CLIO: Enabling automatic compilation of deep learning pipelines across IoT and Cloud

Authors:
Jin Huang, Colin Samplawski, Deepak Ganesan, Benjamin Marlin
Heesung Kwon

Context

- There have been significant advances in low-power neural accelerators(specialised processors) that aim to bring deep learning to IoT devices.

- However, state of the art deep learning models are often too large for low-power accelerators.

- This paper aims to solve the problem of running large deep learning models on highly resource constrained devices by partitioning the model and offloading computation to cloud.

Prevalent approaches

- Model compression - does not work for such devices (Requires 5x to 20x compression)
- Compressed data transmission to cloud - Use low power radio signals and are therefore sensitive to dynamics in env.
- Partition the model : Static partitioning does not account for dynamics in environment.

Clio’s approach

- Dynamic partitioning of model based on bandwidth available.
- Allows partial compilation of models on resource constrained devices
- **Progressive transmission :** Allows only some portion of the intermediate data output at a layer to be sent to the next layer
- .Essentially a compilation framework for deep learning models deployed on highly resource constrained devices.

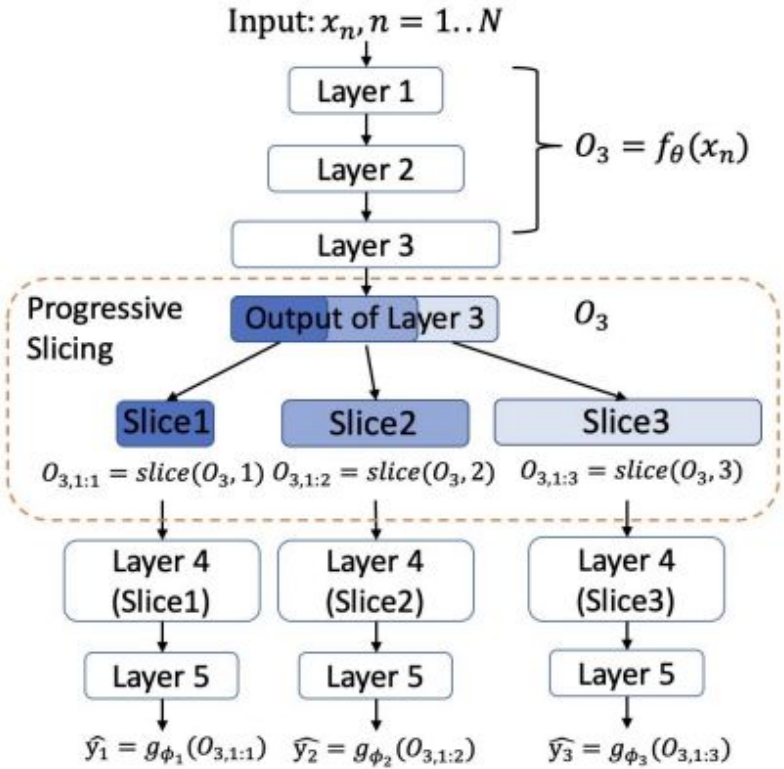
Device constraints

- Radio channels are set up as and when needed (to save power). So bandwidth is unknown when it wakes up and different wake periods can have different bandwidths.

- Computationally constrained : 1 GFlops, 64KB L1 cache, 512 KB L2 cache, freq of 175 MHz

- Devices considered in the paper : GAP8, STM32F7

CLIO Design - Progressive Slicing



From input x to model partition:

$$O_k = f_\theta(x)$$

Progressive Slicing:

$$O_{k,1:i} = slice(i,\ O_k)$$

Use slice as input:

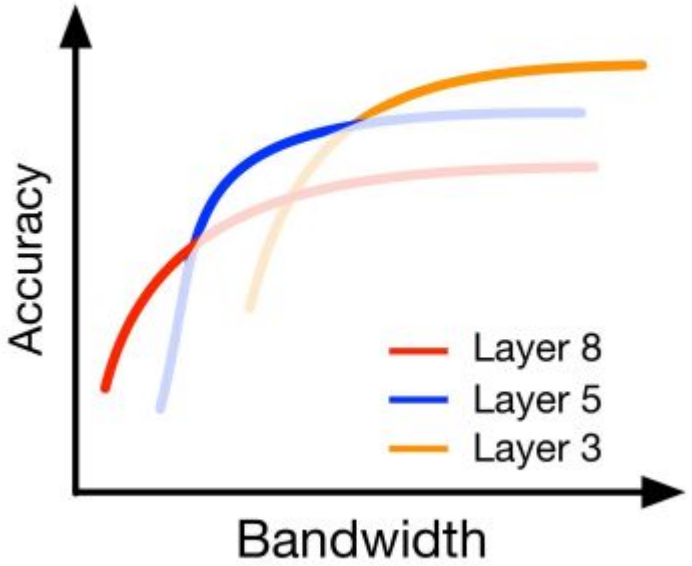
$$y = \phi_i(O_{1:i,k})$$

Objective Function:

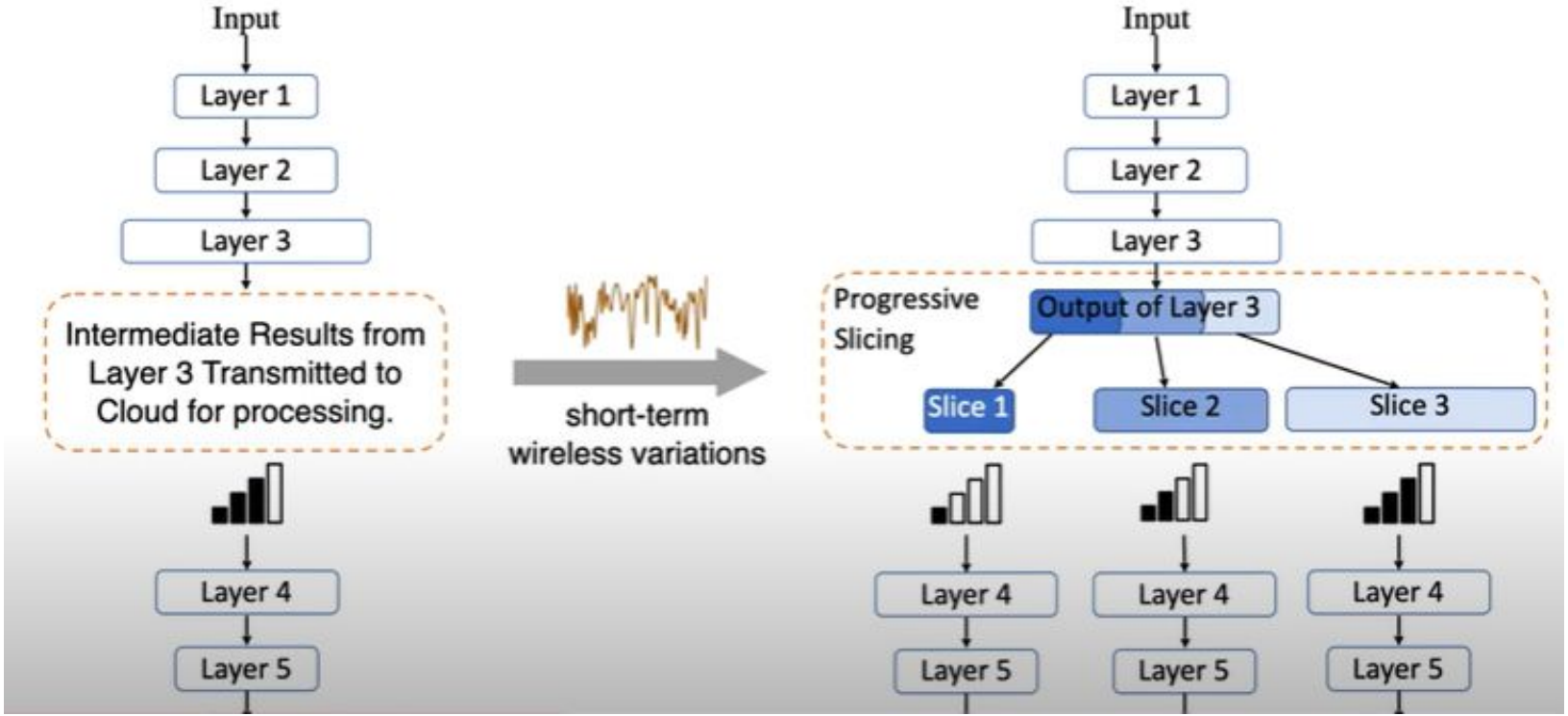
$$\mathcal{L}_i(D; \theta,\ \phi_i) = \sum_{n=1}^{N} \ell\left(y_n, g_{\phi_i}\left(slice\left(i,\ f_\theta(x_n)\right)\right)\right)$$

Adaptive partitioning

- Best operating point given a bandwidth can be identified by <partition, slice>



Partitioning + Progressive slicing



Reducing training time

- Training across all possible slices of the intermediate layer (for progressive transmission) and across all possible layers as potential partition points can be extremely time consuming.

- Training time may be expressed as:

$$C(\mathbf{M}) = C(\mathbf{L1{:}k}) + \#\mathbf{slices} \times C(\mathbf{Lk{+}1{:}K})$$

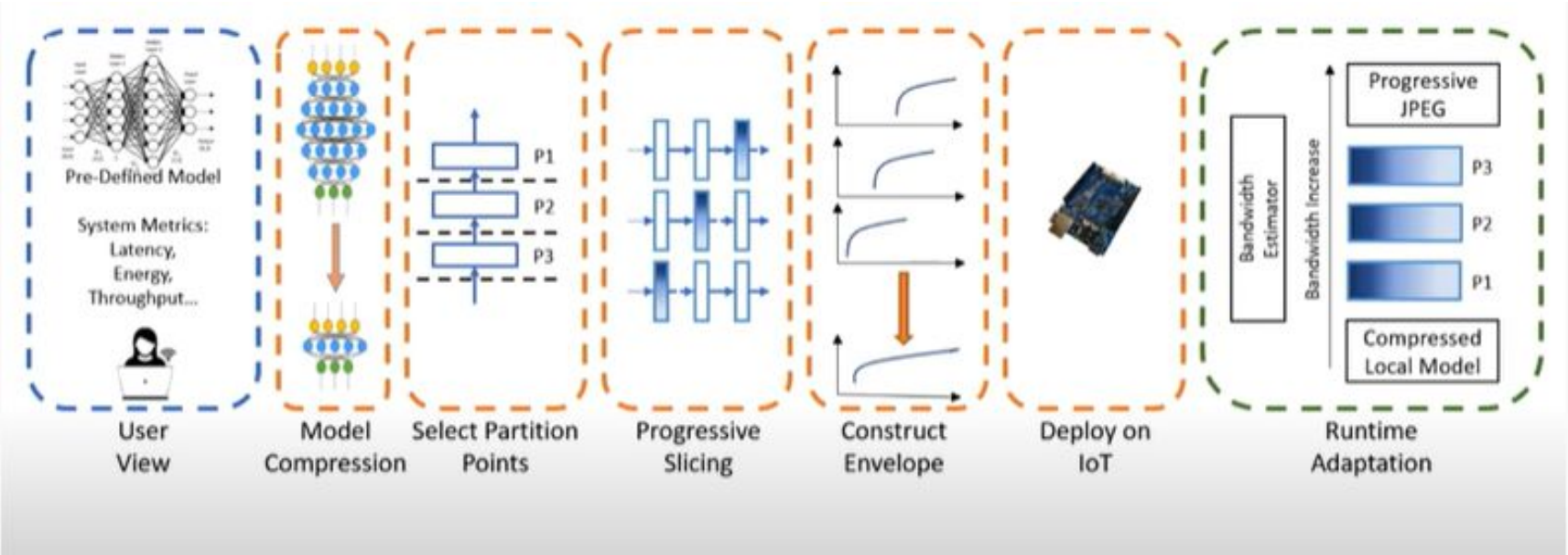
- Clio provides methods to help reduce number of partitions and number of possible slices.

Reducing training time (Contd.)

- For progressive slicing :
  - Only have consider number of channels that are multiples of some number
  - Also we observe that for every slice only the next layer needs to be different, layers afterward can use the same model.

- Reducing potential partition points:
  - Given a target latency, and a layer k.
  - We know c(k) = C(L 1:k) + C_cloud(L:k+1,end)
  - Then transmission time allowed : target - c(k)
  - From this we can determine range of bandwidth at this layer that can meet the conditions.

Implementation

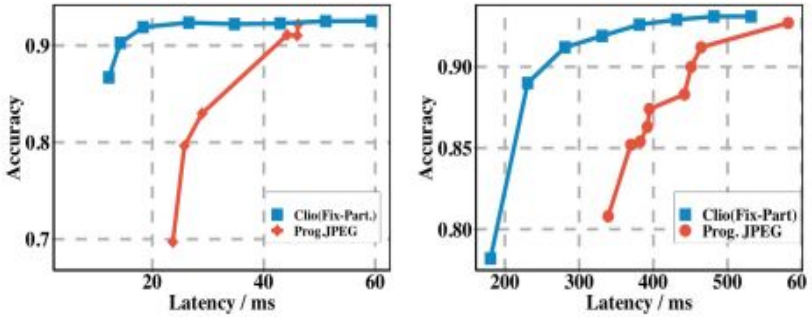


Evaluation

- Dataset : CIFAR-10 and modified Imagenet

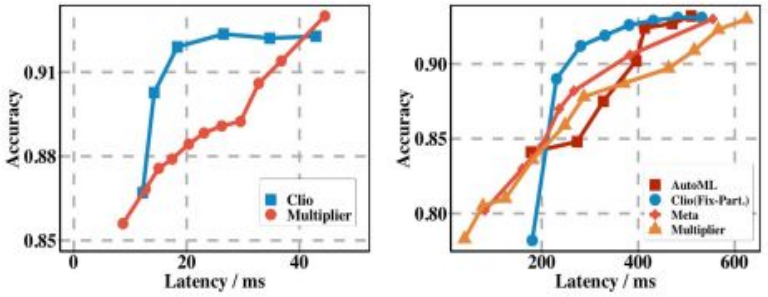
- Comparisons:
  - Progressive Slicing : Compared against Prog. JPEG
  - Model compression : Compared against Auto ML, Meta etc
  - Comparing Local, Remote and Clio

Progression Slicing and Model compression



(a) CIFAR-10 (b) ImageNet-20

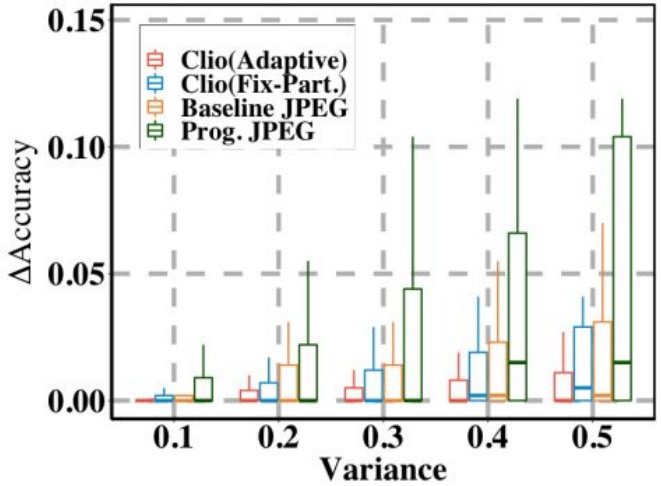
Figure 6: Prog. Slicing vs Prog. JPEG for CIFAR-10 and ImageNet-20 datasets. Latency corresponds to GAP8 and BLE radio (numbers in §4.2). Prog. Slicing has significantly better performance than Prog. JPEG for both CIFAR10 and ImageNet-20 datasets.

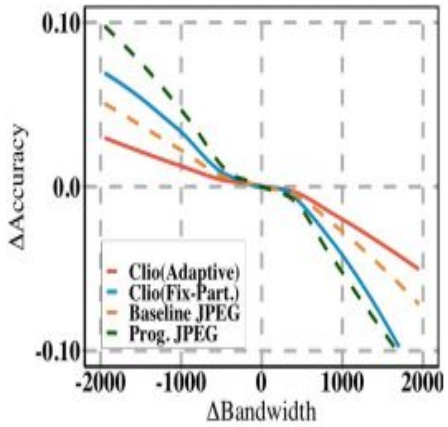


(a) CIFAR-10 (b) ImageNet-20

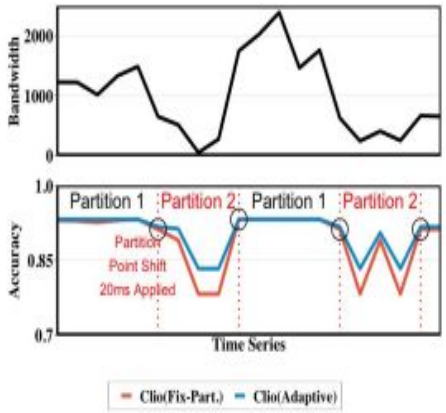
Figure 7: Clio vs Model Compression. Latency corresponds to GAP8 and BLE radio @ 250kbps. On CIFAR10, only width multipliers are effective and Clio is better than this method. On ImageNet-20, Clio is better than AutoML, meta-pruning and width multipliers except at low latencies where there is insufficient time for communication.

Results (contd)





(a) Effect of estimation error



(b) Time Series

Figure 9: We use synthetic traces and gradually increase variance in bandwidth to show robustness of different methods. Clio with adaptive partitioning is most robust to bandwidth variations and degrades more gracefully than Baseline JPEG.

Discussion

- Compilation of networks where blocks of layers may have logical significance is not handled.
- Greedy method to select partitioning layers should not just depend on bandwidth.
- At low latency requirements, model compression seems to work better.