Learning Based Spatial Power Characterization and Full-Chip Power Estimation for Commercial TPUs

Jincong Lu, **Jinwei Zhang**, Wentian Jin, Sachin Sachdeva, Sheldon Tan

VSCLAB

Department of Electrical and Computer Engineering

University of California, Riverside

Outline

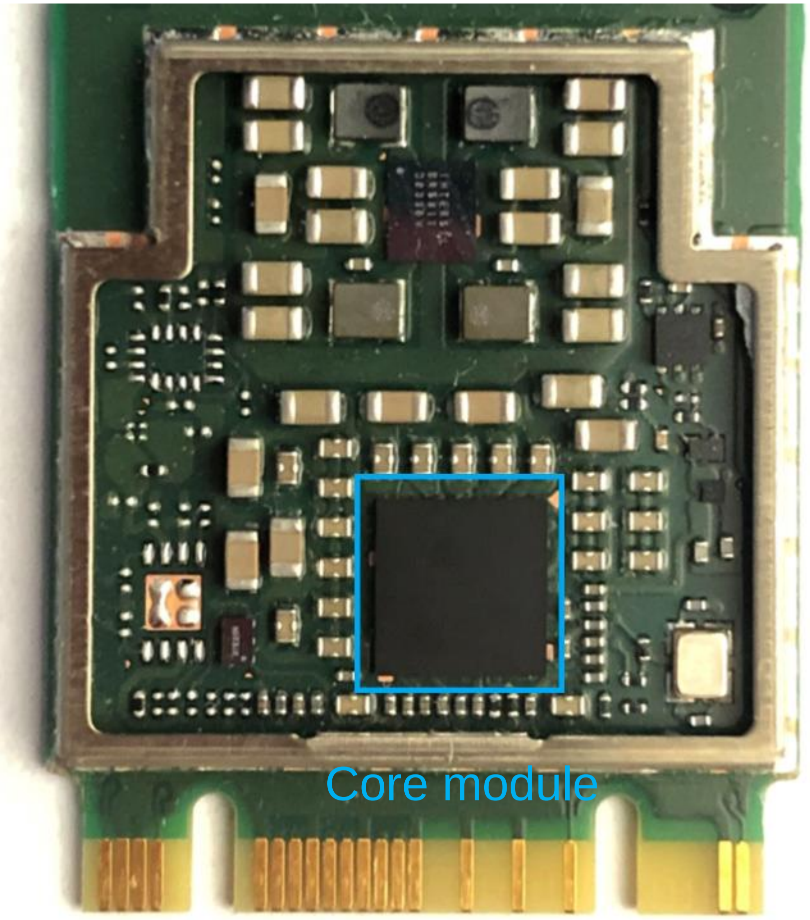
- Background of TPU Power Estimation
- Proposed Approach
  - Power map measurement
  - Power map prediction
- Results & Discussion
- Conclusion

Background

- For modern processors, with better performance, the heat produced increases
- Effective thermal control is important
- Accurate spatial power information of the entire chip area benefits the control decision

Background

- Temperature sensors cost space and energy
- Number of on-chip sensors is limited
- E.g., Tensor Processor Unit (TPU)

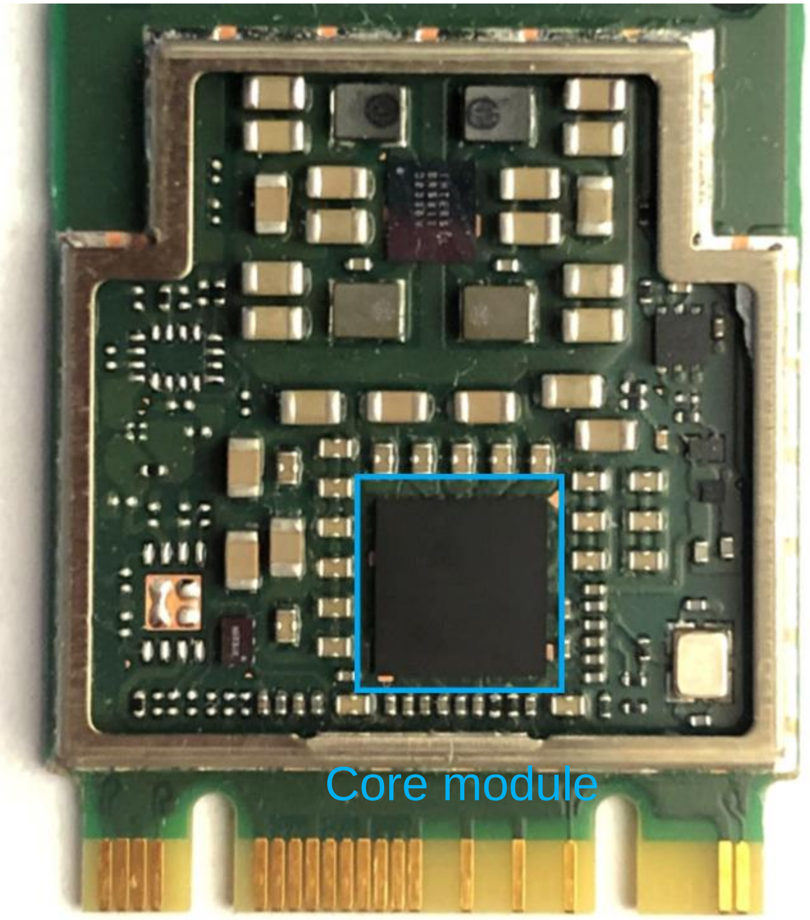


Coral Edge TPU

Background

Tensor Processing Unit (TPU)

- Application-specific integrated circuit (ASIC)
- Designed for machine learning computation
- Unlike GPU - No graphics hardware
- More computation power per joule

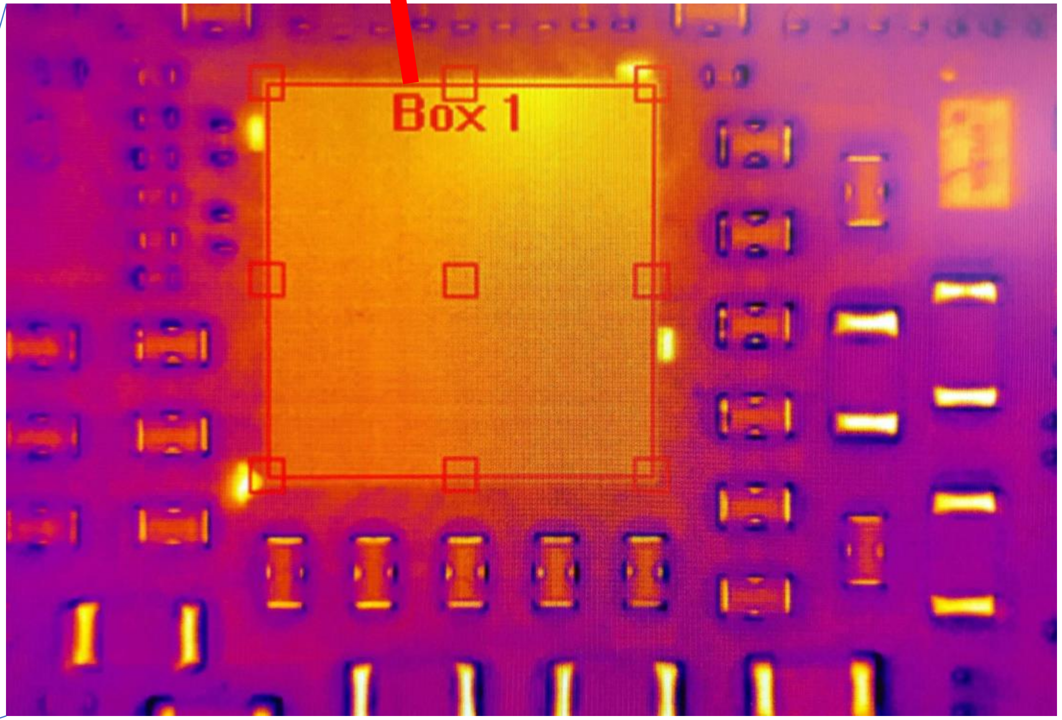


Coral Edge TPU

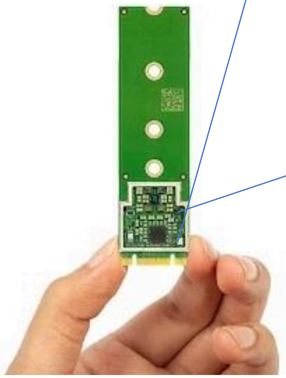
Background

- Sensors cost space and energy
- Number of on-chip sensors is limited
- E.g., Tensor Processor Unit (TPU)

The core module: 4.94mm * 5.06mm



Only one overall temperature sensor

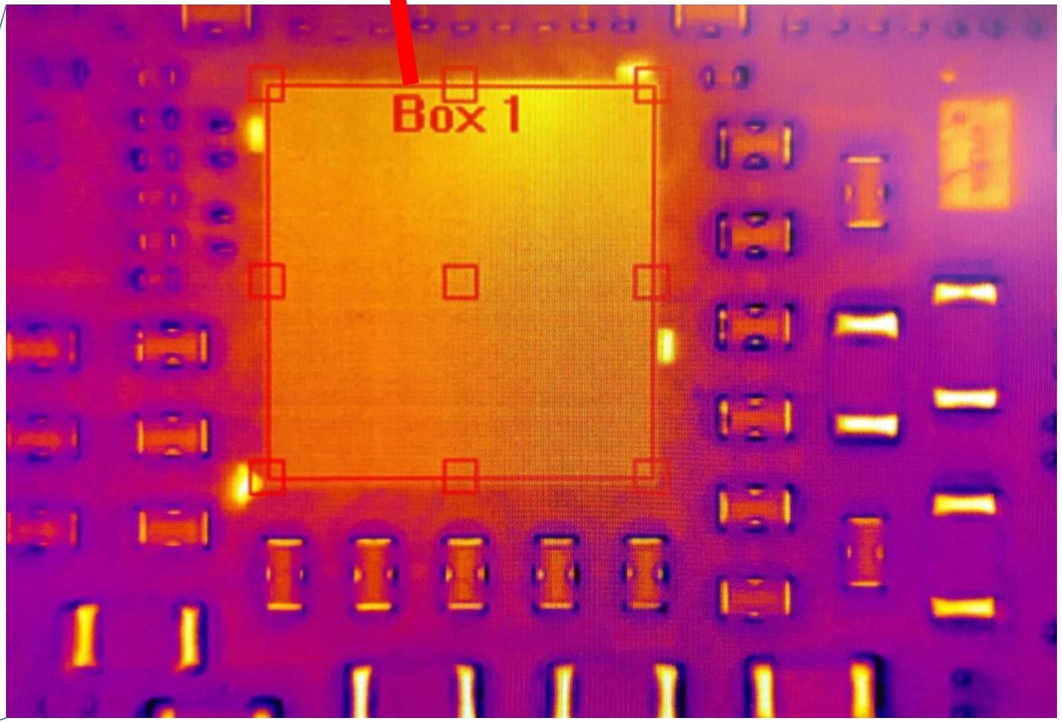


Coral Edge TPU

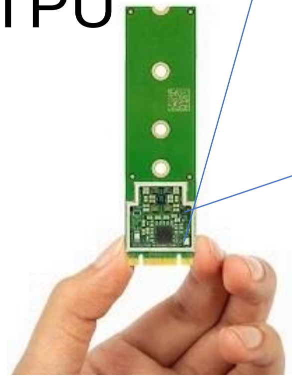
Background

- Sensors cost space and energy
- Number of on-chip sensors is limited
- E.g., Tensor Processor Unit (TPU)
- Power characterization for TPU is rarely studied

The core module: 4.94mm * 5.06mm



Only one overall temperature sensor



Coral Edge TPU

UC RIVERSIDE

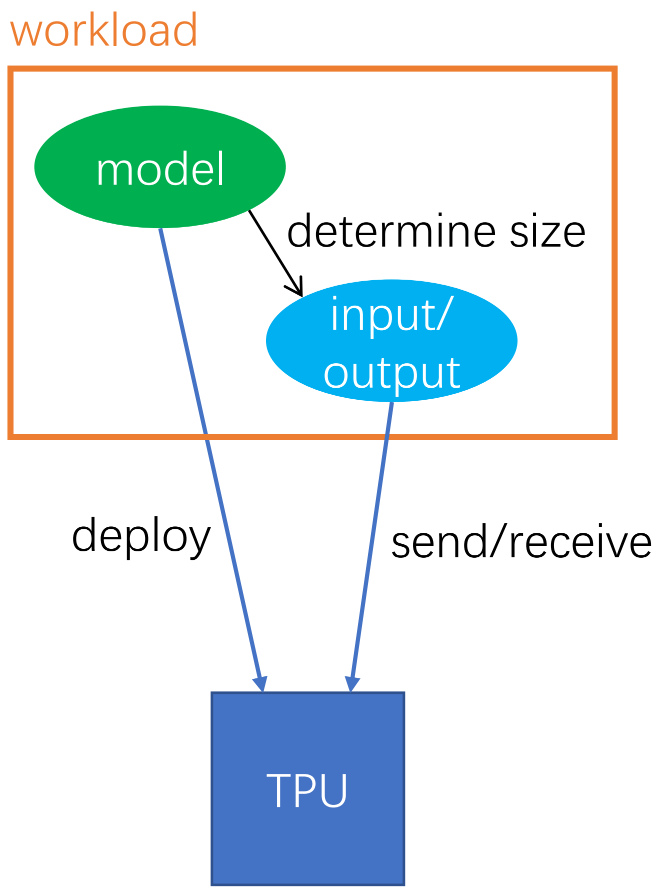
Method

- Characterized TPU power distribution by measuring its surface temperature in thermal steady-state

- Applied machine learning model to establish the relationship between the TPU workload and TPU power map

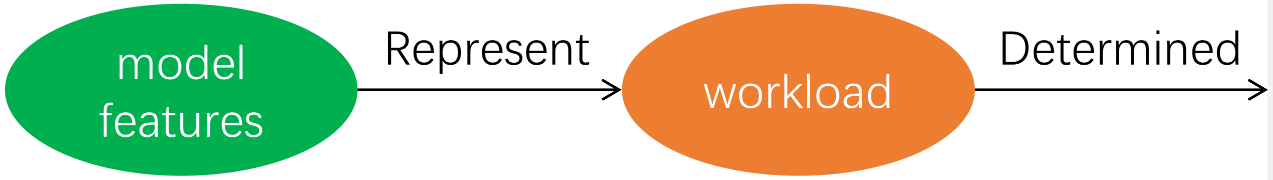
TPU workload

- TPU workloads consists of
    1. A pre-compiled machine learning model
    2. Input/output data
- When the model is given, I/O size is fixed, so the workload is uniquely determined
- Model features represents workload

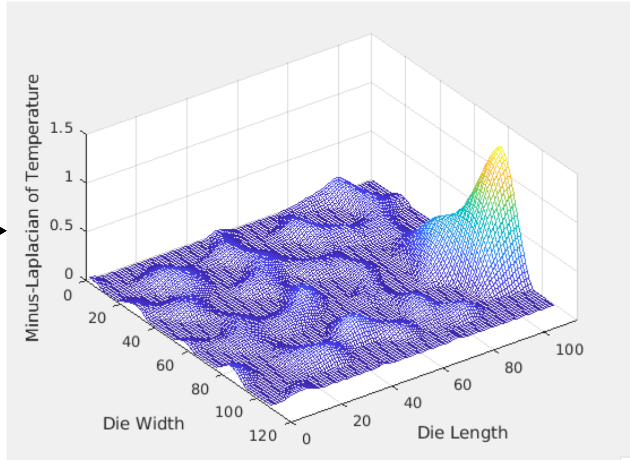


Workload to power map

- Model features represents workload
- Workload determine power distribution
- We can build a connection between model features and power distribution
  (From now on we call them workload features, to distinguish it from our own model later)



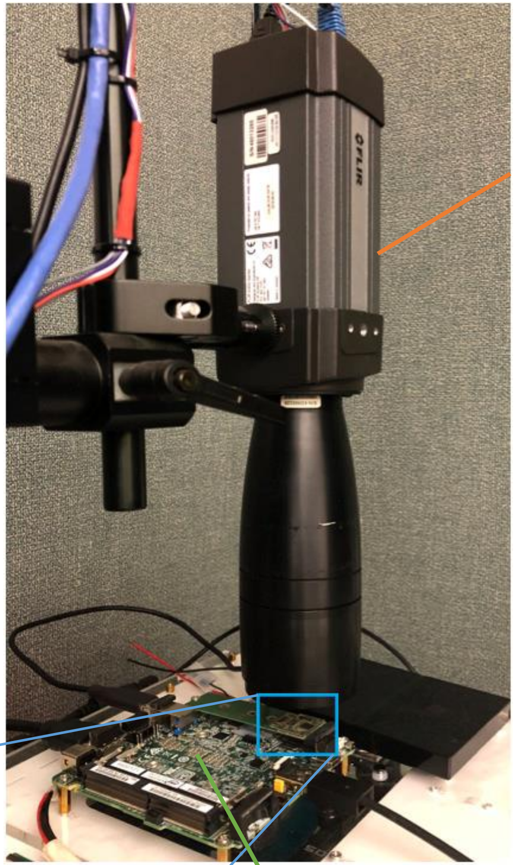
(Software level)



power density

Power map measurement

- We cannot directly measure power distribution
- Instead, we may measure thermal map, then calculate the power map
- Thermal imaging system



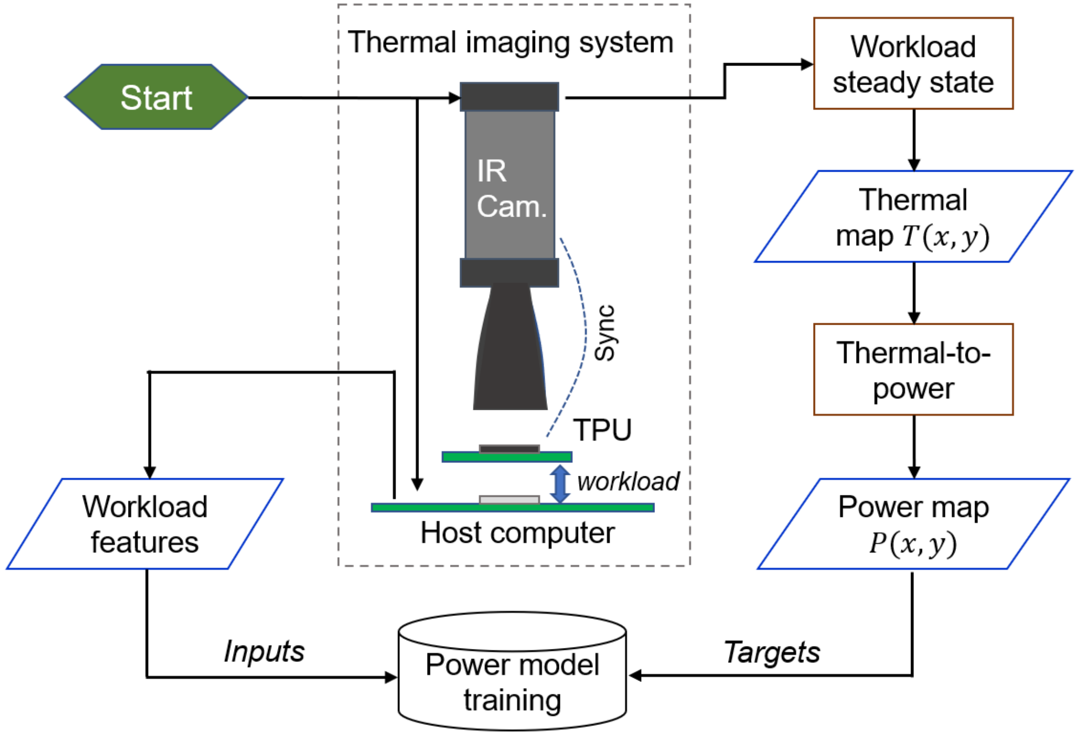
FLIR A325sc

Host board: Intel i7-8650U

Coral Edge TPU

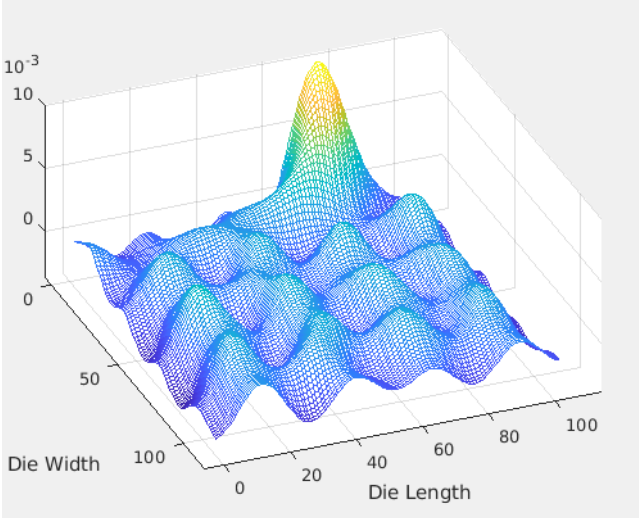
Power map measurement

- Run workload
- Measure thermal map
- Convert thermal map to power map
- Feed features and maps to our model



Thermal-to-power approach

- An approximation approach for chips
- Verified by simulation with a high precision



Negative-Laplacian

- Consider the steady state 2D spatial thermal distribution of TPU as $T(x,y)$

$$p(x,y) \approx \begin{cases} k[-\nabla^2 T(x,y)] & -\nabla^2 T(x,y) > 0 \\ 0 & -\nabla^2 T(x,y) \leq 0 \end{cases}$$

with

$$k = \kappa \Delta z$$

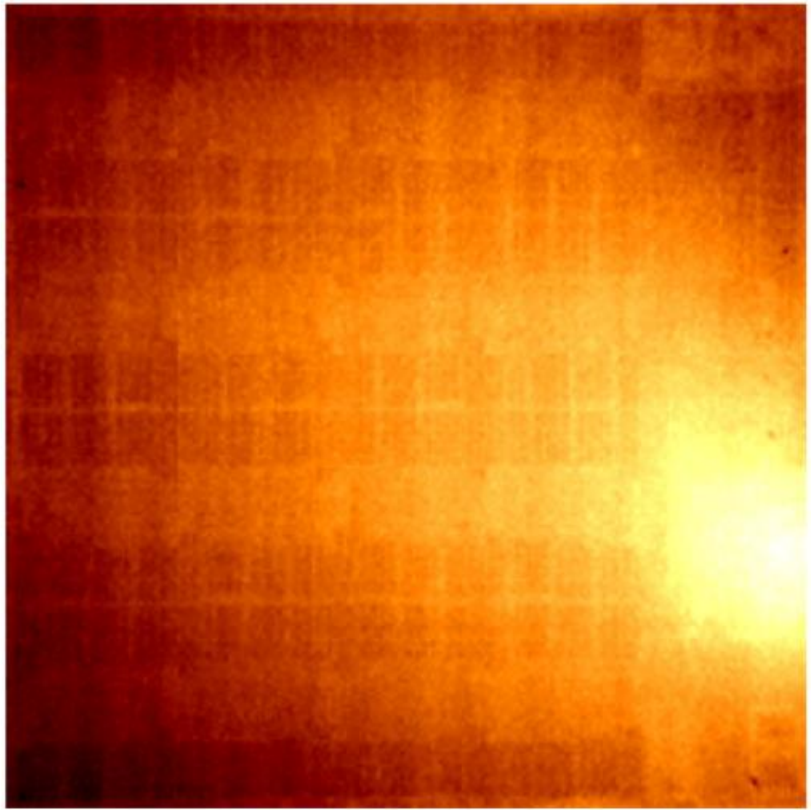
Where $p(x,y)$ stands for the power density distribution

κ and Δz for thermal conductivity and chip thickness

UC RIVERSIDE

Thermal-to-power approach

- Noise can be a big problem when calculate ∇^2
- The local correlation of noise is very low, but the real temperature map is smoother
- ∇^2 of noise significantly affect $\nabla^2 T$



Thermal-to-power approach

Discrete Cosine Transform (DCT)

- Convert thermal map to spatial frequency domain

$$D_{uv} = \alpha_u \alpha_v \sum_{x=1}^{W} \sum_{y=1}^{H} T(x,y) \cos\left(\frac{\pi(2x-1)u}{2W}\right) \cos\left(\frac{\pi(2y-1)v}{2H}\right), \qquad \begin{matrix} 0 \le u < W \\ 0 \le v < H \end{matrix}$$

where H, W are the height and width of the map, and

$$\alpha_u = \begin{cases} \sqrt{1/W} & u = 0 \\ \sqrt{2/W} & 1 \le u < W \end{cases} \qquad \alpha_v = \begin{cases} \sqrt{1/H} & v = 0 \\ \sqrt{2/H} & 1 \le v < H \end{cases}$$

- u,v are spatial frequencies along x,y
- $\{D_{uv}\}$ are the DCT frequency coefficients.

Thermal-to-power approach

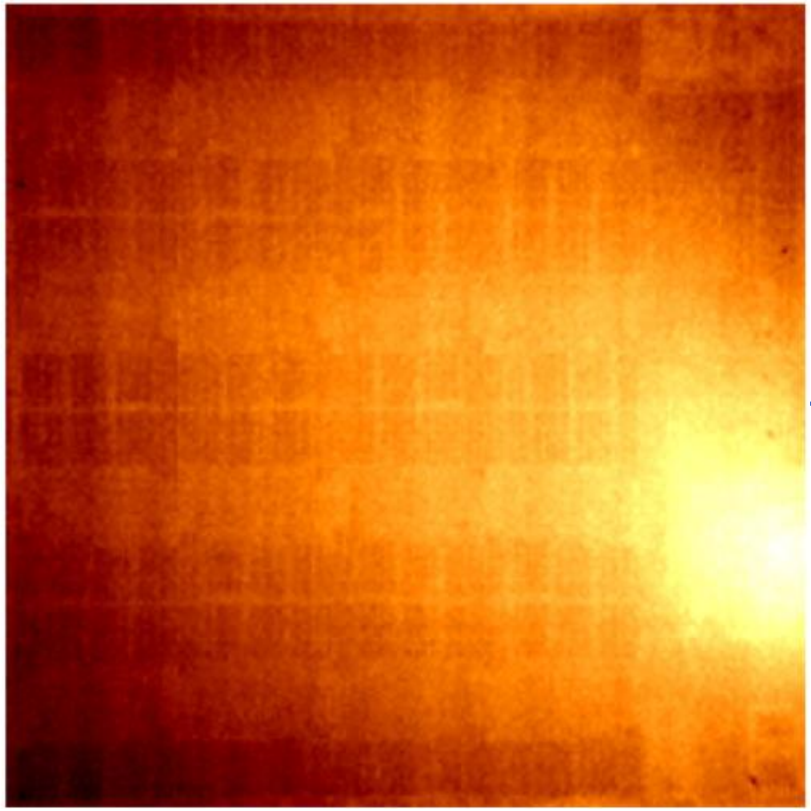
- Noise has a high frequency because it has no spatial relevance
- Major information of thermal map is in a few low-frequency coefficients
- We can remove the noise by dropping the high frequency terms

$$T(x,y) = \sum_{u=0}^{W-1} \sum_{v=0}^{H-1} \alpha_u \alpha_v D_{uv} \cos\left(\frac{\pi(2x-1)u}{2W}\right) \cos\left(\frac{\pi(2y-1)v}{2H}\right), \qquad \begin{array}{c} 1 \leq x \leq W \\ 1 \leq y \leq H \end{array}$$

$$\downarrow$$

$$T'(x,y) = \sum_{u=0}^{f} \sum_{v=0}^{f} \alpha_u \alpha_v D_{uv} \cos\left(\frac{\pi(2x-1)u}{2W}\right) \cos\left(\frac{\pi(2y-1)v}{2H}\right), \qquad \begin{array}{c} 1 \leq x \leq W \\ 1 \leq y \leq H \end{array}$$

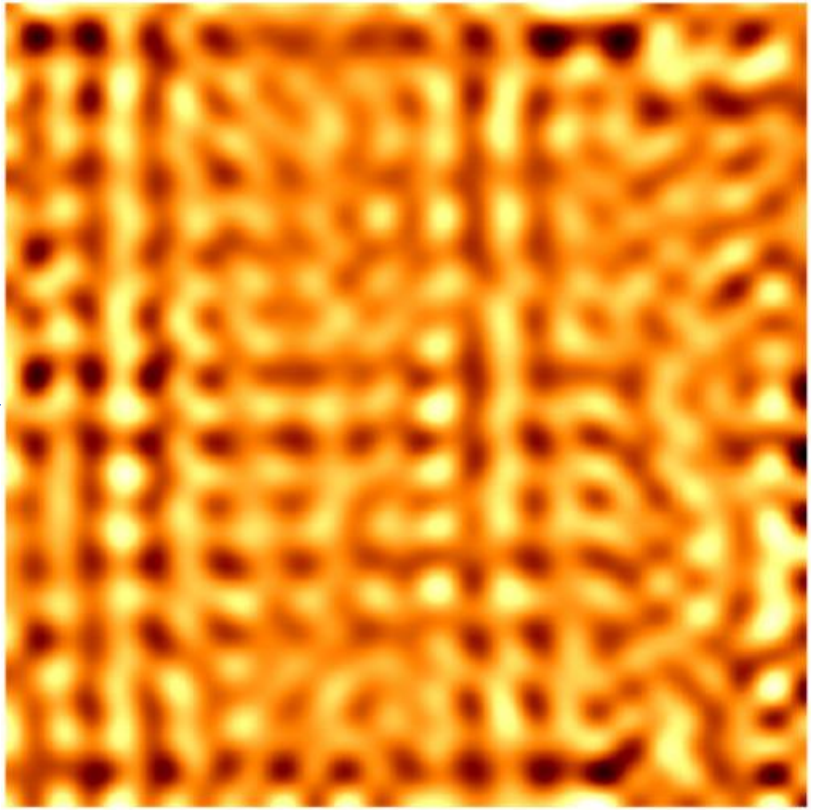
Thermal-to-power approach



Original thermal map

Remove high-frequency noise

$-\nabla^2$



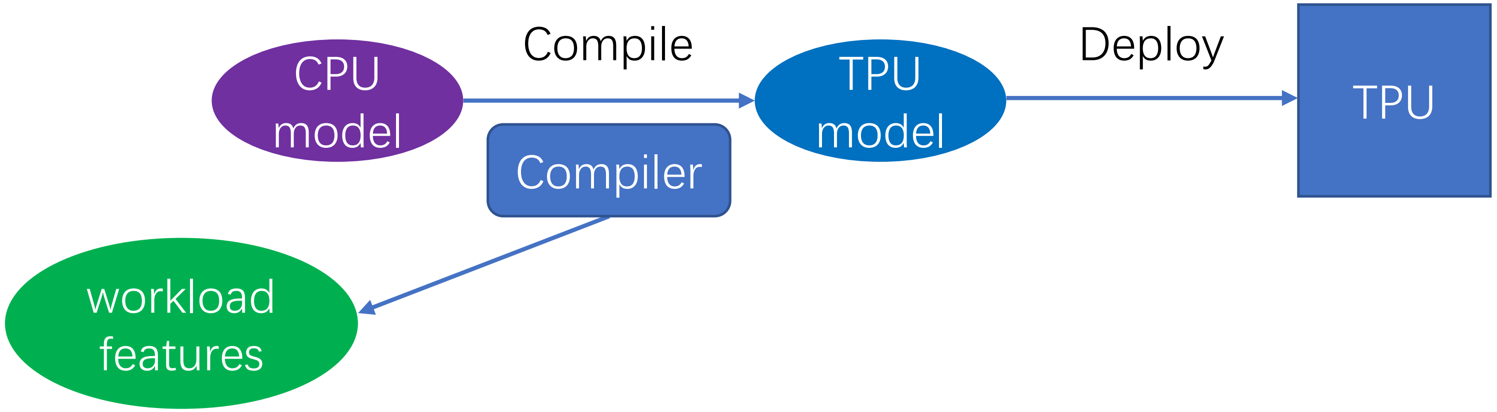
Power map

Workload features

1. image_shape
2. width_multiplier
3. depth_multiplier
4. pooling_mode
5. model_size
6. onchip_mem_used
7. onchip_mem_remain
8. offchip_mem_used
9. total num of op
10. num of op on TPU
11. num of op on CPU
12. inference time on TPU
13. ADD count
14. AVERAGE_POOL_2D count
15. CONCATENATION count
16. CONV_2D count
17. FULLY_CONNECTED count
18. DEPTHWISE_CONV_2D count
19. L2_NORMALIZATION count
20. MAX_POOL_2D count
21. MEAN count
22. MUL count
23. PAD count
24. QUANTIZE count
25. RESHAPE count
26. SOFTMAX count
27. SUB count
28. RELU count
29. REDUCE_MAX count
30. STRIDED_SLICE count
31. HARD_SWISH count

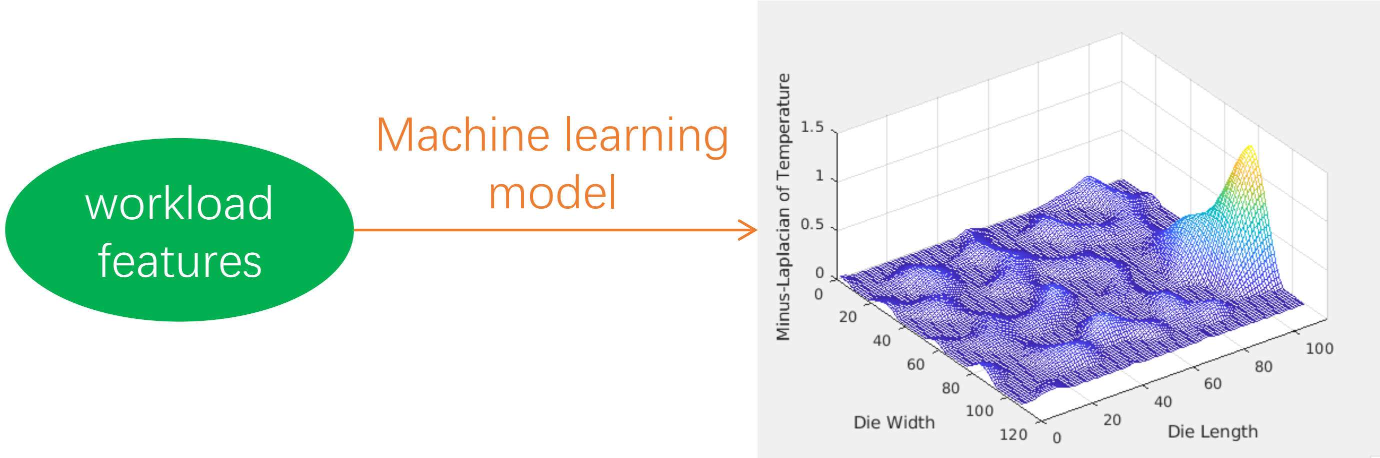
Machine learning model on TPU

- A machine learning model cannot directly run on TPU
- We need to compile it from CPU version to TPU version
- This process record which operations are supported on TPU, model size, memory usage, …
- All these features can be collected from the compilation report



Learning-based estimation

- It is difficult for experiments to cover all workloads
- We cannot keep all the workload-map pairs at runtime
- Solution: train a model to estimate power maps

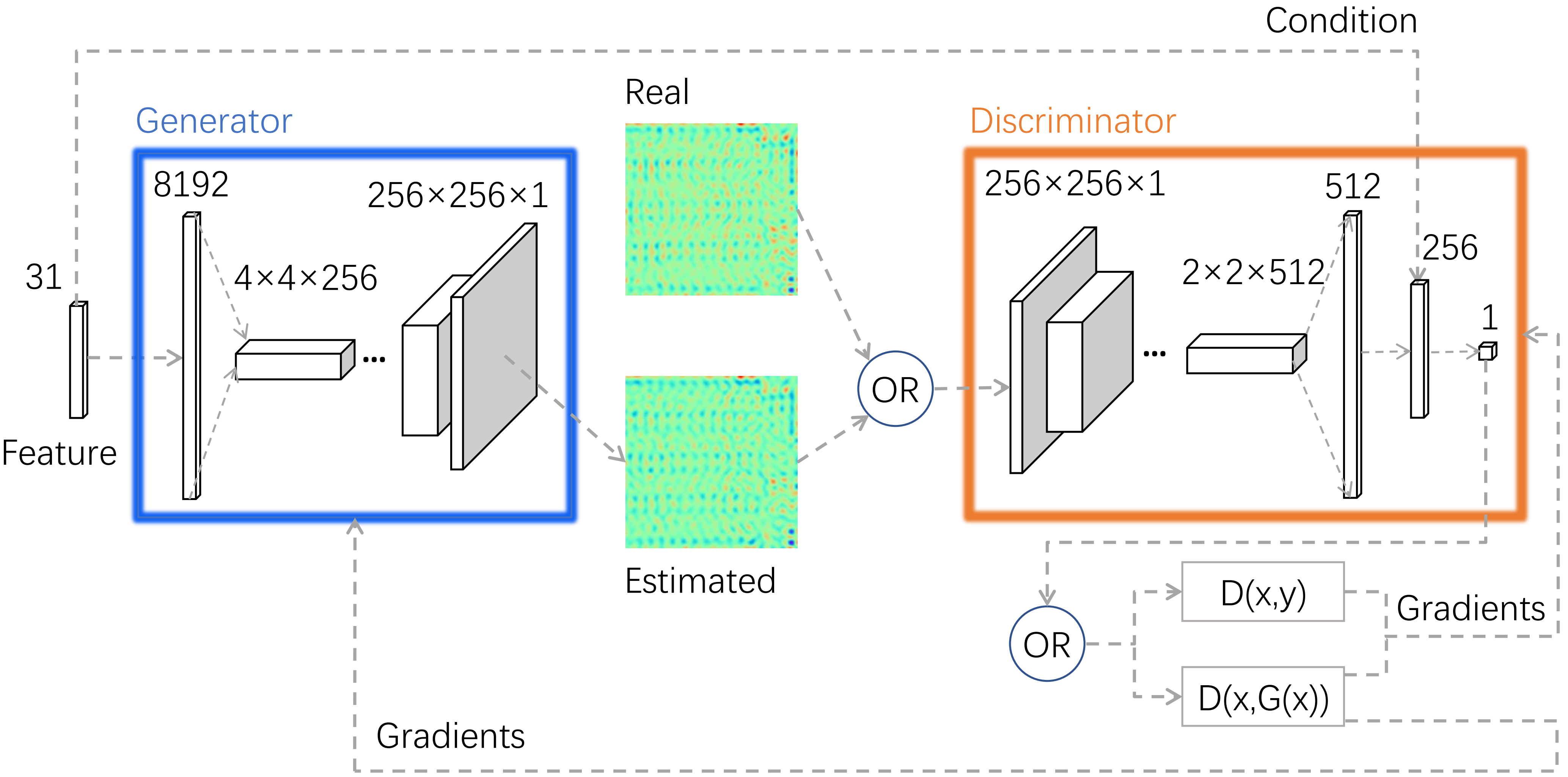


Power density

TPU Workloads selection

- Popular image recognition models
- EfficientNet, ResNet, MobileNet, …, etc.
- 7066 datapoints
- 6359 for training, 707 for testing

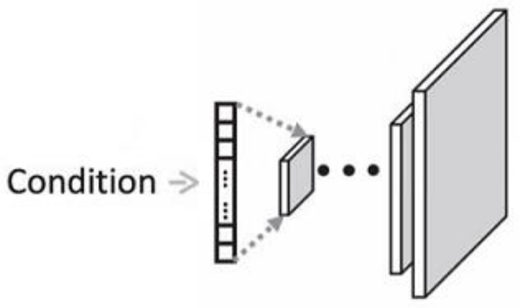
Conditional Generative Adversarial Network (CGAN)

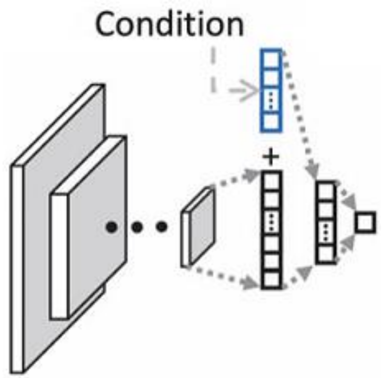


Generator and Discriminator

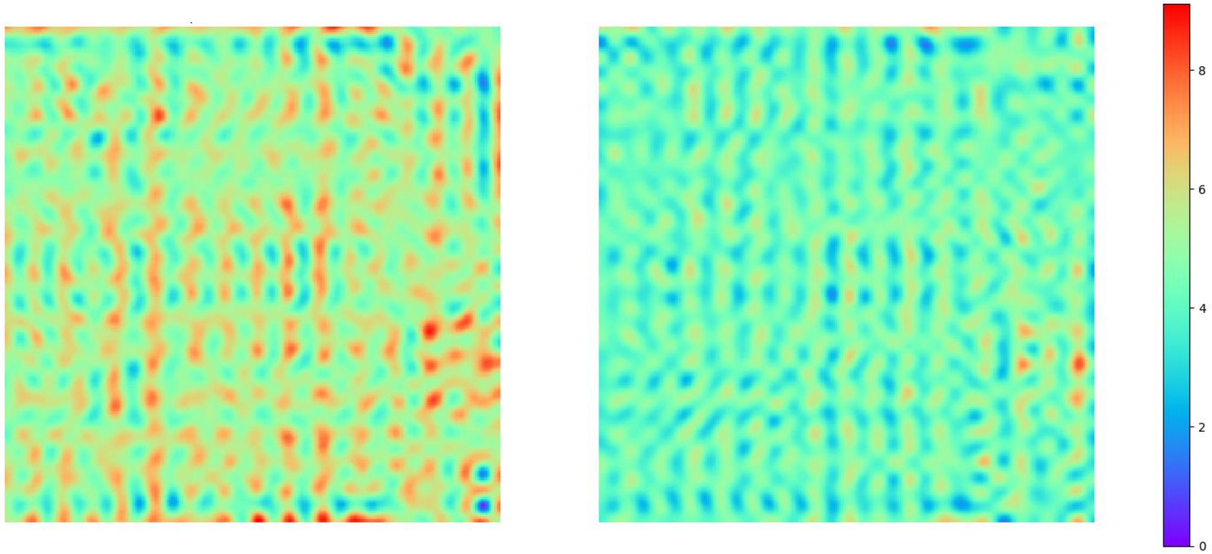
Generator			
Layer	Kernel	#Output	Activation
FC	-	8192	Leaky ReLU
Reshape	-	4x4x512	-
Conv_trans	5x5	8x8x512	Leaky ReLU
Conv_trans	5x5	16x16x512	Leaky ReLU
Conv_trans	5x5	32x32x256	Leaky ReLU
Conv_trans	5x5	64x64x128	Leaky ReLU
Conv_trans	5x5	128x128x64	Leaky ReLU
Conv_trans	5x5	256x256x1	-

Discriminator			
Layer	Kernel	#Output	Activation
Conv	5x5	128x128x64	Leaky ReLU
Conv	5x5	64x64x128	Leaky ReLU
Conv	5x5	32x32x256	Leaky ReLU
Conv	5x5	16x16x512	Leaky ReLU
Conv	5x5	8x8x512	Leaky ReLU
Conv	5x5	4x4x512	Leaky ReLU
Conv	5x5	2x2x512	Leaky ReLU
FC	-	512	Leaky ReLU
(+Cond)FC	-	256	Leaky ReLU
FC	-	1	-





Result metric



Root-Mean-Square Error (RMSE)

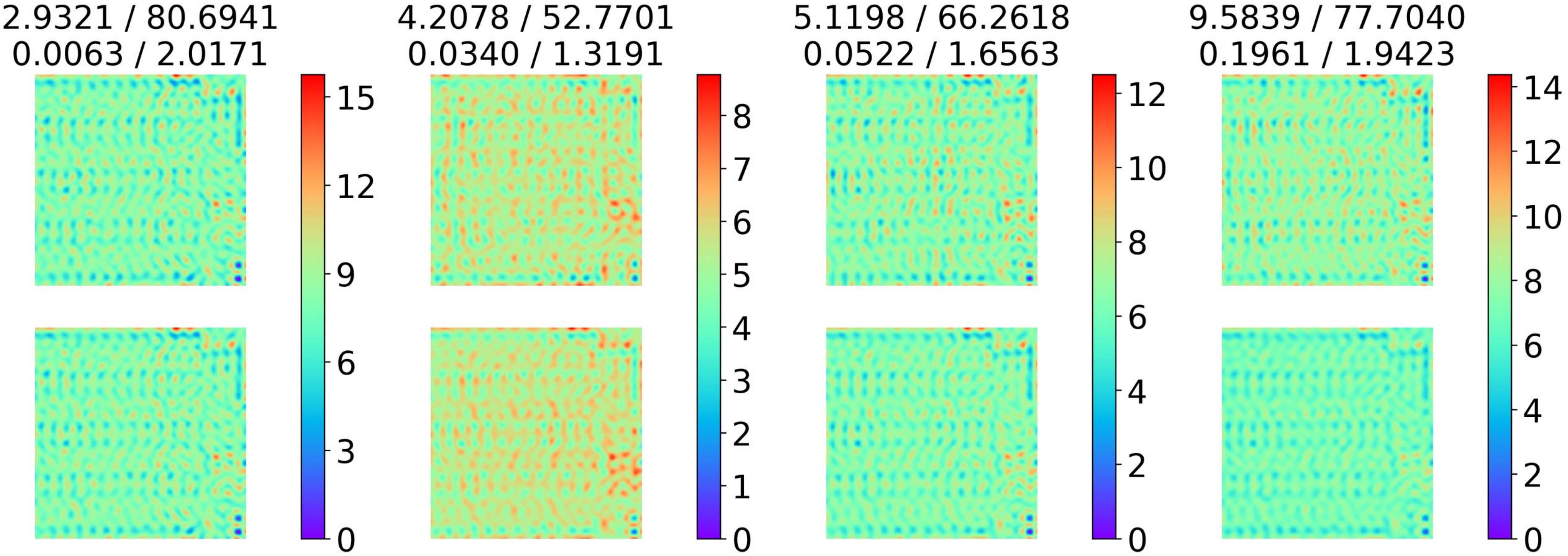
$$RMSE = \sqrt{\frac{\sum_{x=1}^{W}\sum_{y=1}^{H}[p(x,y) - p'(x,y)]^2}{W \times H}}$$

where p, p' are the ground truth and the predicted power map

H, W are the height and width of the map

Results & Discussion

Power Density RMSE | Average Power Density (unit: mW/mm^2)
Total Power Percentage Error | Total Power (unit: W)



- Row #1: Measured power density map (unit: W/cm^2=$10mW/mm^2$)
- Row #2: Estimated power density map

Result & Discussion

- Average RMSE of power density: 4.98 mW/mm^2
- Standard deviation: 2.53 mW/mm^2
- Power density range: 0 - 189.34 mW/mm^2
- Average inference time: 6.9ms on Intel Core i7-10710U
- Accurate and fast enough for the real-time power estimation

UC RIVERSIDE

Conclusion

- Proposed a machine-learning-based approach for real-time estimation of full-chip power maps for commercial Google Coral M.2 TPU chips for the first time.
- Experiment results show that the predictions are accurate, with the RMSE of only $4.98 mW/mm^2$, or 2.6% of the full-scale error.
- The inference speed on an Intel Core i7-10710U is as fast as 6.9ms, which is suitable for real-time estimation.