27] have to activate the whole gigantic ResNet [28] encoder even just to execute a single monocular depth estimation task or so. If the number of tasks scale up [29] and the backbone keeps growing bigger, the "per task" inference efficiency of the resultant MTL model could become catastrophically poor.

To tackle these bottlenecks, we propose a model-accelerator **co-design** framework that enables efficient on-device MTL. Specifically, <u>in the software level</u>, we propose to adapt mixture of experts (MoE) layers [30, 31] into the MTL backbone, as MoE can adaptively divide-and-conquer the entire model capacity into smaller sub-models [30, 32]. Here, we replace the dense feed-forward-network in the ViT with sparsely activated MoE experts (MLPs). A task-dependent gating network will be trained to select the subset of experts for each input token, conditioning on tasks. During training, this task-dependent routing principle effectively disentangles the parameter spaces, balancing feature reuse and automatically avoiding different tasks' training conflicts. Meanwhile, at the inference stage with any task of interest, this design naturally allows for sparse activation of only the experts corresponding to the task instead of the full model, thus achieving highly sparse and efficient inference for the specific task. <u>In the hardware level</u>, we propose a novel computation reordering mechanism tailored for memory-constrained MTL and MoE, which allows scaling up to any number of experts and also achieves zero-overhead switching between tasks. Specifically, based on ViT, we push tokens to per-expert queues to enable expert-by-expert computation rather than token-by-token. We then implement a double-buffered computation strategy that hides the memory access latency required to load each experts' weights from off-chip memory, regardless of task-specific expert selection. This design naturally incurs no overhead for switching between frames or tasks in FPGA.

To validate the effectiveness, we evaluate our performance gain using the ViT-small backbone on the NYUD-v2 and PASCAL-Context datasets. On the NYUD-v2 dataset with two tasks, our model achieves comparable results with encoder-focused MTL methods while reducing 71% FLOPs for single-task execution. When we evaluate on the PASCAL-Context dataset with more tasks, our model achieves even better performance (2.68 *vs.* 0.60) and reduces 88% inference FLOPs. We found the MTL performance gain brought by MoE layers consistently increases as the task count grows. When implemented on a hardware platform of one Xilinx ZCU104 FPGA, our co-design framework reduces the memory requirement by 2.40× while achieving energy efficiency (as the product of latency and power) up to 9.23× higher than comparable FPGA baselines and up to 10.79× higher than the GPU implementation. Our contributions are outlined below:

- We target the problem of efficient MTL, and adopt the <u>more realistic inference setting</u> (activating one task a time, while switching between tasks). We introduce MoE as the unified tool to attain two goals: both resolving cross-task training conflicts (*better MTL performance*), and sparsely activating paths for single-task inference (*better efficiency*). Specifically for MTL, the MoE layer is accompanied with a *task-dependent* gating network to make expert selections conditioning on the current task.

- We implement the proposed MTL MoE ViT framework on a hardware platform of one Xilinx ZCU104 FPGA, which enables us to exploit a memory-efficient computation reordering scheme that consolidates per-expert Multiply-and-ACcumulate (MAC) operations such that only one expert's weights are needed on-chip at a time. Our design is scalable to any number of experts while requiring no frame-switching or task-switching overhead.
- We conduct extensive experiments to justify its inference effectiveness in both accuracy and on-edge efficiency metrics. Our framework, dubbed M^3ViT, achieves higher accuracies than encoder-focused MTL methods, while significantly reducing **88%** inference FLOPs; on hardware, it reduces the memory requirement by **2.40**× and costs up to **9.23**× and **10.79**× less energy compared to the FPGA and GPU baselines, respectively.

2 Related Works

Multi-task Learning The generic multi-task learning problem has been studied for a long history. Some non-deep learning-based methods propose to use distance metric [33, 34, 35], probabilistic prior [36, 37, 38, 39, 40] to model the common information among tasks. With the emergence of the deep learning technique, MTL is performed to learn shared representation among tasks. The emerging of ViT further makes it possible to extent the task range from only vision tasks to other modalities tasks (e.g., text, audio) [41, 42, 43, 44, 45]. Current MTL models can be roughly categorized into two types based on where the task interactions take place in the network. The *encoder-focused* architectures [23, 46, 26, 27] only share information in the encoder, before decoding each task with an independent task-specific head. Cross-stitch networks [23] introduce linear combination in each layer. NDDR-CNN [26] improves it by dimensional reduction. MTAN [27] leverages an attention mechanism to learn between tasks. The second type, *decoder-focused* models [47, 48, 49, 50], make initial task predictions in decoder and then leverage features from these initial predictions to further improve output. Although they report higher performance, their models consume a large number of FLOPs [14]. This makes it difficult to deploy them onto those real-world systems that are often resource-constrained or latency-sensitive. And they need to execute all the tasks for initial prediction, which is heavily inefficient in the common scenario when only one or few tasks are needed. Hence, we focus on encoder-focused architecture in this work. Moreover, some other methods [25, 20, 51, 27] are proposed to handle the multi-task training conflicts problem.

Mixture of Experts (MoE) MoE contains a series of sub-models (i.e., experts) and performs conditional computation in an input-dependent fashion [52, 53, 54, 55, 56], based on learned or deterministic routing policies [57, 56]. The traditional dense MoEs suffer from intensive computational costs since they select all experts [58]. Recent studies [30, 59, 60] in natural language processing (NLP) propose sparse MoE that sparsely activates a few experts during both training and inference, thus substantially reducing the cost and allowing gigantic language models even with trillions of parameters [60]. Unfortunately, such a sparse-gated manner still has limitations of unstable training and imbalanced selections among experts. Various solutions are invented from regularization [61, 59, 60] and optimization [62, 63] perspectives. Moreover, MoE has drawn increasing popularity in computer vision [58, 64, 65, 66, 67, 68, 69], where it mainly focuses on considerably smaller network backbones compared to the ones in NLP. For instance, [66] and [67] formulate the channel and kernel of convolutional layers as experts and establish the MoE framework. Several pioneer investigations also explore MoE for multi-task learning, which are related to this work. Particularly, [17, 70, 71] introduce task-specific gating networks to choose different parts of models for processing information from each task. They present certain possibilities of using MoE to solve MTL problems in some cases like classification for medical signals [70], digital number images (MNIST) [71], and recommendation systems [17]. We make a further attempt to adapt MoE into a compact model for dense prediction multi-task learning, along with software-hardware co-design.

Hardware FPGA acceleration of Transformer-based models has attracted increasing attention. Works such as VAQF [72], SPViT [73], FTRANS [74], Qi *et al.* [75], and Peng *et al.* [76] note that Transformer models are computation- and memory-intensive and are too large to fit on the FPGA. Therefore, various model compression methods have been proposed, such as activation quantization, token pruning, block-circulant matrices (BCM) for weights, block-balanced weight pruning, and column-balanced block weight pruning. Such compression methods are lossy and require compression-aware training to regain accuracy. To the best of our knowledge, there is *no existing FPGA accelerator for MoE* in a Transformer-based model. The MoE mechanism exposes great challenges to FPGA since it requires swift expert switching between tokens and frames, which

may introduce significant overhead of memory and parameter loading. In this work, however, we propose a novel expert-by-expert computation-reordering approach that can reduce the overhead to negligible despite the number of experts, and does not require model compression or re-training.

3 Method

Overview We first describe the standard Vision Transformer and MoEs, and then show the proposed MoE ViT design for MTL. To enable dynamically adapting between different tasks with minimum overhead on FPGA, we detail the hardware implementation. Figure 1 shows the whole framework.

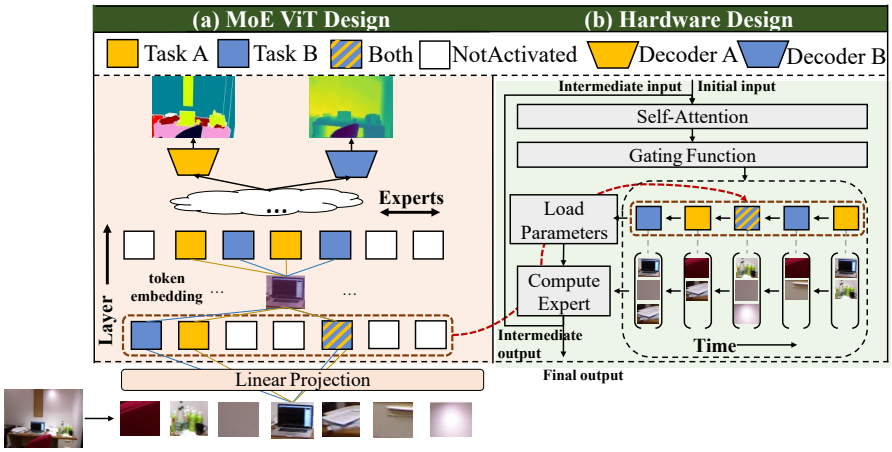
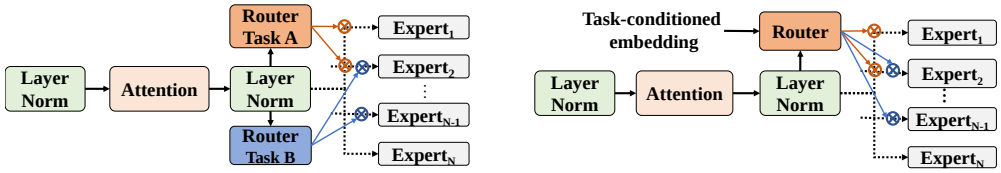


Figure 1: **The overall structure of the proposed M^{3}ViT pipeline**. The input image is split into fixed-size patches, embedded, and combined with position embeddings. In training, the MTL MoE ViT adaptively activates the model by sparsely selecting relevant experts using its task-dependent routers. During inference, only one task will be performed at a time. The hardware collects all patches allocated for each expert and processes them expert-by-expert with the "load parameters" and "compute expert" modules. We choose two experts out of sixteen candidates in each MoE layer.



(a) Multi-gate MoE layer design. **(b) Task-conditioned MoE layer design.**

Figure 2: **The proposed two variants of MTL MoE layers**. In the left figure, each task selects its experts using its own router. In the right one, all tasks share one router, while a task-specific embedding is concatenated with the token embedding to formulate the input of the shared router.

3.1 Task-dependent MoE ViT Design

Vision Transformer The representative Vision Transformer architecture [3] first splits the input image into non-overlapped patches and projects the patches to a higher hidden dimension using one convolutional layer. The projected patches (a.k.a. tokens) are then passed through several consecutive transformer layers. Each layer contains a self-attention module and a feed-forward-network (MLPs). The self-attention is computed using the scaled-dot product:

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{C}}\right)\boldsymbol{V} \tag{1}$$

where $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{N \times C}$ are the query, key and value matrices computed from input tokens; N and C indicate the token number and the hidden dimension. In our experiments, we adopt the DeiT [4] as the backbone encoder, which is a data-efficient ViT variant that distills tokens to ensure the student learns from the teacher through attention.

Mixture of Experts Layer A Mixture of Experts (MoE) layer typically consists a group of N experts $f_1, f_2, \cdots, f_N$ along with a router $\mathcal{R}$ (or gating network) to select the corresponding experts. The experts network can be several layers of convolutional filters [77] in CNNs or multi-layer perceptrons [60, 78] in ViTs. The router $\mathcal{R}$ plays a key role within our MoE ViT design as it determines task routings via only sparsely activating relevant experts. We adopt a representative router called top-K gating [30] based on ViT. With input $\boldsymbol{x}$, the resultant output of MoE layers can be formulated as the summation of the selected top K experts from N expert candidates using a router:

$$y = \sum_{k=1}^{K} \mathcal{R}(\boldsymbol{x})_k \cdot f_k(\boldsymbol{x}), \tag{2}$$

$$\mathcal{R}(\boldsymbol{x}) = \mathrm{TopK}(\mathrm{softmax}(\mathcal{G}(\boldsymbol{x}), K)), \tag{3}$$

$$\mathrm{TopK}(\boldsymbol{v}, K) = \begin{cases} \boldsymbol{v} & \text{if } \boldsymbol{v} \text{ is in the top } K \text{ elements} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $\mathcal{G}$ represents the learnable network within the router, for which we employ a single-layer MLP in practice. The $\mathrm{softmax}(\cdot)$ together with $\mathrm{TopK}(\cdot, K)$ sets all elements of the vector to zero except the elements with the largest K values. In practice, we choose K = 4 out of N = 16 expert candidates. Each expert is computed with $W_2\sigma_{\text{gelu}}(W_1 x)$, where σ_{gelu} is the GELU activation [79]. W_1 and W_2 are two learnable weight matrix. Note that we scale down the expert size by four times compared to that in standard ViT MLP layers to make the computation FLOPs equivalent. We also employ the load and important balancing loss with weight of 0.01 following [30] to avoid always picking the same experts while ignoring others. This loss term is also employed for the two task-dependent MTL MoE designs that we introduce next.

Multi-gate MoE ViT for MTL MoE brings training dynamics to balance between large capacity and efficiency, by selecting only a subset of experts using the router. To adapt vanilla MoE into our dense prediction MTL framework, we first propose to assign each dense prediction task a router $\mathcal{R}_i$ to specify its own experts, denoted as multi-gate MTL MoE ViT:

$$y_i = \sum_{k=1}^{K} \mathcal{R}^i(\boldsymbol{x})_k \cdot f_k(\boldsymbol{x}) \tag{5}$$

where i denotes task index. Expert candidates f^k are shared across tasks. The flow chart of the multi-gate variant is shown in Figure 2(a); task-dependent routers take as input the shared token embedding and do their expert selections.

Task-conditioned MoE ViT for MTL Conditional encoding has been widely applied to multi-modal [80] and multi-task [81] models. To achieve task-dependent routing with one gating network, we propose the task-conditioned MTL MoE ViT shown in Figure 2(b). Specifically, suppose we have n tasks in training. We manually define a n-dimensional one-hot task-conditioned vector. The vector is fed into a two-layer MLP to extract a 64-dimensional task embedding, which is then concatenated with token embeddings to form the task-dependent input for the router in the MoE layer:

$$y_i = \sum_{k=1}^{K} \mathcal{R}(\boldsymbol{x}, \boldsymbol{t_i})_k \cdot f_k(\boldsymbol{x}), \tag{6}$$

$$\boldsymbol{t_i} = \mathrm{ReLU}(\mathcal{T}(\boldsymbol{x}, \boldsymbol{e}_i)) \tag{7}$$

where $\mathcal{T}$ indicates the two-layer MLPs to extract task-conditioned embeddings, $\boldsymbol{e}_i \in \{0,1\}^n$, and $\sum_{j=1}^{n} e_j = 1$. We denote this conditional design as task-conditioned MTL MoE ViT, in which backbone model parameters do not proportionally increase if we include more tasks in training.

3.2 Circuit-level Implementation

We co-design the hardware to support MTL MoE ViT. We design a layer-wise implementation of M^{3}ViT on a Xilinx ZCU104 FPGA, a diagram of which is provided in the supplementary material. The design computes layers sequentially but parallelizes computation steps within each layer. By proposing a novel computation reordering scheme, our hardware design features memory-efficient expert computation that also achieves zero-overhead task switching and frame switching.

Challenges of Naive Method A straightforward (but naive) implementation would compute the output for each token in the order it appears in the input sequence: all tokens choose any K experts out of the N candidates, so ostensibly the only way to avoid data loading overhead would be to keep weights of all N experts on-chip at all times. However, this requires extreme on-chip memory usage, scaling with $O(N)$ and typically exceeding FPGA on-chip memory capacity unless N is very small.

Challenges of Cache-based Method We can adopt a cache to store several experts on-chip at any given time. However, this on-demand approach incurs long delays from off-chip DRAM accesses whenever the cache needs to be repopulated with an expert's weights. Further, we experimentally found that all experts are likely to be activated at least once across all tokens, exhibiting a cache-unfriendly access pattern. Therefore, although a cache-based design alleviates memory inefficiency, it incurs severe delays by frequently loading the weights of experts.

Proposed Solution: Memory-efficient Computation Reordering The crux of the problem lies in the unpredictability of the set of experts that will be needed by tokens at any given time. We address this problem at its root by designing a novel computation reordering scheme that flips the compute pattern on its head: rather than computing the MoE layer token-by-token, we instead compute it expert-by-expert. The overall flow chart of the reordering scheme is shown in Figure 3.

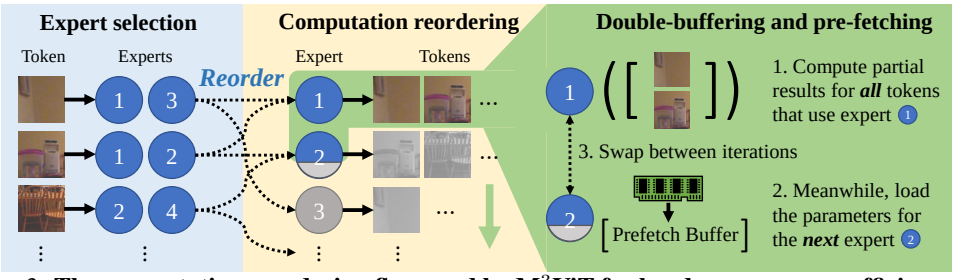


Figure 3: **The computation reordering flow used by M^{3}ViT for hardware memory efficiency.** The MoE gating function selects K experts for each token, which are used to route tokens to per-expert queues. This is followed by a double-buffered computation flow that computes one expert's results on its entire token queue while loading another expert's parameters, swapping buffers between iterations.

Specifically, we propose to add each token to a queue for its selected top-K experts, instead of computing the token output immediately. Our hardware then makes use of the per-expert queues via a double-buffered computation flow: one buffer is filled with an expert's weights from off-chip memory accesses, while another already-loaded buffer is used to compute another expert's results for its entire token queue. After both operations finish, the buffers are swapped, and the process repeats.

Scalability and Efficiency Our approach hides nearly all latency from off-chip memory accesses to load expert weights, and it uses $O(1)$ on-chip memory with respect to K and N, making it scalable to any number of experts. Additionally, our method's efficiency does not rely on any specific usage pattern of experts for a given frame or a given task, so we naturally achieve zero-overhead switches between frames and between tasks. Task switches and frame switches in our hardware design do not change our computation flow at all, and there is no specific step taken to execute the switch.

4 Experiments

4.1 Experiment Setup

To evaluate the propose method, we conduct experiments on two popular dense labeling MTL benchmarks, i.e. NYUD-v2 [2] and PASCAL-Context [1]. Both datasets are described below.

Datasets The **PASCAL-Context** [1] contains a total of 10,103 images, for the five tasks of edge detection (Edge), semantic segmentation (Seg.), human parts segmentation (H.Parts), surface normals (Norm.), and saliency detection (Sal.). The **NYUD-v2 dataset** [2] is an indoor dataset which consists of RGB-D images of 464 indoor scenes. There are 795 images for training and 654 images for testing, both with annotation for semantic segmentation (Seg.) and monocular depth estimation (Depth).

Evaluation Metrics For software level evaluation, we adopt the standard evaluation metrics following [14, 82, 49]. Particularly, we use mean intersection over union (mIoU) for semantic segmentation, human parts segmentation, and saliency; mean error (mErr) for surface normals estimation, root mean square error (rmse) for depth estimation; and optimal dataset F-measure (odsF) [83] for edge detection. Following [14], we use Δ_m to evaluate a MTL model m as the average per task drop with respect to the STL model b over all tasks: $\Delta_m = \frac{1}{T}\sum_i^T (-1)^{l_i}(M_{m,i} - M_{b,i})/M_{b,i}$, where $M_{m,i}$ and $M_{b,i}$ are the metrics of task i for the model m and b respectively, and $l_i = 1$ if a lower value means better performance.

Table 1: Comparisons with encoder-focused MTL architectures on the PASCAL-Context dataset.

Model	Backbone	Seg. (mIoU↑)	Norm. (mErr)↓	H. Parts (mIoU)↑	Sal. (mIoU)↑	Edge (odsF) ↑	Δ_m (%) ↑	FLOPS (G) ↓	Energy (W·s)↓
STL-B	ResNet-18	66.2	13.9	59.9	66.3	68.8	0.00	167	1.029
MTL-B	ResNet-18	63.8	14.9	58.6	65.1	69.2	−2.86	167	1.029
Uncertainty [25] (MTL-B)	ResNet-18	65.4	16.5	59.2	65.6	68.6	−4.60	167	1.029
DWA [51] (MTL-B)	ResNet-18	63.4	14.9	58.9	65.1	69.1	−2.94	167	1.029
GradNorm [20] (MTL-B)	ResNet-18	64.7	15.4	59.0	64.5	67.0	−3.97	167	1.029
MGDA [27] (MTL-B)	ResNet-18	64.9	15.6	57.9	62.5	61.4	−6.81	167	1.029
MTAN [27]	ResNet-18	63.7	14.8	58.9	65.4	69.6	−2.39	212	5.306
Cross-Stitch [23]	ResNet-18	66.1	**13.9**	60.6	**66.8**	69.9	+0.60	647	6.001
NDDR-CNN [26]	ResNet-18	65.4	**13.9**	60.5	**66.8**	69.8	+0.39	747	5.034
M-ViT (MTL-B)	ViT-small	70.7	15.5	58.7	64.9	68.8	−1.77	**83**	3.062
M^2ViT (+MoE)	MoE ViT-small	**72.8**	14.5	**62.1**	66.3	**71.7**	**+2.71**	84	7.446
M^3ViT (+MoE+Codesign)	MoE ViT-small	72.8	14.5	62.1	66.3	71.7	+2.71	84	**0.690**

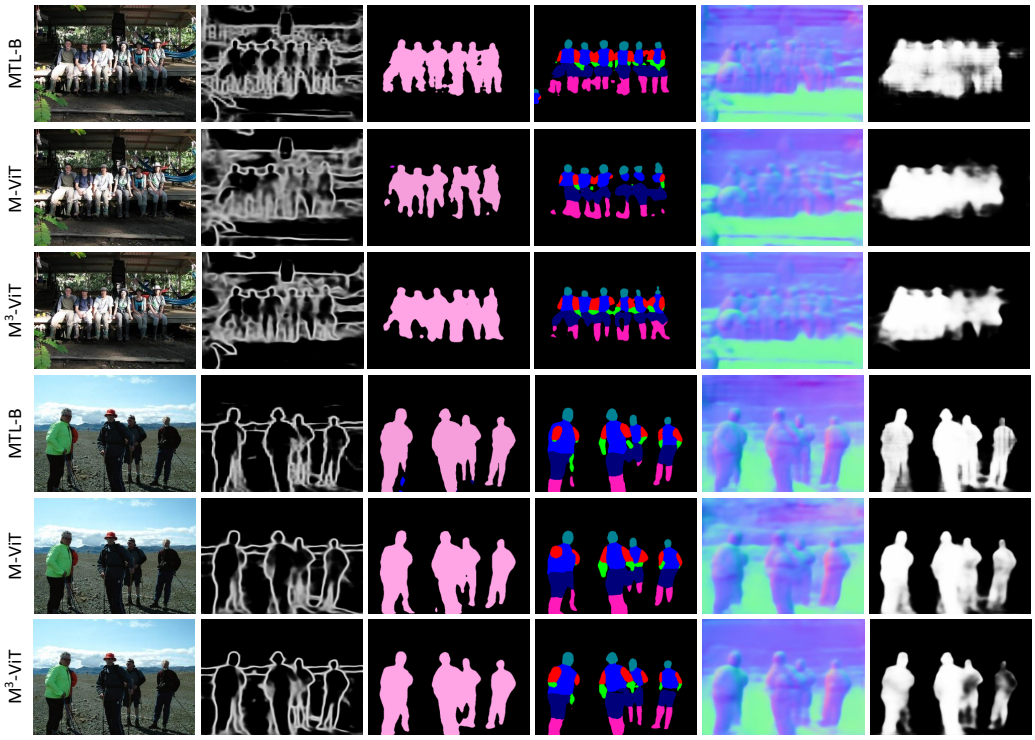


Figure 4: **Qualitative result on PASCAL-Context** We compare between vanilla MTL-B, M-ViT and M^2ViT models, and our model outperform baseline on edge detection, semantic segmentation and human parts segmentation and saliency detection.

To evaluate our model-accelerator design, we consider latency, energy usage (as the product of latency and power), and on-chip memory usage for single-task inference using a batch size of 1.

Network Configuration and Implementation Details We evaluate our model based on several versions of ViT backbone including ViT-tiny, ViT-small, and ViT-base. Our FPGA designs target the Xilinx ZCU104 FPGA at a 300 MHz clock frequency, consuming 10 W of power. The GPU baselines are measured on the NVIDIA Quadro RTX 8000. The results reported below are based on ViT-small. Please refer to the supplement for training setup, more details on network configuration, results on ViT-tiny and ViT-base, and details of our target hardware platforms.

Table 2: Comparisons with encoder-focused MTL architectures on the NYUD-v2 dataset.

Model	**Backbone**	**Seg.** (mIoU)↑	**Depth** (rmse)↓	$\mathbf{\Delta}_m$ (%) ↑	**FLOPS** (G) ↓	**Energy** (W·s) ↓
STL-B	ResNet-50	43.9	0.585	0.00	192	2.145
MTL-B	ResNet-50	44.4	0.587	+0.41	192	2.145
Uncertainty [25] (MTL-B)	ResNet-50	44.0	0.590	–0.23	192	2.145
DWA [51] (MTL-B)	ResNet-50	44.1	0.591	–0.28	192	2.145
GradNorm [20] (MTL-B)	ResNet-50	44.2	0.581	+1.45	192	2.145
MGDA [27] (MTL-B)	ResNet-50	43.2	0.576	+0.02	192	2.145
MTAN [27]	ResNet-50	45.0	0.584	+1.32	320	5.036
Cross-Stitch [23]	ResNet-50	44.2	**0.570**	**+1.61**	310	4.221
NDDR-CNN [26]	ResNet-50	44.2	0.573	+1.38	340	4.244
M-ViT (MTL-B)	ViT-small	40.9	0.631	–6.27	**100**	2.097
M^2ViT (+MoE)	MoE ViT-small	**45.6**	0.589	+1.59	**100**	8.189
M^3ViT (+MoE+Co-design)	MoE ViT-small	45.6	0.589	+1.59	**100**	**0.845**

Table 3: Effect of task-dependent MoE design. M^3ViT-Single, M^3ViT-Multi., and M^3ViT-Task-cond. refer to MTL MoE model with single router, multi routers, and task-conditioned router, respectively.

PASCAL-Context	**Seg.** (mIoU↑)	**Norm.** (mErr)↓	**H. Parts** (mIoU)↑	**Sal.** (mIoU)↑	**Edge** (odsF) ↑	$\mathbf{\Delta}_m$ (%) ↑	**FLOPS** (G) ↓
STL-B	66.2	**13.9**	59.9	**66.3**	68.8	0.00	167
M-ViT (MTL-B)	70.7	15.5	58.7	64.9	68.8	–1.76	**83**
M^3ViT-Single	71.5	14.8	61.2	65.9	71.5	+1.40	84
M^3ViT-Multi.	**72.8**	14.5	**62.1**	**66.3**	71.7	**+2.71**	84
M^3ViT-Task-cond.	72.0	14.4	61.3	65.8	**71.8**	+2.22	85
NYUD-v2	**Seg.** (mIoU)↑	**Depth** (rmse)↓	–	–	–	$\mathbf{\Delta}_m$ (%) ↑	**FLOPS** (G) ↓
STL-B	43.9	**0.585**	–	–	–	0.00	192
M-ViT (MTL-B)	40.9	0.631	–	–	–	–6.27	**100**
M^3ViT-Single	45.3	0.600	–	–	–	+0.31	**100**
M^3ViT-Multi.	**45.6**	0.589	–	–	–	**+1.59**	**100**
M^3ViT-Task-cond.	45.3	0.595	–	–	–	+0.74	101

4.2 Comparison with State-of-the-art Dense Prediction MTL

As we target an efficient MTL system under the single-task inference setting, we conduct experiments on encoder-focused architectures (more details in Section 2). MTL-B [14] is a vanilla multi-task learning baseline model which is composed of a shared backbone in combination with task-specific heads. Several state-of-the-art (SoTA) encoder-focused MTL models, including MTAN [27], Cross-Stitch [23] and NDDR-CNN [26], improve MTL-B by proposing feature sharing methods in the encoder. Our methods are all conducted on vanilla MTL-B, namely, applying MTL on ViT (M-ViT), adding task-dependent MoE design (M^2ViT), and adding hardware co-design on FPGA (M^3ViT). We also compare with previous works that handle the multi-task training conflicts problem, including uncertainty weighting [25], GradNorm [20], DWA [51], and MGDA [27], and they are evaluated on MTL-B. Single task learning baseline (STL-B) is used for MTL performance evaluation Δ_m. As multi-gate MoE shows better performance than task-conditioned MoE, the reported M^2ViT and M^3ViT results are based on the multi-gate design.

Results on PASCAL-Context Dataset As shown in Table 1, even using a vanilla MTL-B framework, introducing MoE (M^2ViT) can achieve the highest performance over all previous encoder-focused works (+2.71% MTL performance); meanwhile, it significantly reduces their single task inference FLOPs (particularly, reducing Cross-Stitch by 88%). Comparing against Uncertainty, DWA, GradNorm, and MGDA, the superior performance of M^2ViT demonstrates its strong capacity in handling training conflict. Moreover, leveraging the Model-Accelerator co-design helps us to consume less than one-tenth the energy cost when deploying our model on FPGA. Some qualitative results are shown in Figure 4.

Table 4: Quantitative comparisons of hardware metrics between FPGA and GPU implementations. **CR** indicates usage of our memory-efficient computation reordering method (Section 3.2).

Platform	Backbone	CR	Memory (MiB)	PASCAL-Context		NYUD-v2	
				Latency (ms)	**Energy** (W·s)	**Latency** (ms)	**Energy** (W·s)
GPU	ResNet-18	–	21.336	3.489	1.029	–	–
GPU	ResNet-50	–	44.939	–	–	7.270	2.145
GPU	ViT-small	–	42.058	10.381	3.062	7.110	2.097
GPU	MoE ViT-small	–	82.747	25.239	7.446	27.760	8.189
FPGA	ViT-small	–	4.828	68.931	0.689	84.418	0.844
FPGA	MoE ViT-small	✗	4.840	637.478	6.375	750.557	7.506
FPGA	MoE ViT-small	✓	4.840	69.033	0.690	84.538	0.845

Results on NYUD-v2 Dataset On this dataset, M^2ViT can reduce previous SoTA's inference FLOPs by 68% while achieving comparable MTL performance. Introducing MoE to M-ViT helps to enlarge the model capacity without increasing inference FLOPs, which results in a MTL performance boost from −6.27% to +1.59%. With our Model-Accelerator co-design, we also see a nearly tenfold increase in energy efficiency. Results are shown in Table 2.

4.3 Effect of Task-dependent MoE Design

To evaluate the effectiveness of our task-dependent MoE design, we compare between several models in Table 3 including STL-B, MTL ViT (M-ViT), M^3ViT with one gating function for all the tasks, M^3ViT with multi gates, and M^3ViT with task-conditioned token input. Results on both PASCAL-Context and NYUD-v2 datasets show that adding MoE layers into ViT with only one gating function for all tasks can already improve the M-ViT model. Making MoE selection task-dependent can further improve the performance, where multi-gating performs better than task-conditioned gating design. Particularly, comparing our model performance with STL-B as well as previous SoTAs in Table 1 and 2, we find that our MoE model better demonstrates its effectiveness when more tasks need to be encapsulated in the system. More results about our model's performance on different numbers of tasks can be found in the supplement.

4.4 Hardware Performance Results

Results comparing hardware performance metrics on FPGA and GPU are shown in Table 4. We first discuss our memory efficiency. The naive approach described in Section 3.2 would require 11.610 MiB of on-chip memory (too much for our FPGA platform), but our compute-reordering design achieves the same result using only 4.840 MiB, demonstrating a 2.40× reduction. Moreover, when comparing against a memory-constrained MoE ViT on FPGA without our compute-reordering method (using the cache-based method described in Section 3.2), we see that our method takes 9.23× less latency and energy on the PASCAL-Context dataset and 8.88× less latency and energy on NYUD-v2. Our compute-reordering M^3ViT on FPGA also beats all GPU baselines in energy efficiency; e.g., it beats MoE ViT on GPU by 10.79× on PASCAL-Context and by 9.69× on NYUD-v2. For discussion on the latency breakdown of M^3ViT, please refer to the supplement.

5 Conclusion, Discussion of Limitation and Broader Impact

In this paper, we propose a model-accelerator co-design for efficient on-device MTL. By customizing MTL mixture-of-experts layers into a ViT backbone, we achieve the following: (i) we sparsely activate task-specific experts in training to mitigate MTL gradient conflicts. (ii) Given any task of interest, we can activate only the task-corresponding sparse "expert" pathway for efficiency. With our hardware-level co-design, we can achieve zero-overhead switching between tasks and can scale to any number of experts. Extensive experiments that show the proposed M^3ViT surpasses the top-performing encoder-focused MTL methods, reduces 88% FLOPs, and saves more than 8× energy over our baseline.

The limitation of our work is that the proposed M^3ViT is mainly evaluated on academic datasets; we will try real applications like autonomous driving in the future. For broader impact, this work reduces the resource and energy consumption needed for MTL regimes, while still maintaining SoTA performance, which can effectively serve the goal of GreenAI.

References

[1] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 891–898, 2014.

[2] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[3] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[4] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers amp; distillation through attention. In *International Conference on Machine Learning*, volume 139, pages 10347–10357, July 2021.

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[7] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021.

[8] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

[9] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021.

[10] Dong-Gyu Lee. Fast drivable areas estimation with multi-task learning for real-time autonomous driving assistant. *Applied Sciences*, 11(22):10713, 2021.

[11] Wonsuk Kim and Junhee Seok. Indoor semantic segmentation for robot navigating on mobile. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 22–25. IEEE, 2018.

[12] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*, 2016.

[13] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[14] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[15] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. *Advances in neural information processing systems*, 26, 2013.

[16] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018.

[17] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1930–1939, 2018.

[18] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[19] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[20] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning*, pages 794–803. PMLR, 2018.

[21] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33:2039–2050, 2020.

[22] Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. *arXiv preprint arXiv:2010.05874*, 2020.

[23] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[24] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. Dynamic task prioritization for multitask learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 270–287, 2018.

[25] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[26] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3205–3214, 2019.

[27] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[29] Introducing Pathways: A next-generation AI architecture. `https://blog.google/technology/ai/introducing-pathways-next-generation-ai-architecture/`.

[30] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[31] Sebastian Jaszczur, Aakanksha Chowdhery, Afroz Mohiuddin, Lukasz Kaiser, Wojciech Gajewski, Henryk Michalewski, and Jonni Kanerva. Sparse is enough in scaling transformers. *Advances in Neural Information Processing Systems*, 34:9895–9907, 2021.

[32] Yoshua Bengio. Deep learning of representations: Looking forward. In *International conference on statistical language and speech processing*, pages 1–37. Springer, 2013.

[33] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8(1), 2007.

[34] Laurent Jacob, Jean-philippe Vert, and Francis Bach. Clustered multi-task learning: A convex formulation. *Advances in neural information processing systems*, 21, 2008.

[35] Jiayu Zhou, Jianhui Chen, and Jieping Ye. Clustered multi-task learning via alternating structure optimization. *Advances in neural information processing systems*, 24, 2011.

[36] BJ Bakker and TM Heskes. Task clustering and gating for bayesian multitask learning. 2003.

[37] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd international conference on Machine learning*, pages 1012–1019, 2005.

[38] Su-In Lee, Vassil Chatalbashev, David Vickrey, and Daphne Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of the 24th international conference on Machine learning*, pages 489–496, 2007.

[39] Hal Daumé III. Bayesian multitask learning with latent hierarchies. *arXiv preprint arXiv:0907.0783*, 2009.

[40] Abhishek Kumar and Hal Daume III. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*, 2012.

[41] Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. One model to learn them all. *arXiv preprint arXiv:1706.05137*, 2017.

[42] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. *arXiv preprint arXiv:1901.11504*, 2019.

[43] Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 12-in-1: Multi-task vision and language representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10437–10446, 2020.

[44] Ronghang Hu and Amanpreet Singh. Unit: Multimodal multitask learning with a unified transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1439–1449, 2021.

[45] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, et al. Florence: A new foundation model for computer vision. *arXiv preprint arXiv:2111.11432*, 2021.

[46] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829, 2019.

[47] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 675–684, 2018.

[48] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Yan Yan, Nicu Sebe, and Jian Yang. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4106–4115, 2019.

[49] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Zequn Jie, Xiang Li, and Jian Yang. Joint task-recursive learning for semantic segmentation and depth estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 235–251, 2018.

[50] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. In *European Conference on Computer Vision*, pages 527–543. Springer, 2020.

[51] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018.

[52] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[53] Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

[54] Ke Chen, Lei Xu, and Huisheng Chi. Improved learning algorithms for mixture of experts in multiclass classification. *Neural networks*, 12(9):1229–1252, 1999.

[55] Seniha Esen Yuksel, Joseph N. Wilson, and Paul D. Gader. Twenty years of mixture of experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1177–1193, 2012.

[56] Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason E Weston. Hash layers for large sparse models. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[57] Dheeru Dua, Shruti Bhosale, Vedanuj Goswami, James Cross, Mike Lewis, and Angela Fan. Tricks for training sparse translation models. *arXiv preprint arXiv:2110.08246*, 2021.

[58] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

[59] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[60] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

[61] Jakob Vogdrup Hansen. Combining predictors: comparison of five meta machine learning methods. *Information Sciences*, 119(1-2):91–105, 1999.

[62] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR, 2021.

[63] Aidan Clark, Diego de las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. *arXiv preprint arXiv:2202.01169*, 2022.

[64] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. Network of experts for large-scale image categorization. In *European Conference on Computer Vision*, pages 516–532. Springer, 2016.

[65] Sam Gross, Marc'Aurelio Ranzato, and Arthur Szlam. Hard mixtures of experts for large scale weakly supervised vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6865–6873, 2017.

[66] Xin Wang, Fisher Yu, Lisa Dunlap, Yi-An Ma, Ruth Wang, Azalia Mirhoseini, Trevor Darrell, and Joseph E Gonzalez. Deep mixture of experts via shallow embedding. In *Uncertainty in artificial intelligence*, pages 552–562. PMLR, 2020.

[67] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in Neural Information Processing Systems*, 32, 2019.

[68] Alhabib Abbas and Yiannis Andreopoulos. Biased mixtures of experts: Enabling computer vision inference under data transfer limitations. *IEEE Transactions on Image Processing*, 29:7656–7667, 2020.

[69] Svetlana Pavlitskaya, Christian Hubschneider, Michael Weber, Ruby Moritz, Fabian Huger, Peter Schlicht, and Marius Zollner. Using mixture of expert models to gain insights into semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 342–343, 2020.

[70] Raquel Aoki, Frederick Tung, and Gabriel L Oliveira. Heterogeneous multi-task learning with expert diversity. *arXiv preprint arXiv:2106.10595*, 2021.

[71] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[72] Mengshu Sun, Haoyu Ma, Guoliang Kang, Yifan Jiang, Tianlong Chen, Xiaolong Ma, Zhangyang Wang, and Yanzhi Wang. Vaqf: Fully automatic software-hardware co-design framework for low-bit vision transformer. *arXiv preprint arXiv:2201.06618*, 2022.

[73] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Bin Ren, Minghai Qin, Hao Tang, and Yanzhi Wang. Spvit: Enabling faster vision transformers via soft token pruning. *arXiv preprint arXiv:2112.13890*, 2021.

[74] Bingbing Li, Santosh Pandey, Haowen Fang, Yanjun Lyv, Ji Li, Jieyang Chen, Mimi Xie, Lipeng Wan, Hang Liu, and Caiwen Ding. Ftrans: energy-efficient acceleration of transformers using fpga. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, pages 175–180, 2020.

[75] Panjie Qi, Yuhong Song, Hongwu Peng, Shaoyi Huang, Qingfeng Zhuge, and Edwin Hsing-Mean Sha. Accommodating transformer onto fpga: Coupling the balanced model compression and fpga-implementation optimization. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pages 163–168, 2021.

[76] Hongwu Peng, Shaoyi Huang, Tong Geng, Ang Li, Weiwen Jiang, Hang Liu, Shusen Wang, and Caiwen Ding. Accelerating transformer-based deep learning models on fpgas using column balanced block pruning. In *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pages 142–148. IEEE, 2021.

[77] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella Yu. Long-tailed recognition by routing diverse distribution-aware experts. In *International Conference on Learning Representations*, 2021.

[78] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems*, 34, 2021.

[79] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[80] Chao Xiong, Xiaowei Zhao, Danhang Tang, Karlekar Jayashree, Shuicheng Yan, and Tae-Kyun Kim. Conditional convolutional neural network for modality-aware face recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3667–3675, 2015.

[81] Jonathan Pilault, Amine Elhattami, and Christopher Pal. Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data. *arXiv preprint arXiv:2009.09139*, 2020.

[82] Guolei Sun, Thomas Probst, Danda Pani Paudel, Nikola Popović, Menelaos Kanakis, Jagruti Patel, Dengxin Dai, and Luc Van Gool. Task switching network for multi-task learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8291–8300, 2021.

[83] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] please see section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] please see section 5
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] Our work does not contain theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A] Our work does not contain theoretical results.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We used publicly available data in all of our experiments. Meanwhile we either provide the detailed implementations or cite the papers of them following the authors instructions (please refer to our implementation details and appendix). All of our codes will be released.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We provides all the training details in appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We test our model on PASCAL-Context and NYUD-v2 for three times and report their mean results.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We describe the details of computation resources in the appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We used publicly available data, i.e., PASCAL-Context, in our experiments. We cited the corresponding papers published by the creators.
   (b) Did you mention the license of the assets? [N/A] The license of ImageNet, PASCAL-Context, NYUD-v2 are included in the paper that we have cited.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] The ImageNet we used are publicly available.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We did not collect/curate new data.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] ImageNet and the used MTL datasets are already publicly available and broadly adopted. We do not think there are any issues of personally identifiable information or offensive content
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]