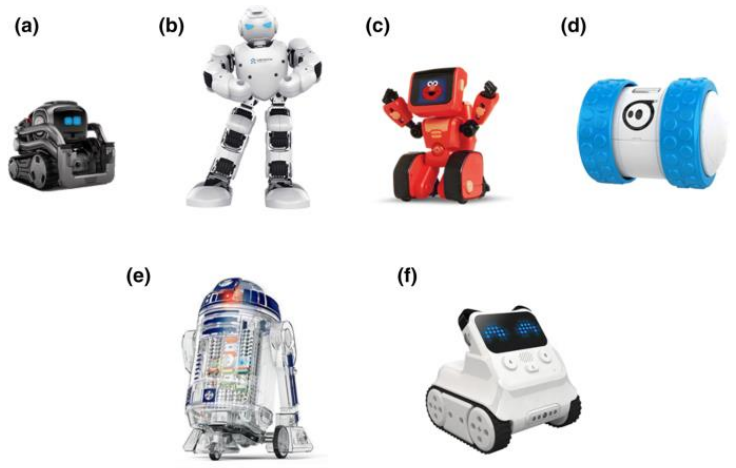
Real-time Neural Network Inference on Extremely Weak Devices: Agile Offloading with Explainable AI

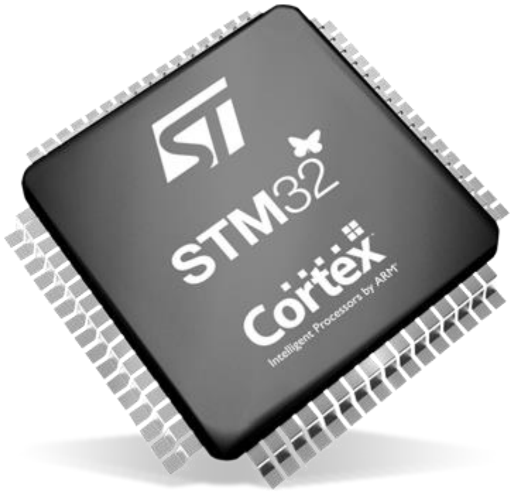
Kai Huang, Wei Gao
University of Pittsburgh, USA

Real-time NN demands on weak embedded devices



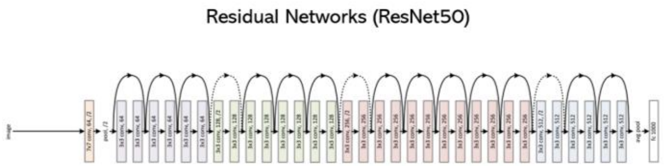
Resources limitation on weak embedded devices

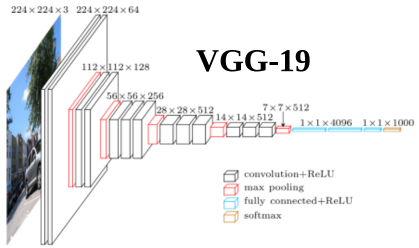
Weak Embedded Devices



<1MB memory and storage
16~216 MHz CPU

Large Neural Networks





Require >100 MB of memory space
>2 GHz CPU for 60 ms latency

Existing Work

- **Local Inference**
  - Prune, Compress, Network Architecture Search
  - Lead to oversimplified NN structures
  - >10% accuracy lost
- Remote Inference
  - Compress raw data before transmission
  - Limited data compressibility when accuracy loss is minimum.
- NN offloading
  - Use a local NN to sparsity & compress data
  - Higher compressibility but expensive local NNs
  - Extra transmission cost

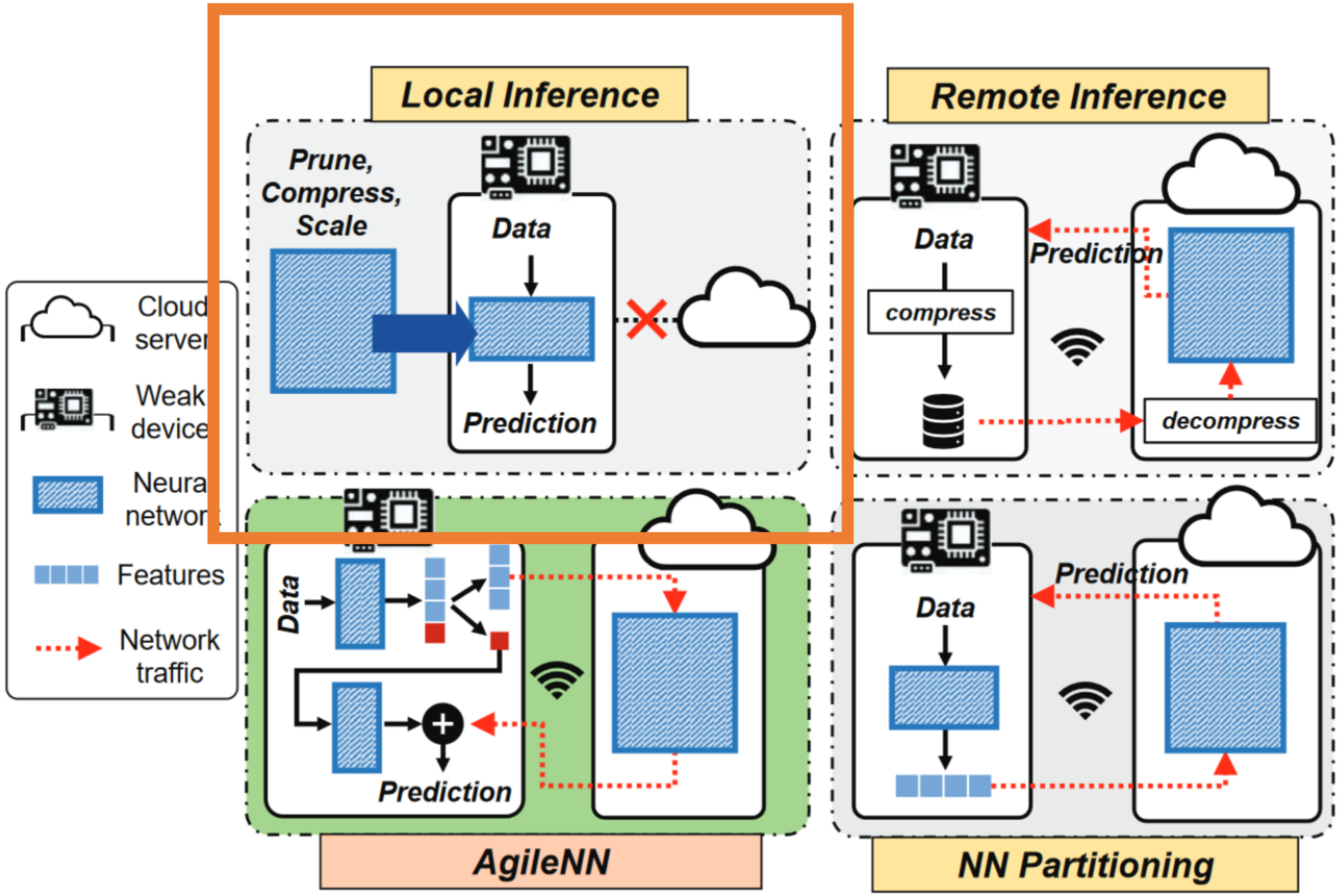


Figure 1: Existing work vs. AgileNN

Existing Work

- Local Inference
  - Prune, Compress, Network Architecture Search
  - Lead to oversimplified NN structures
  - >10% accuracy lost
- Remote Inference
  - Compress raw data before transmission
  - Limited data compressibility when accuracy loss is minimum.
- NN offloading
  - Use a local NN to sparsity & compress data
  - Higher compressibility but expensive local NNs
  - Extra transmission cost

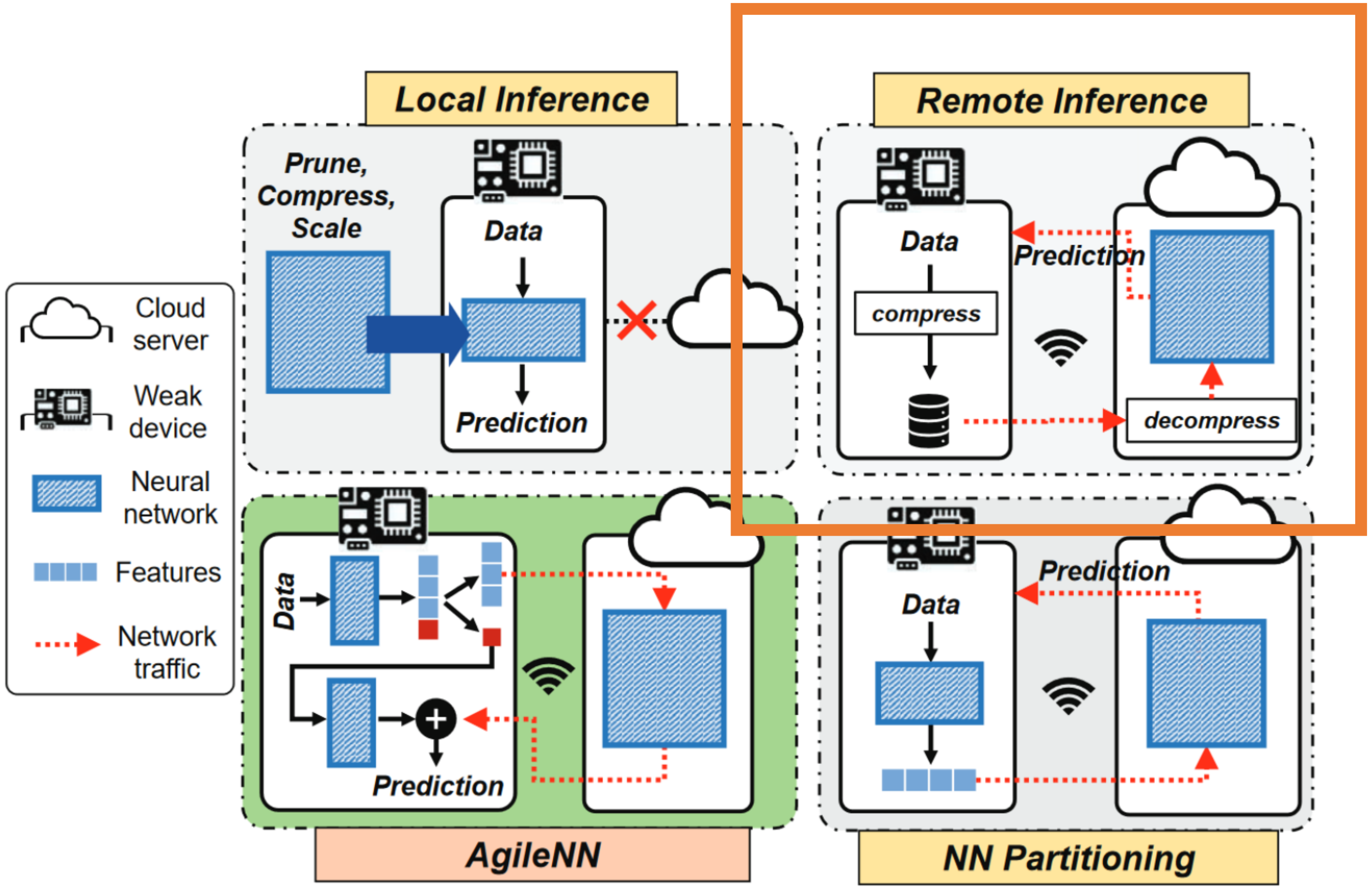


Figure 1: Existing work vs. AgileNN

Existing Work

- Local Inference
  - Prune, Compress, Network Architecture Search
  - Lead to oversimplified NN structures
  - >10% accuracy lost
- Remote Inference
  - Compress raw data before transmission
  - Limited data compressibility when accuracy loss is minimum.
- **NN offloading**
  - **Use a local NN to sparsity & compress data**
  - **Higher compressibility but expensive local NNs**

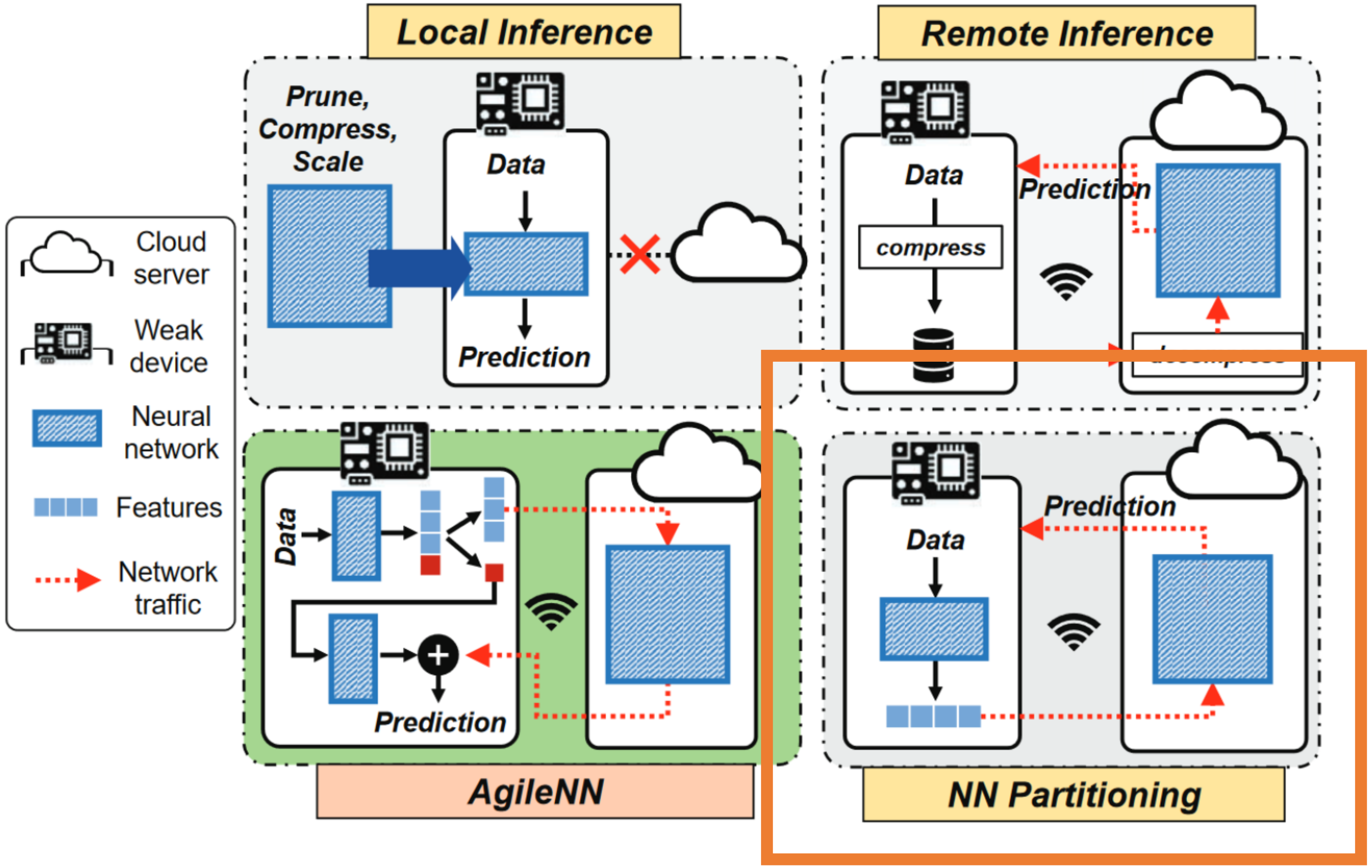
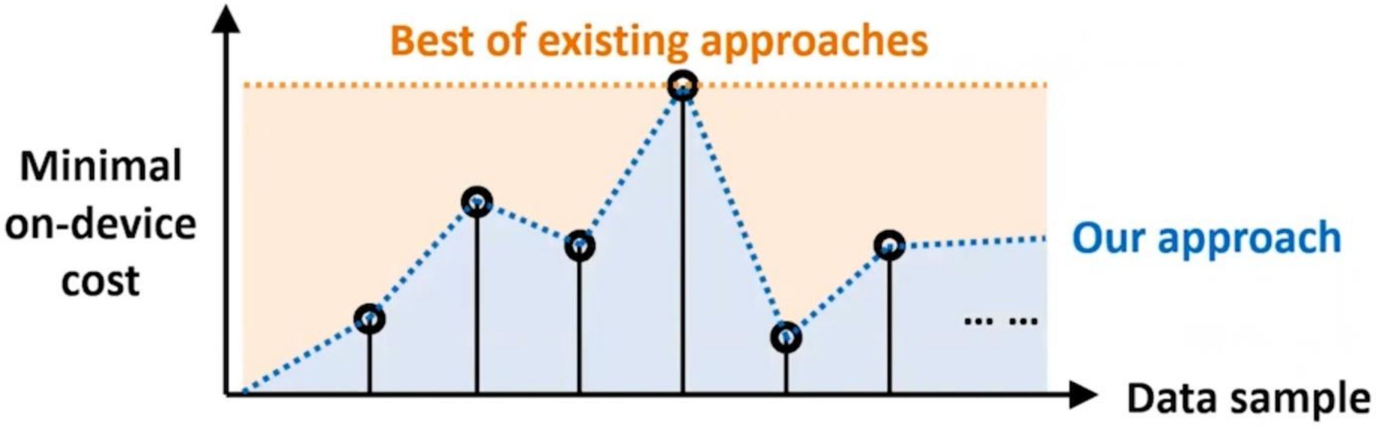


Figure 1: Existing work vs. AgileNN

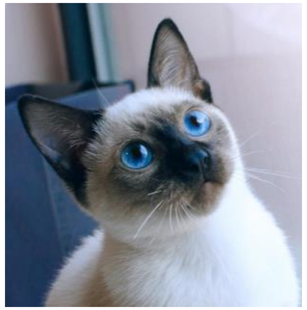
AgileNN : From fixed to data-centric and agile

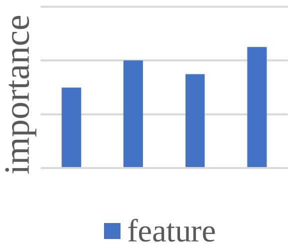


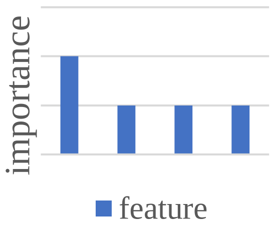
Data-centric
Incorporate the knowledge about different input data's **heterogeneity** in training

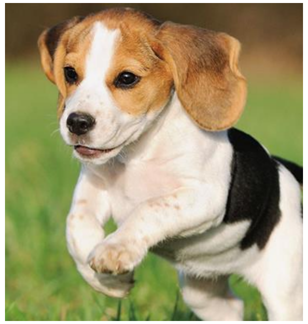
Agile Offloading
Adaptive partition to minimize the offloading cost

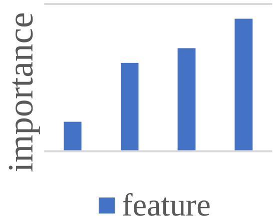
AgileNN : From fixed to data-centric and agile









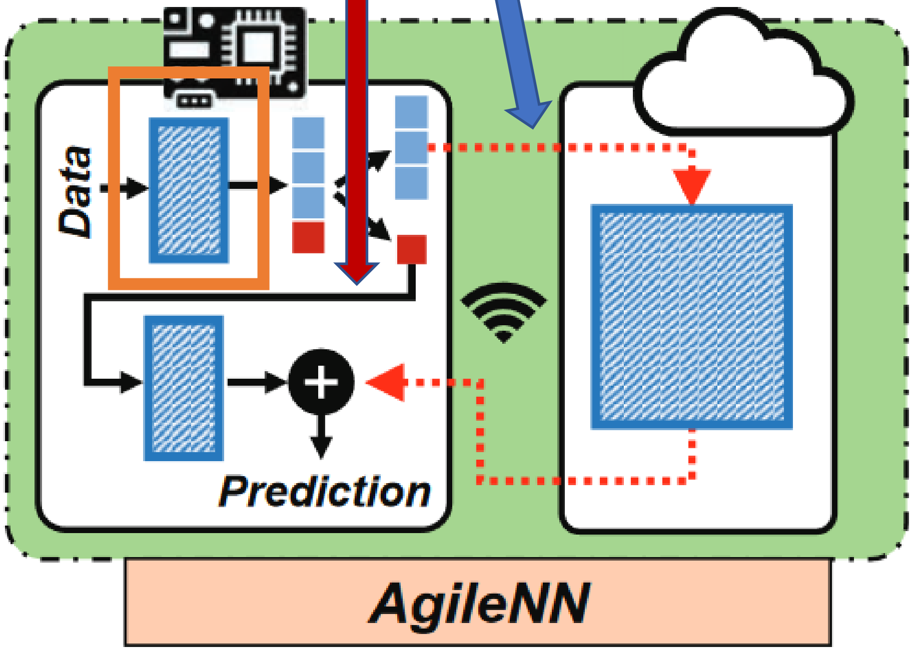


Feature Extractor

For every Input data:

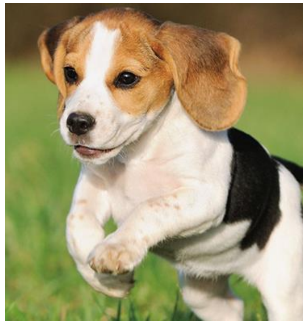
TopK important features – local NN

Less important features – remote NN

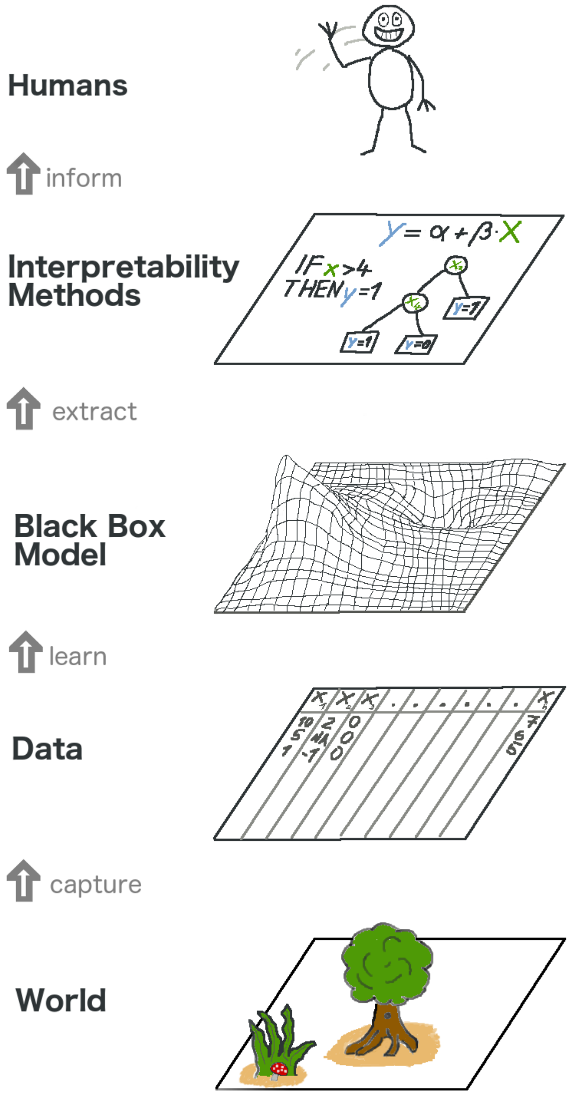


A Summary about Interpretable ML

- **Interpretability**
  - **the degree to which a human can understand the cause of a decision.**



Why it's a dog?



XAI Tools - Integrated Gradients

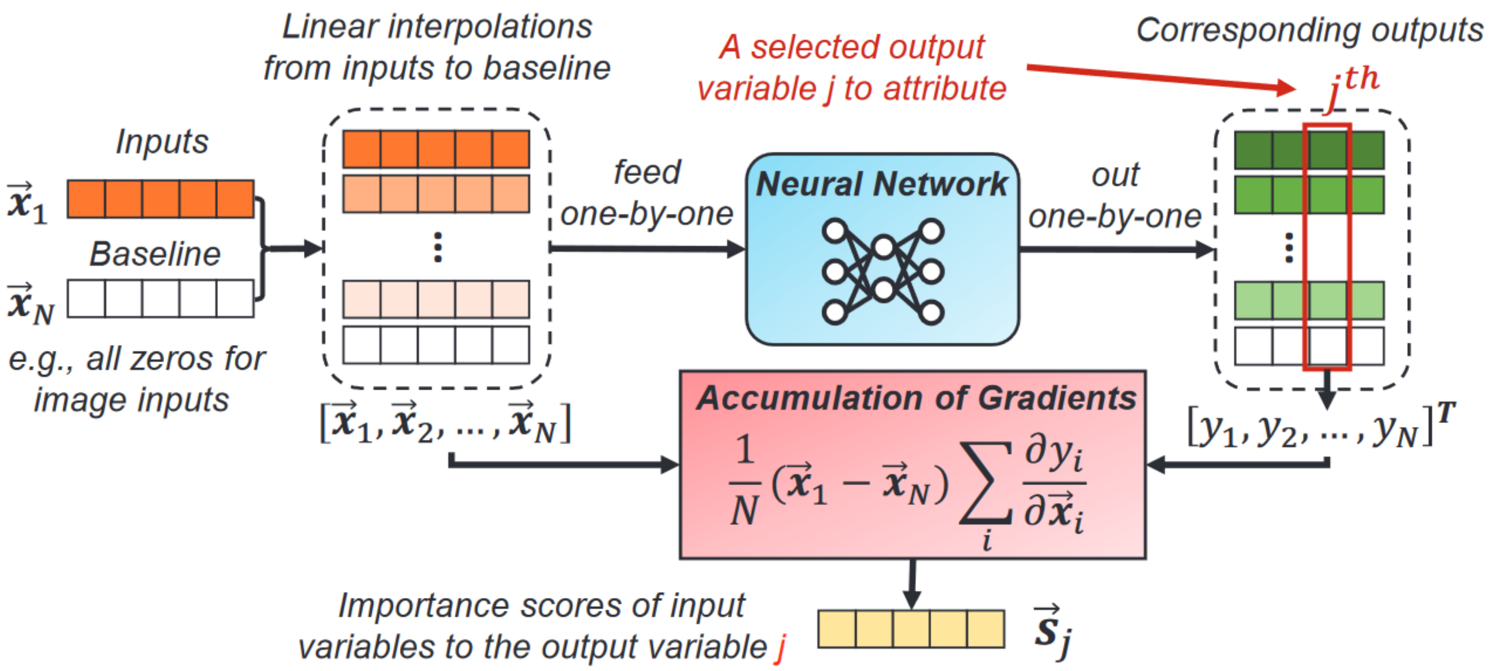
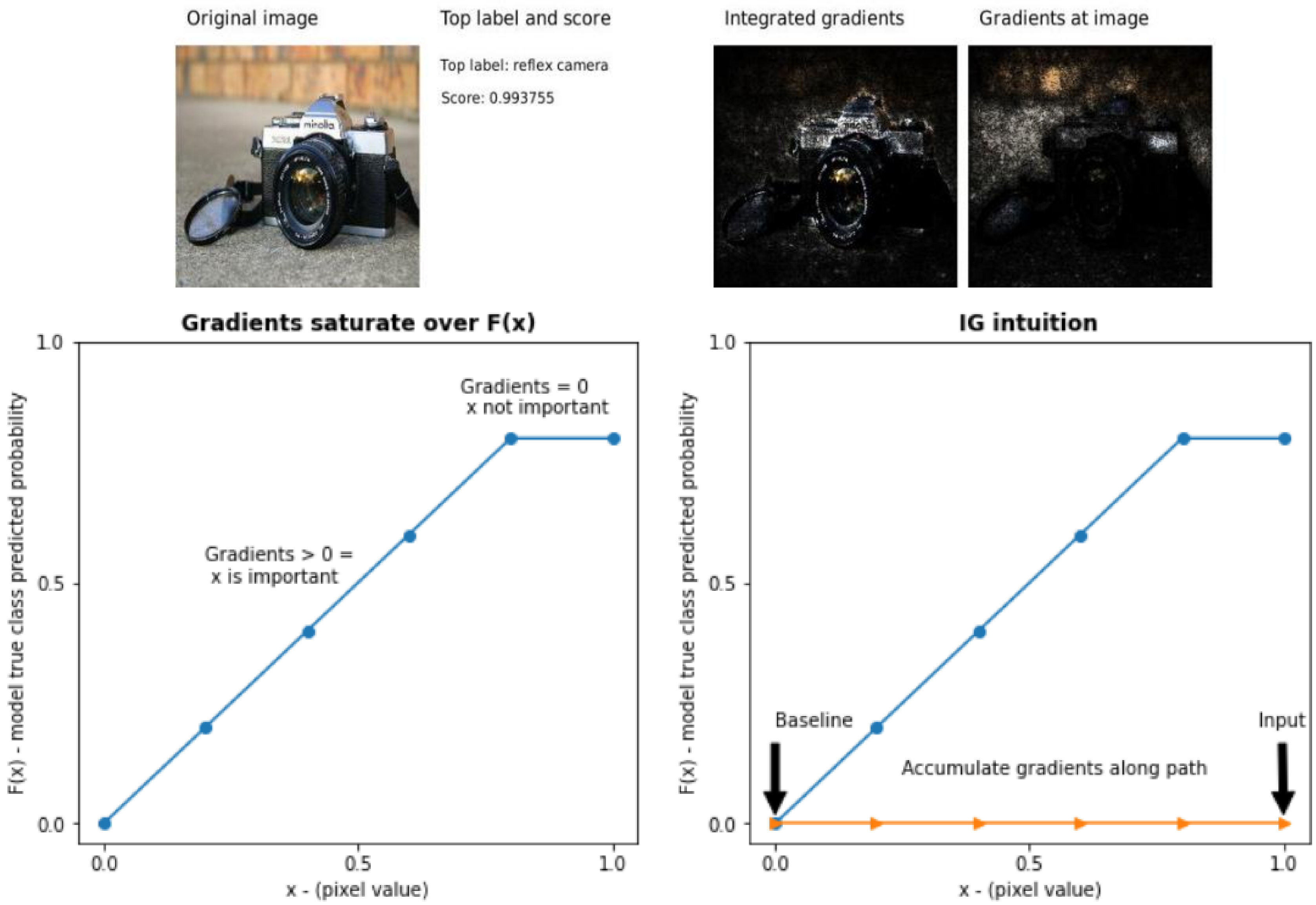


Figure 3: Integrated Gradients

Integrated Gradients aims to explain the relationship between the model predictions and its features.

XAI Tools - Integrated Gradients



XAI Tools - Integrated Gradients

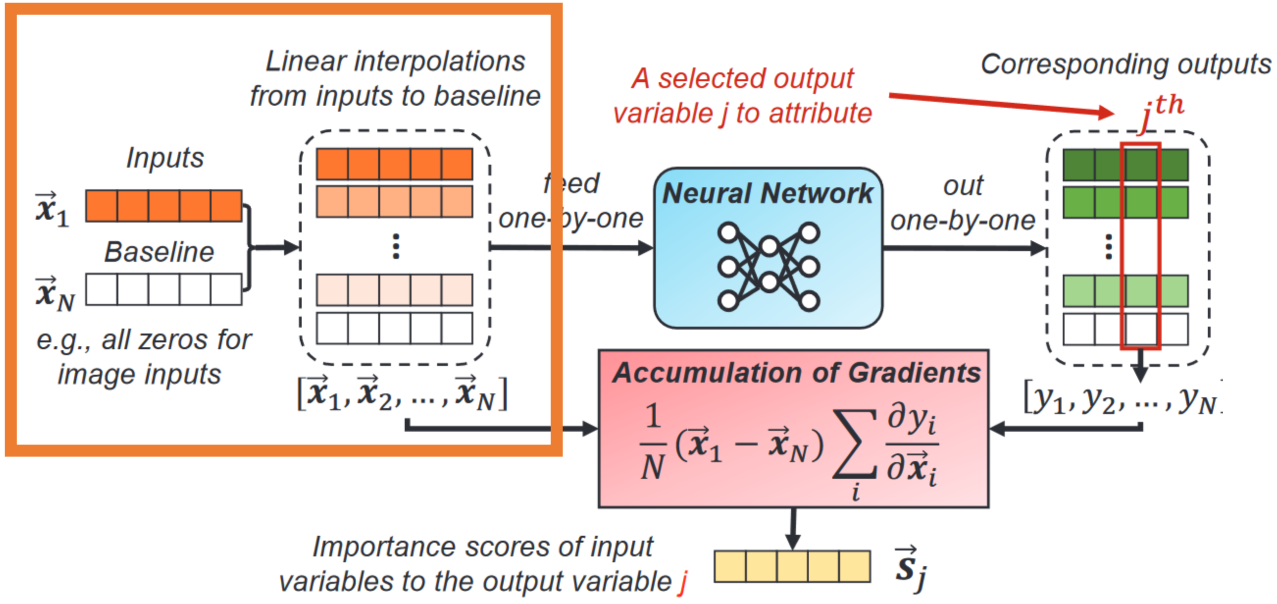
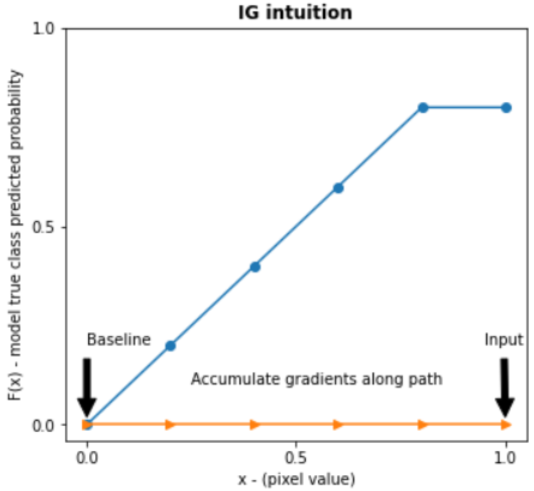


Figure 3: Integrated Gradients

1. Generate alphas α
2. Generate interpolated images = $(x' + \frac{k}{m} \times (x - x'))$



Integrated Gradients aims to explain the relationship between the model predictions and its features.

XAI Tools - Integrated Gradients

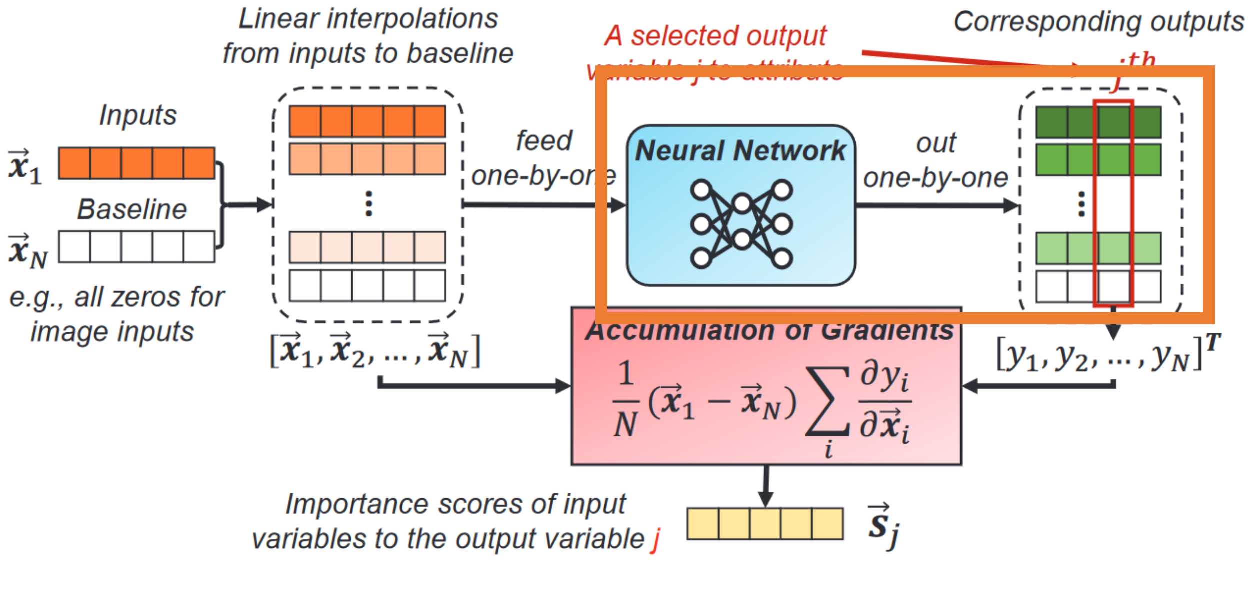
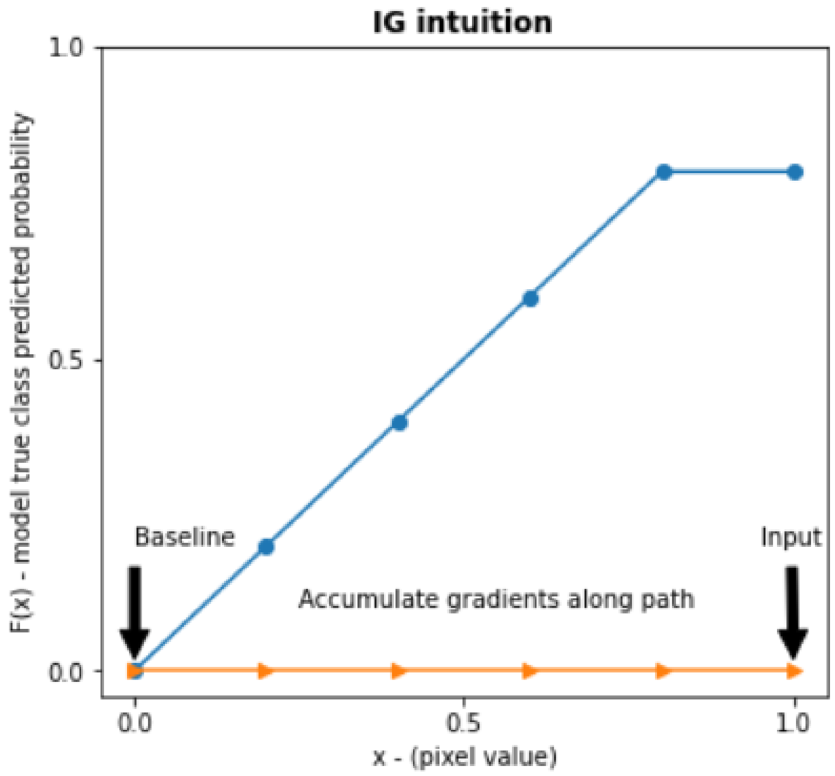


Figure 3: Integrated Gradients



3. Compute gradients between model F output predictions with respect to input features = $\frac{\partial F(\text{interpolated path inputs})}{\partial x_i}$

XAI Tools - Integrated Gradients

$$IntegratedGrads_i^{approx}(x) ::= (x_i - x'_i) \times \overbrace{\textstyle\sum_{k=1}^{m}}^{\text{Sum m local gradients}} \text{gradients(interpolated images)} \times \overbrace{\frac{1}{m}}^{\text{Divide by m steps}}$$

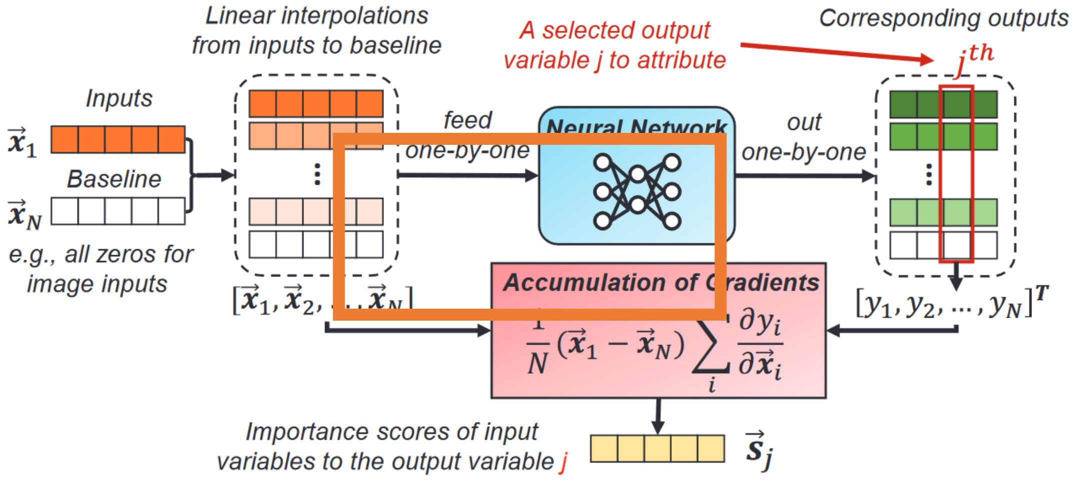


Figure 3: Integrated Gradients

XAI Tools - Integrated Gradients

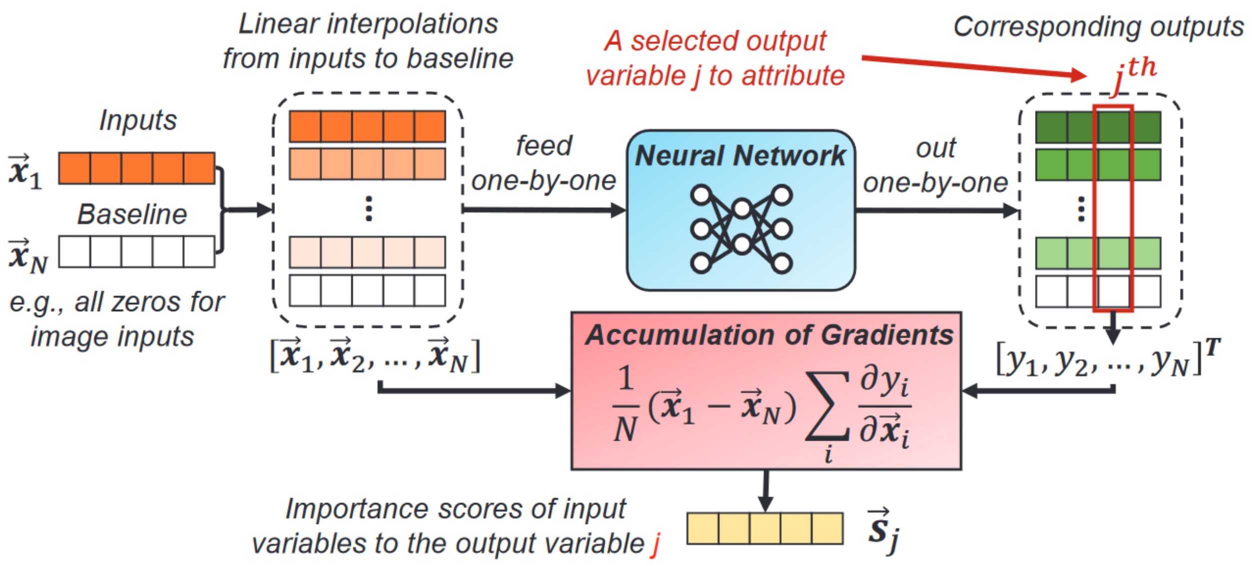
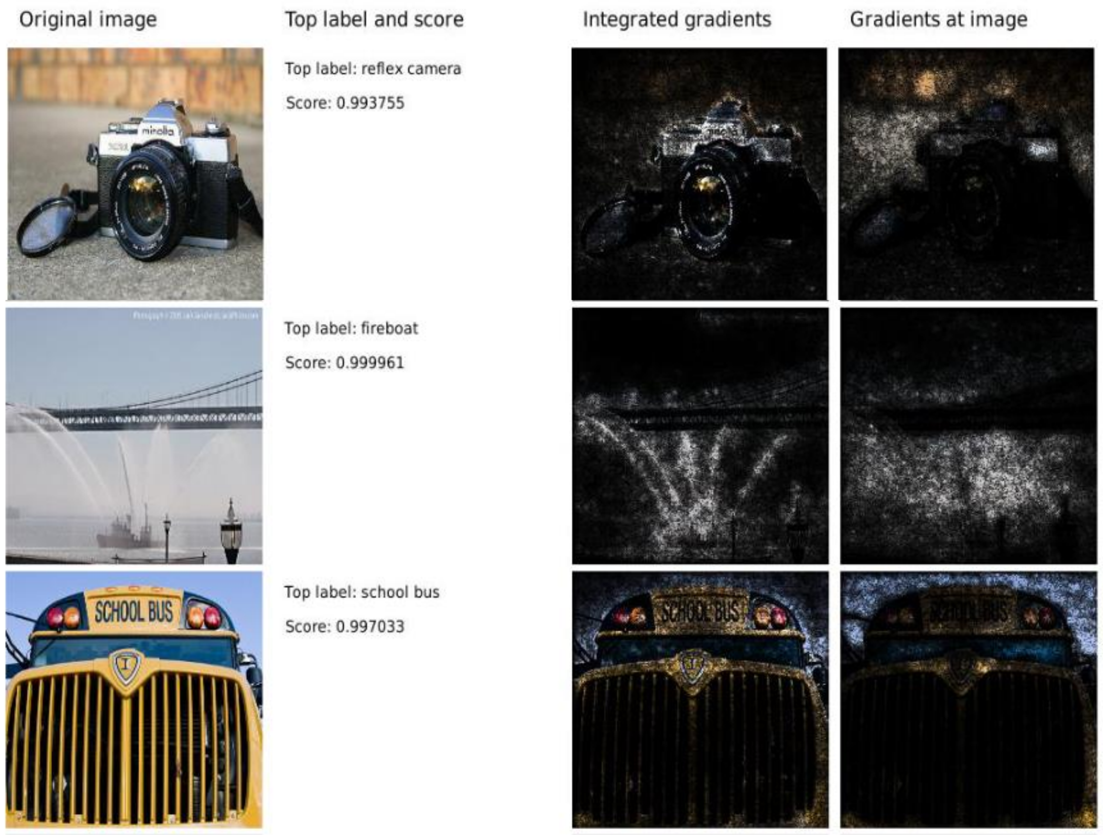


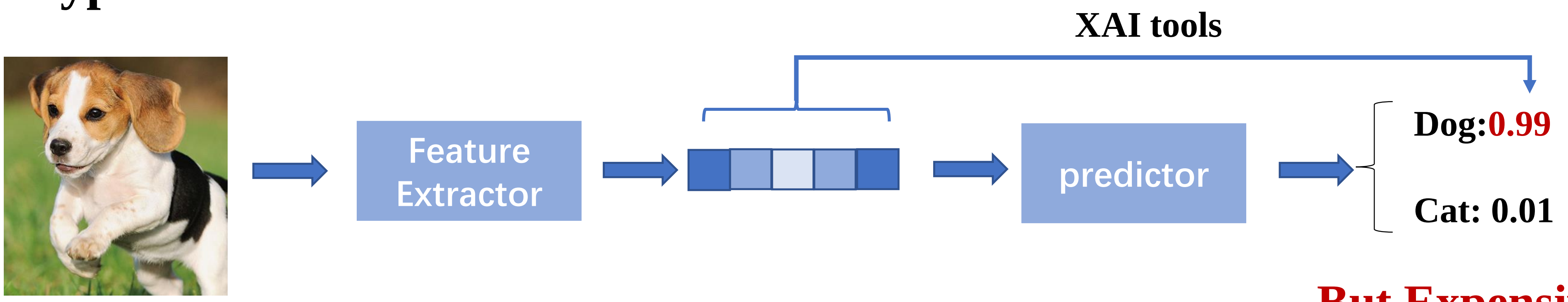
Figure 3: Integrated Gradients



Integrated Gradients aims to explain the relationship between the model predictions and its features.

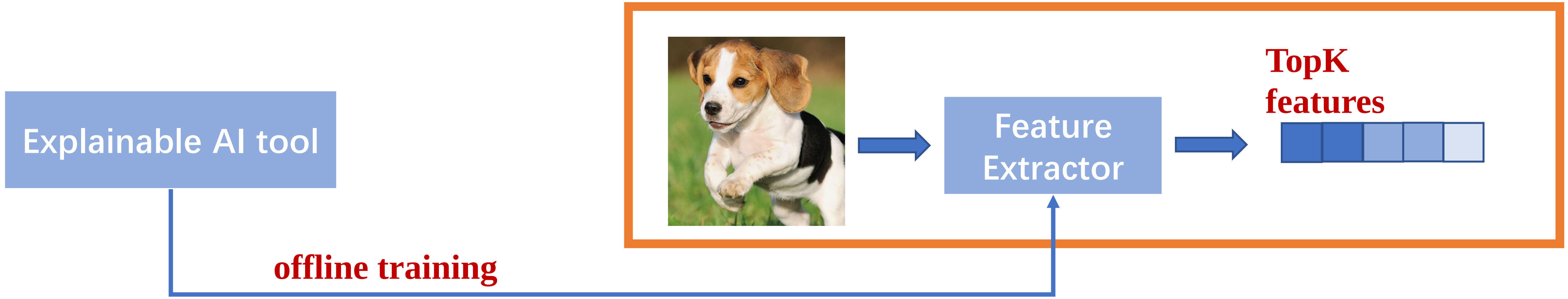
XAI-enabled feature extractor

A typical XAI workflow in NN inference

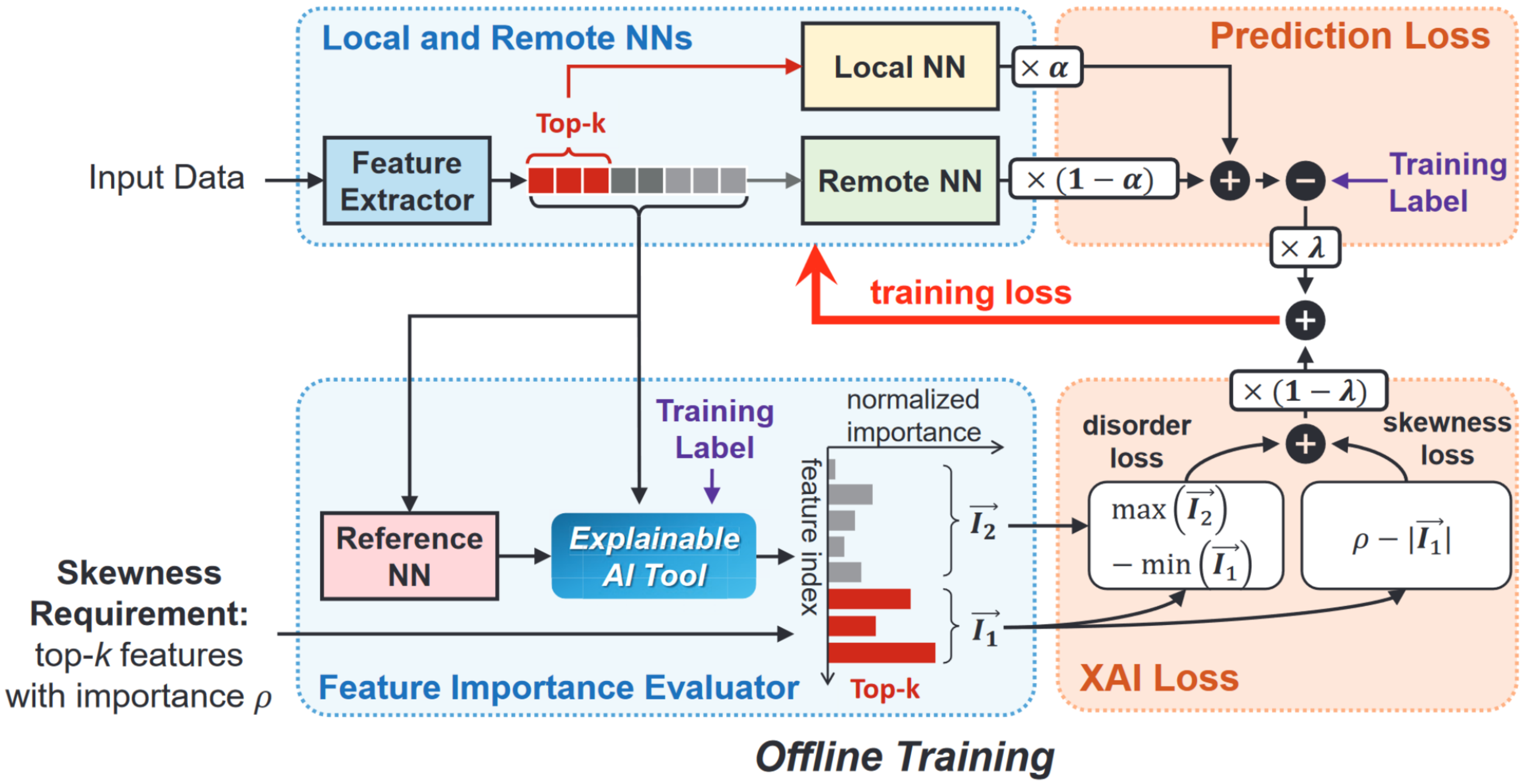


But Expensive!!

Instead, we suggest to address this challenge via agile NN offloading, which migrates the required computations in NN offloading from online inference to offline learning.



AgileNN : Offline Training



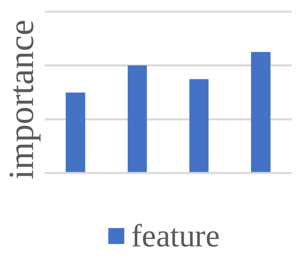
XAI Tools - Integrated Gradients

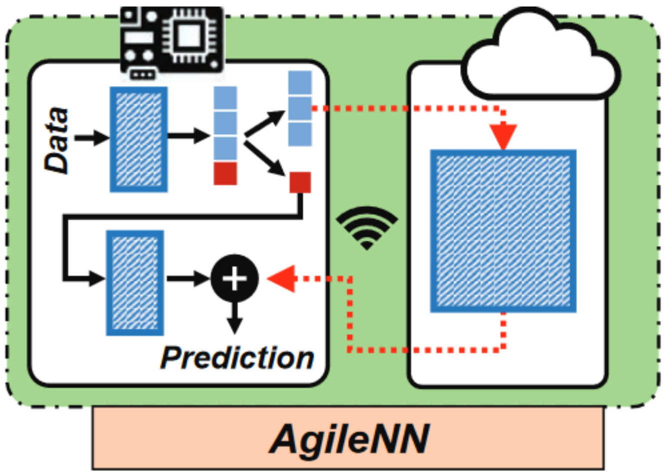
- **Consider 1**

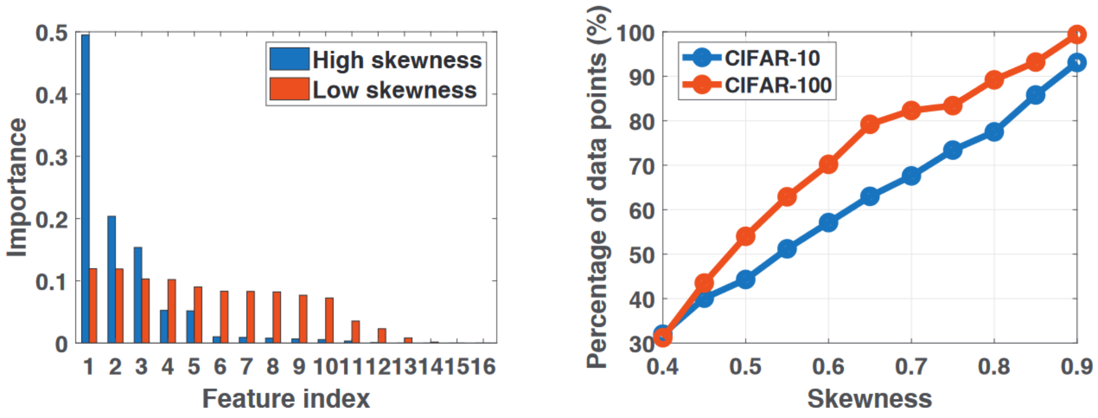
Skewness may not always exist in every input data.

- **Consider 2**

The accuracy of feature importance evaluation builds on accurate NN inference in advance.



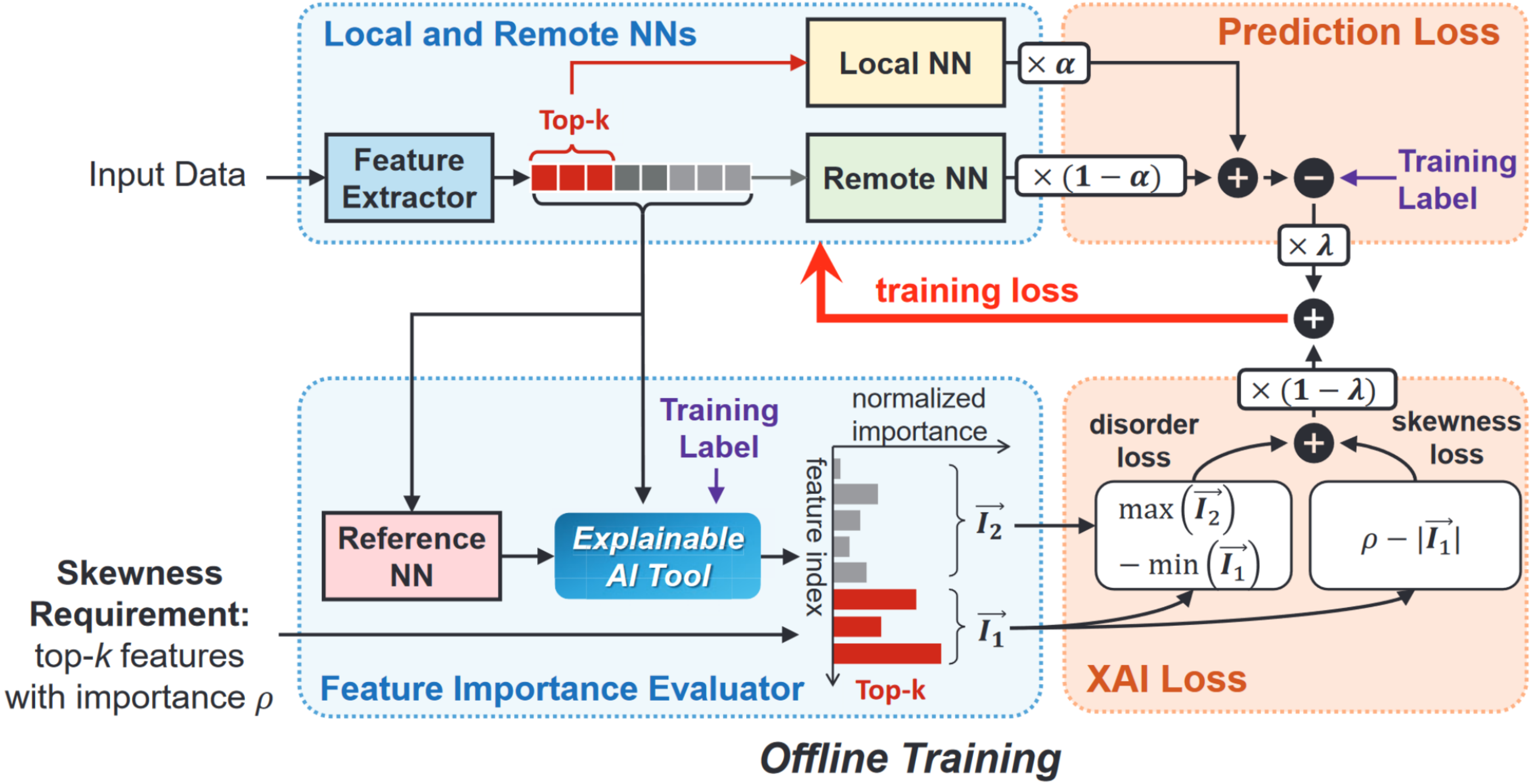




(a) Different levels of skewness (b) Skewness CDF

Figure 4: Skewness of feature importance. Skewness is measured as the normalized importance of the top 20% features, using the MobileNetV2 model [55].

AgileNN : Offline Training



Enforce skewed distribution of features

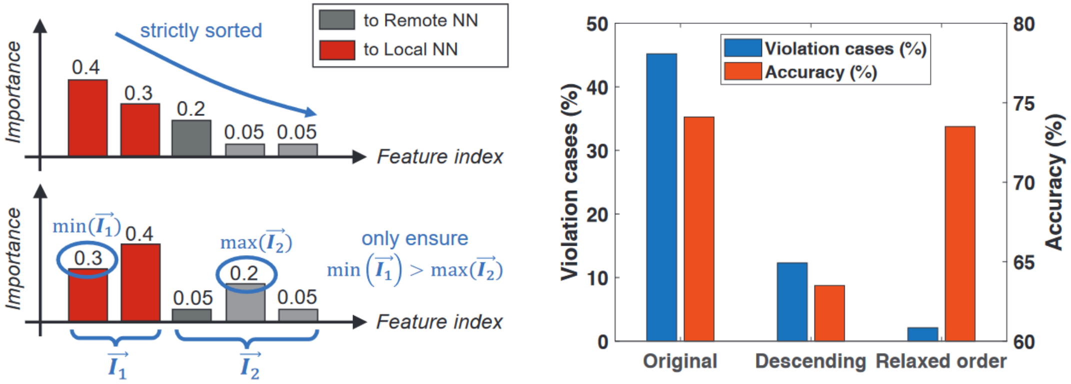
Disorder loss

Ensure topmost important features are extracted into **topK channels**

$$L_{\text{disorder}} = \max\left(0, \max(\vec{I}_2) - \min(\vec{I}_1)\right)$$

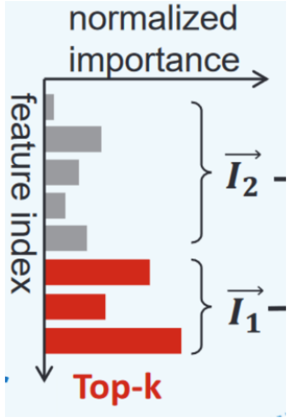
Skewness Loss

Enhance the important of topK features to **ensure compressibility** of the others



(a) Different feature orders (b) Effectiveness of reordering

Figure 9: Feature reordering



Enforce skewed distribution of features

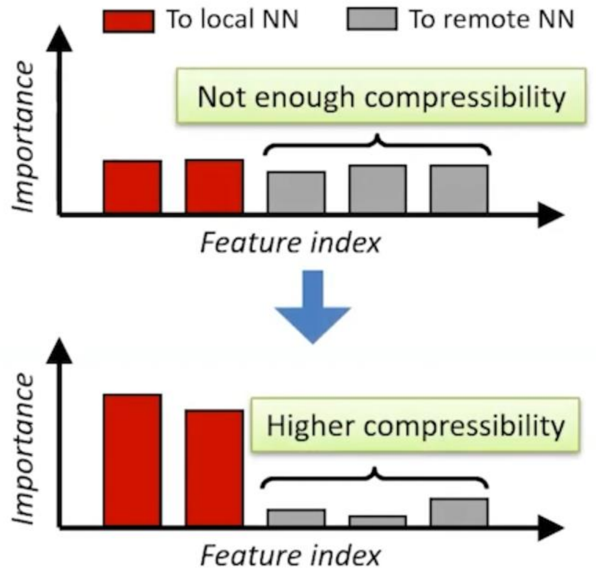
Disorder loss

Ensure topmost important features are extracted into **topK channels**

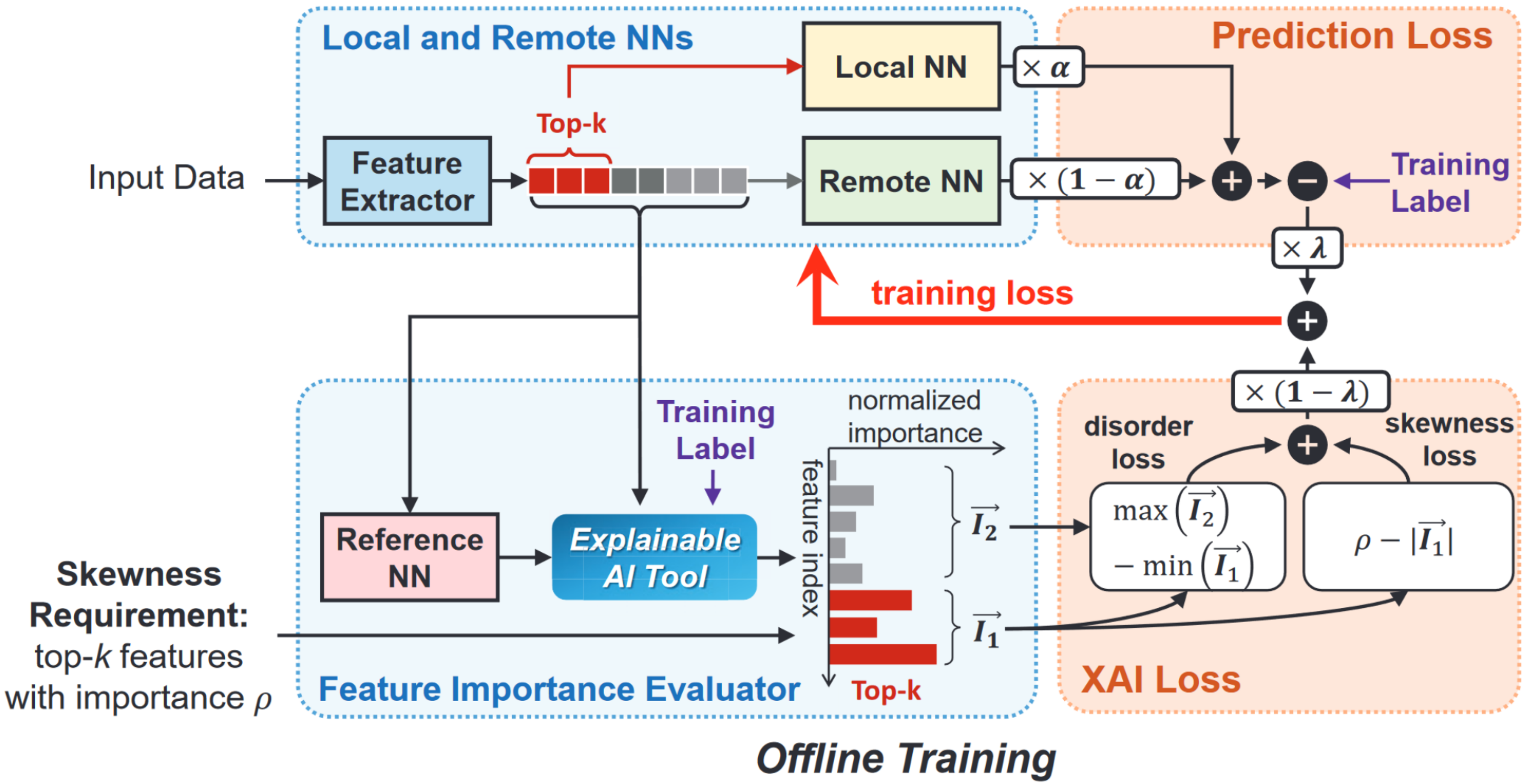
Skewness Loss

Enhance the important of topK features to **ensure compressibility** of the others

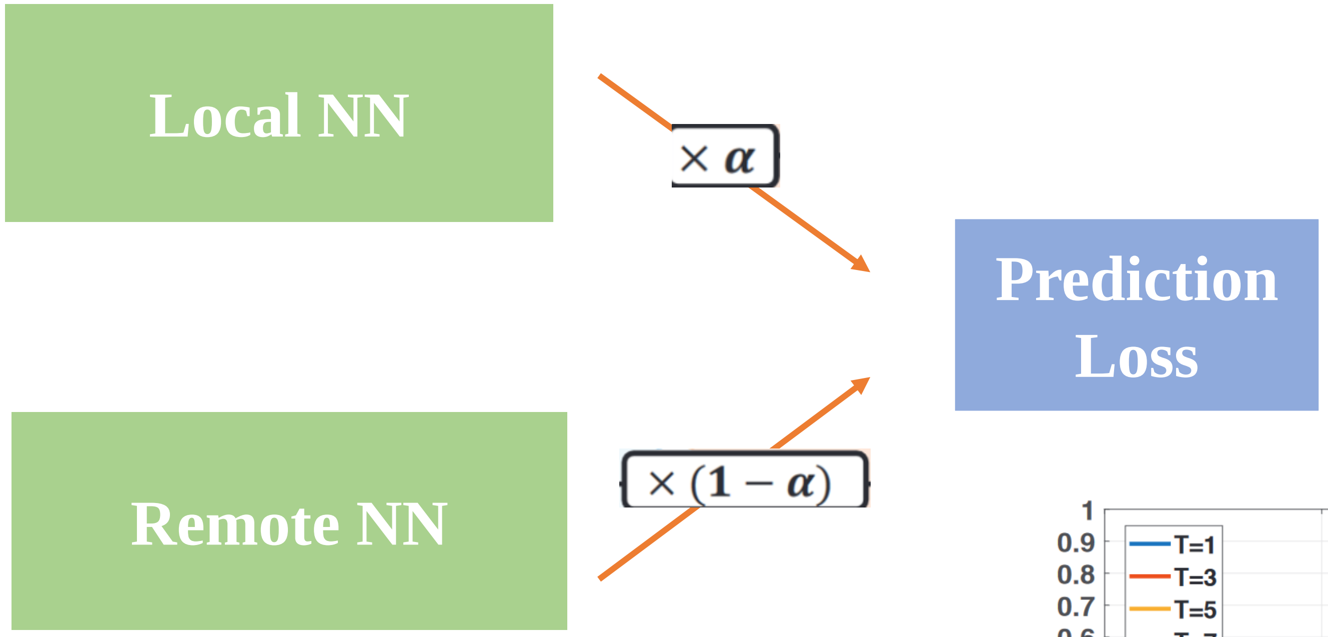
$$L_{\text{skewness}} = \max\left(0, \rho - |\vec{I_1}|\right)$$



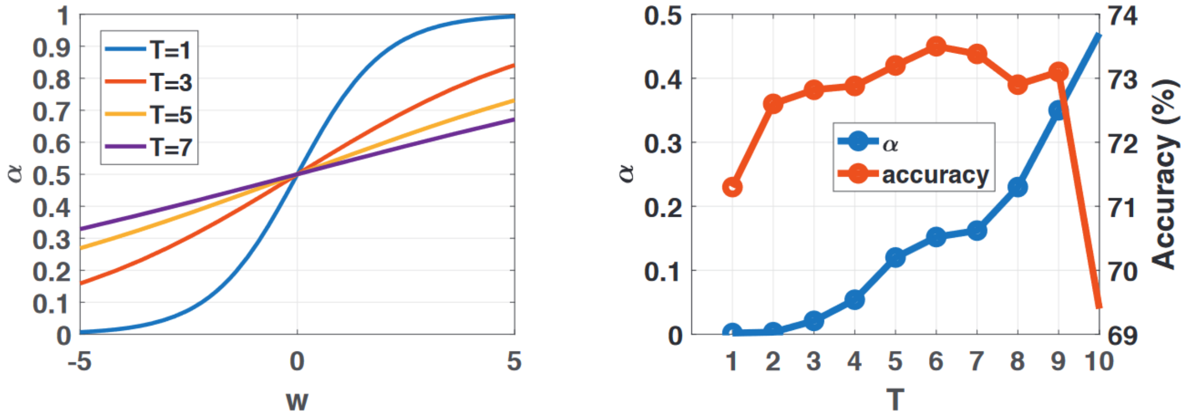
AgileNN : Offline Training



Combining Local and Remote Predictions



$$\alpha(w;T) = \frac{1}{1 + e^{-w/T}}$$



(a) Impact of T on α (b) Choices of T

Figure 8: Prediction weighting with α

Combined Training Loss

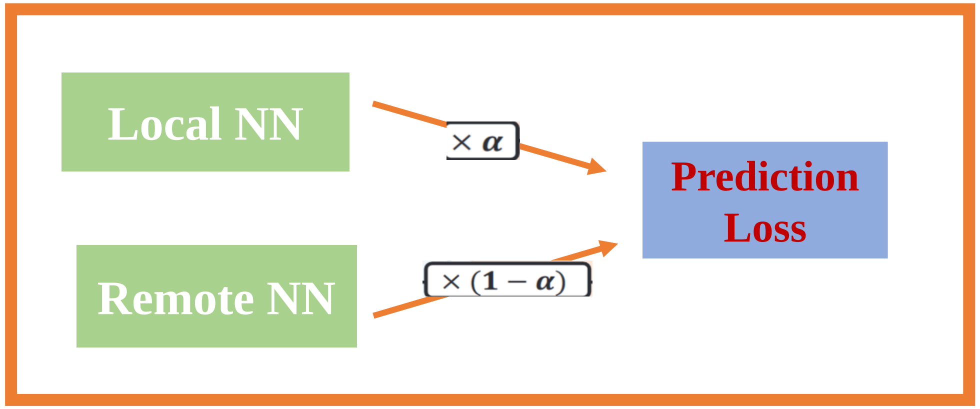
Disorder loss

Ensure topmost important features are extracted into topK channels

Skewness Loss

Enhance the important of topK features to ensure compressibility of the others

$$L = \lambda \cdot L_{\text{prediction}} + (1 - \lambda) \cdot (L_{\text{skewnss}} + L_{\text{disorder}})$$



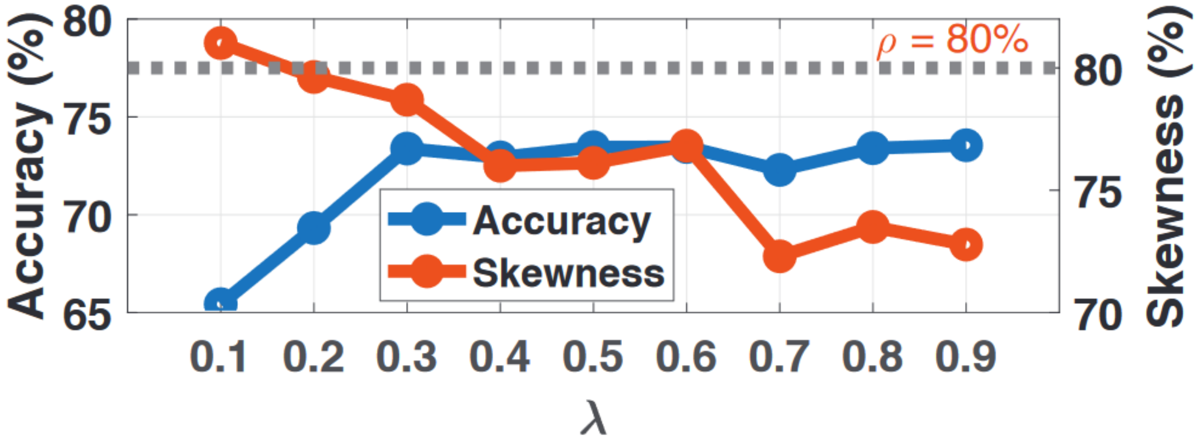


Figure 10: Impact of λ on CIFAR-100 dataset

Pre-processing the feature extractor

Figure 6: Training stability with different numbers of convolutional layers in feature extractor

Pre-processing the feature extractor

Algorithm 1 Selecting the k initial feature channels

Input: $\mathrm{D}_{\mathrm{train}}$: the training dataset with N samples;
$\mathcal{T}_{XAI}(\cdot)$: XAI-enabled Feature Importance Evaluator;
$\mathcal{E}(\cdot)$: Feature extractor that outputs C channels

Output: $(j_1, j_2, ..., j_k)$: The k selected feature channels.

$(p_1, p_2, ..., p_C) \leftarrow 0$ *//initialize*
for each $d_i \in \mathrm{D}_{\mathrm{train}}$ **do**
  $F \leftarrow \mathcal{E}(d_i)$ *// extract features*
  $I \leftarrow \mathcal{T}_{XAI}(F)$ *//evaluate feature importance*
  $F_{\mathrm{sorted}} \leftarrow \mathrm{sort}_I(F)$ *//sort features by their importance in descending order*
  $F_{top-k} \leftarrow F_{\mathrm{sorted}}[1:k]$ *//extract the top-k features with high importance*
  for c = 1,...,C **do**
    if $F[c] \in F_{top-k}$ **then**
      $p_c \leftarrow p_c + 1/N$
$R \leftarrow \mathrm{argsort}(p_1, p_2, ..., p_C)$ `//get the ranking of channels by their likelihood`
$(j_1, j_2, ..., j_k) \leftarrow R[1:k]$ `//decide top-`k` channels`

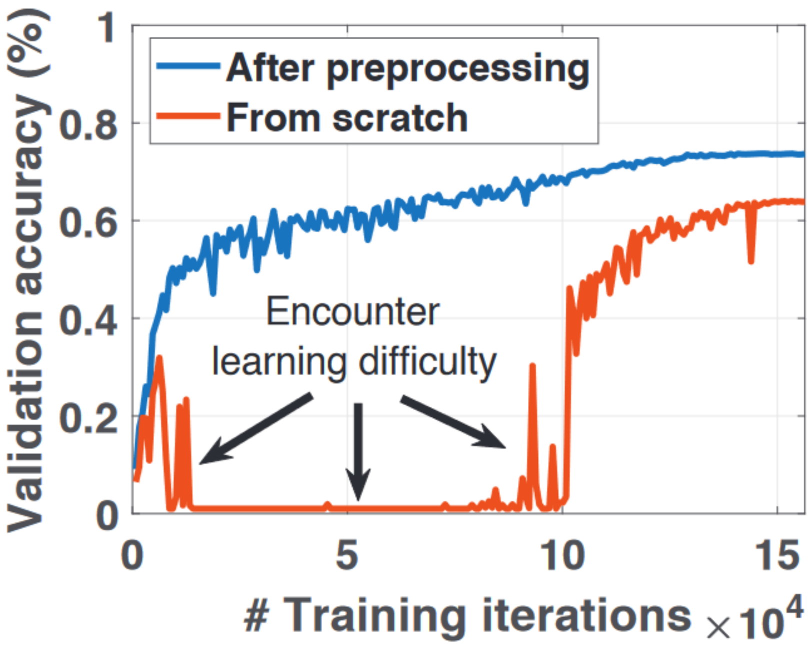


Figure 11: Effectiveness of Pre-processing

Pre-processing the feature extractor

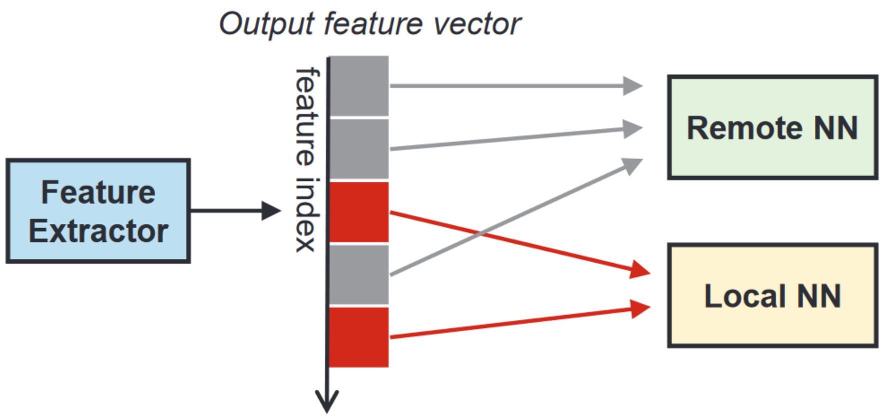


Figure 7: Pre-processing the feature extractor

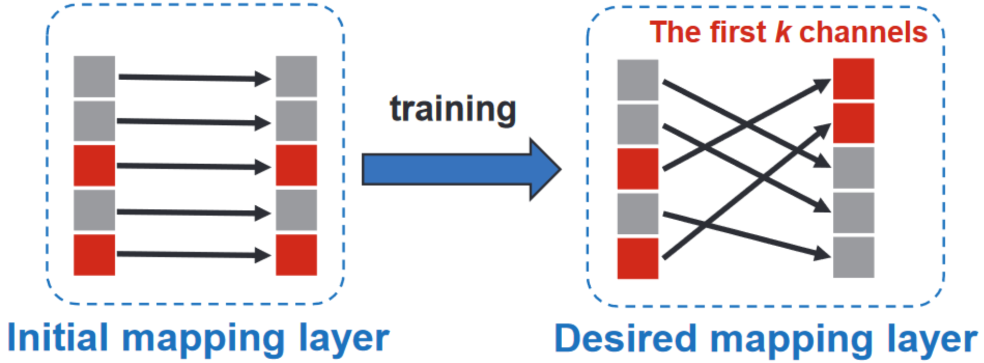


Figure 12: Training the mapping layer

Rearrange the indices to reduce unsorted cases in training data

XAI Tools - Integrated Gradients

- ## Consider 1

Skewness may not always exist in every input data.

- ## Consider 2

The accuracy of feature importance evaluation builds on accurate NN inference in advance.

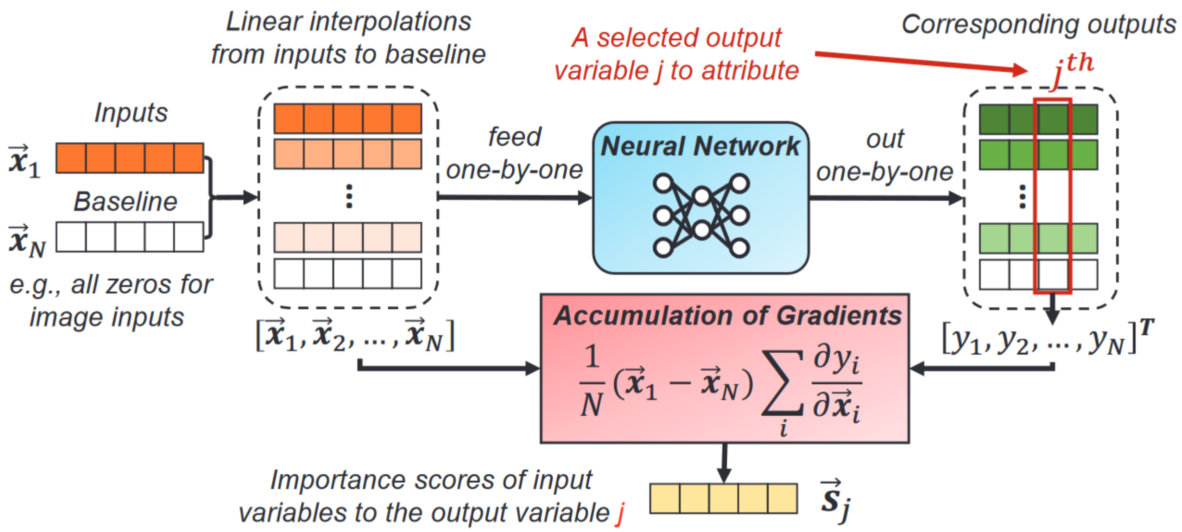


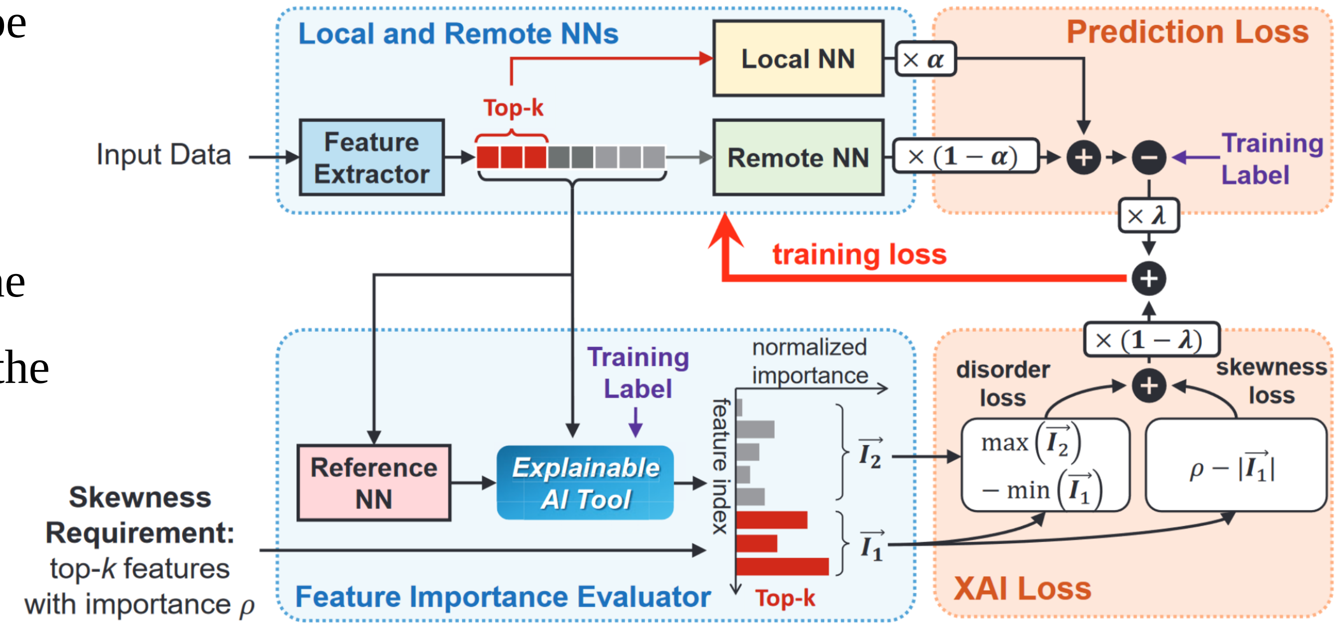
Figure 3: Integrated Gradients

AgileNN : Offline Training

- The **higher** the skewness is (i.e., smaller k and larger ρ), the **lower** resource consumption will be at the local.

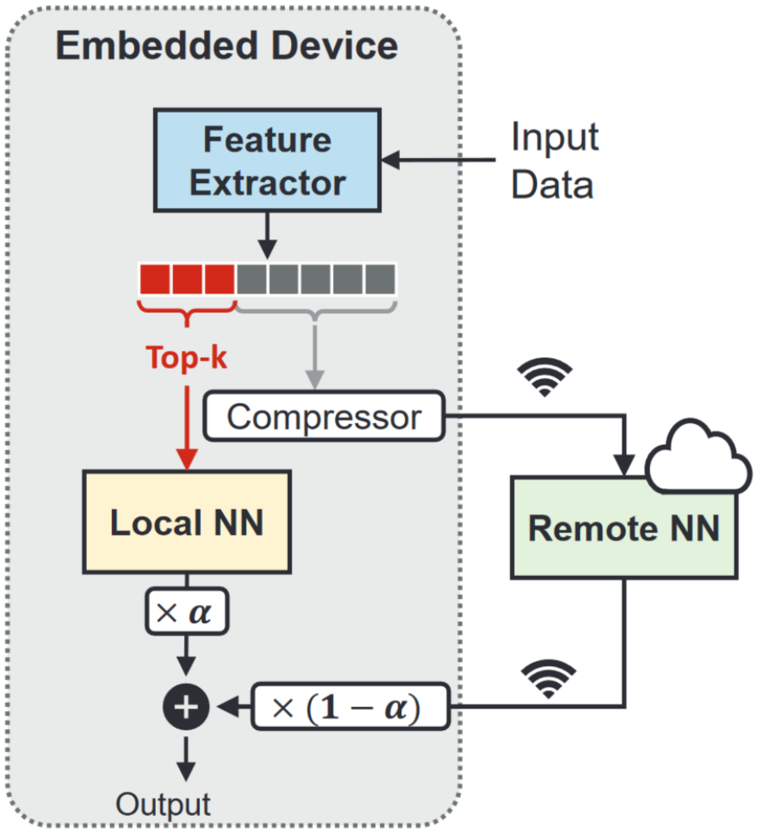
- But the NN inference is **more affected** due to the feature extractor's non-linear transformation in the feature space.

- *AgileNN* allow **flexible tradeoffs** between the accuracy and cost of NN inference on embedded devices, without incurring any extra computing or storage cost.



Offline Training

AgileNN : Online Inference



Online Inference

Implementation

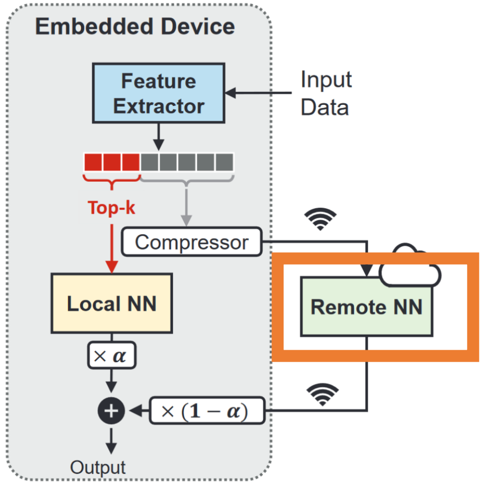
- **Local device**

  STM32F746 board, 216MHz, 320kb SRAM, 1MB FRAM

  ESP-WROOM-O2D WiFi module @ 6Mbps

- **Remote device**

  Dell-Precision 7820 workstation



Online Inference

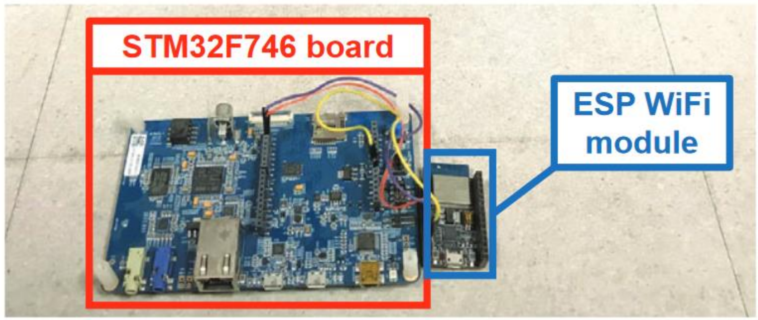


Figure 13: Devices in our implementation

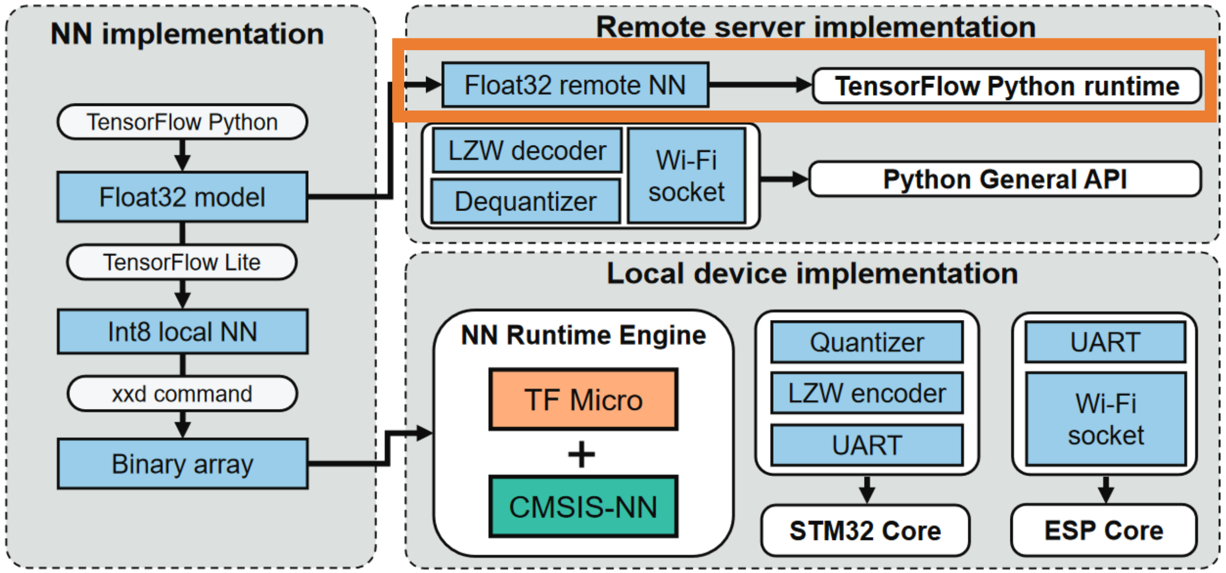


Figure 14: AgileNN implementation

Implementation

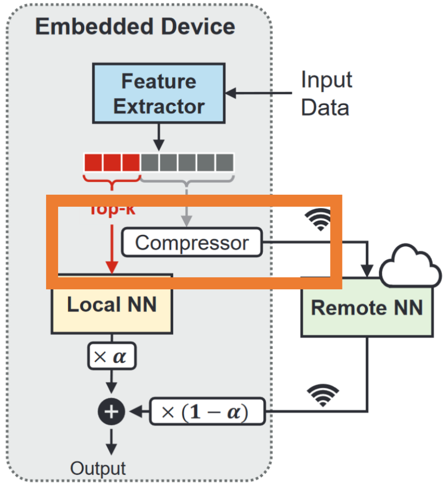
- **Local device**

  STM32F746 board, 216MHz, 320kb SRAM, 1MB FRAM

  ESP-WROOM-O2D WiFi module @ 6Mbps

- **Remote device**

  Dell-Precision 7820 workstation



Online Inference

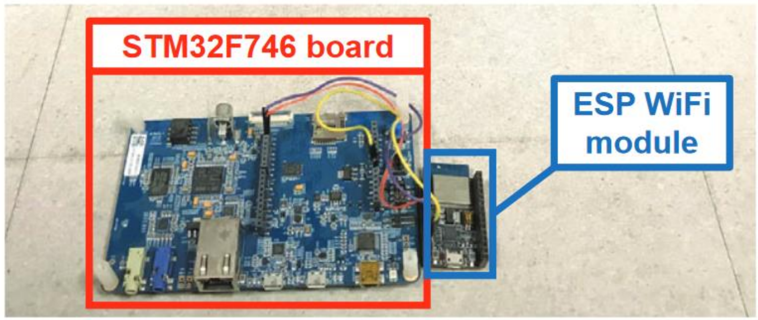


Figure 13: Devices in our implementation

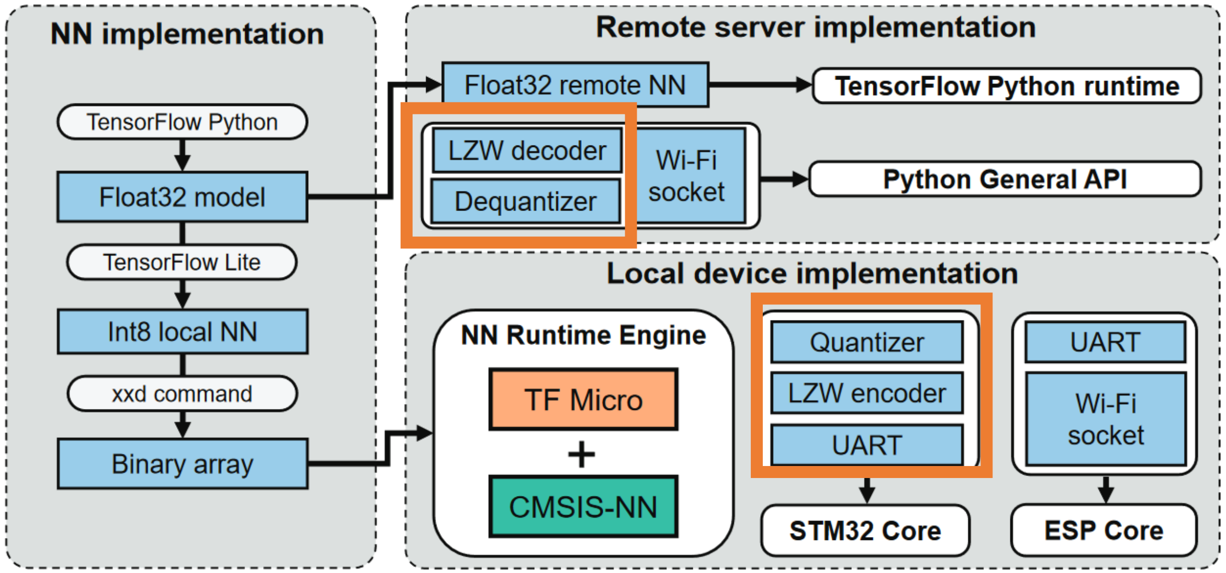


Figure 14: AgileNN implementation

Implementation

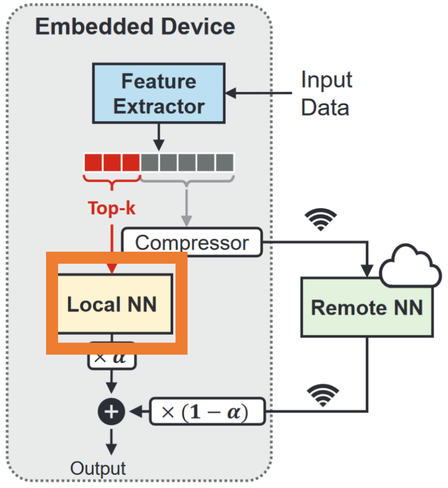
- **Local device**

  STM32F746 board, 216MHz, 320kb SRAM, 1MB FRAM

  ESP-WROOM-O2D WiFi module @ 6Mbps

- **Remote device**

  Dell-Precision 7820 workstation



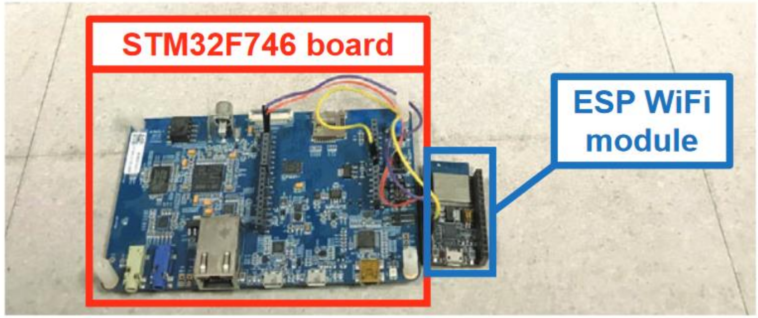


Figure 13: Devices in our implementation

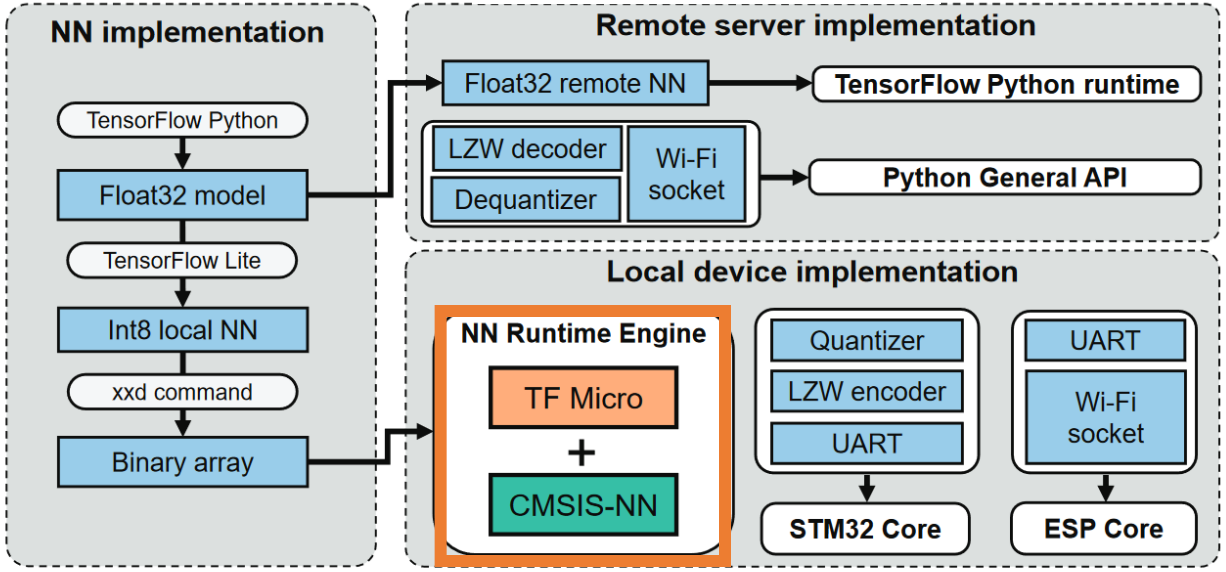


Figure 14: AgileNN implementation

Implementation

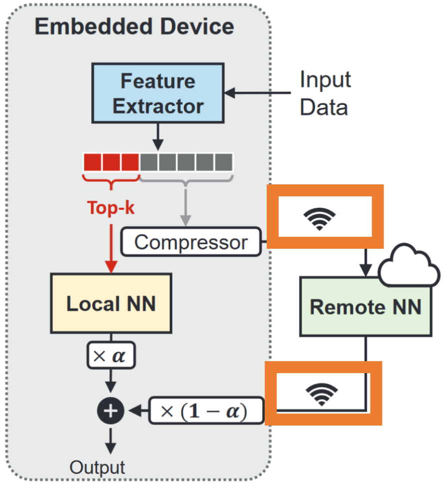
- **Local device**

  STM32F746 board, 216MHz, 320kb SRAM, 1MB FRAM

  ESP-WROOM-O2D WiFi module @ 6Mbps

- **Remote device**

  Dell-Precision 7820 workstation



Online Inference

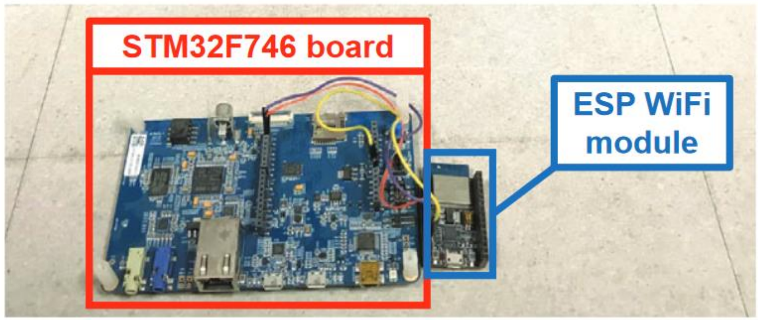


Figure 13: Devices in our implementation

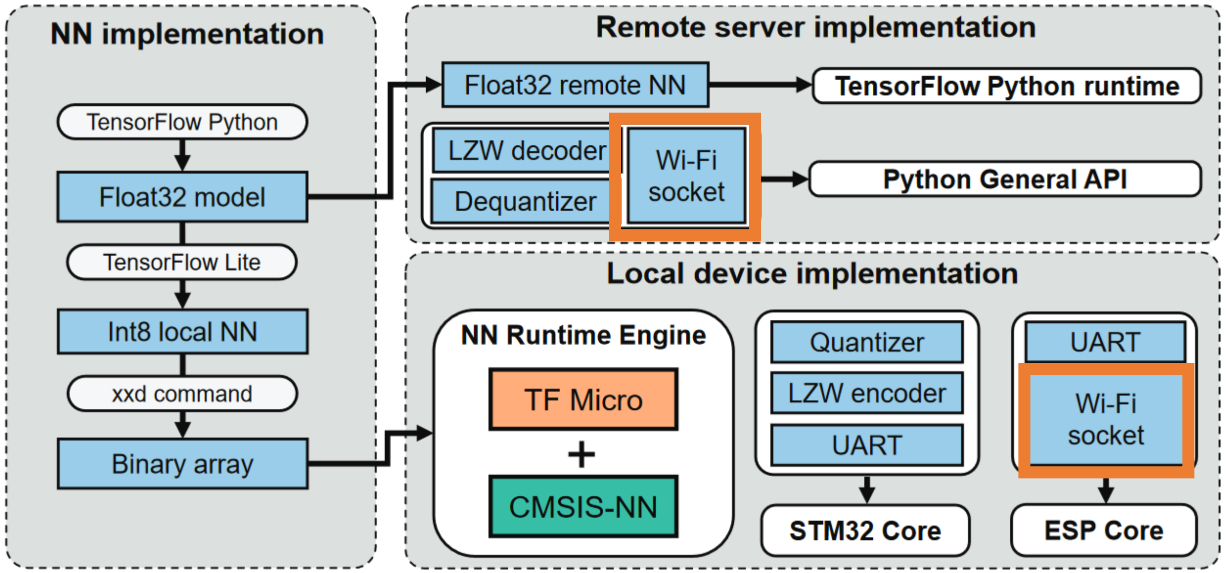


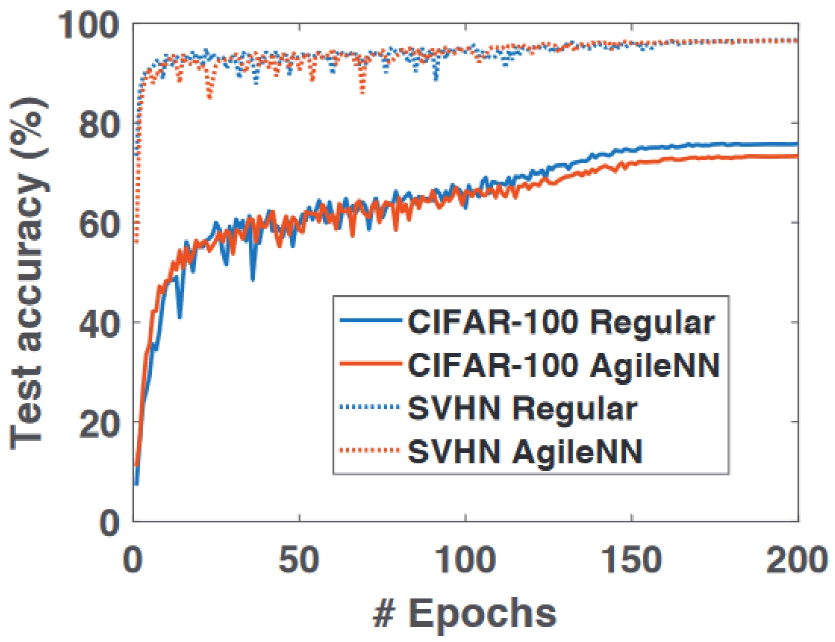
Figure 14: AgileNN implementation

Evaluation Setup

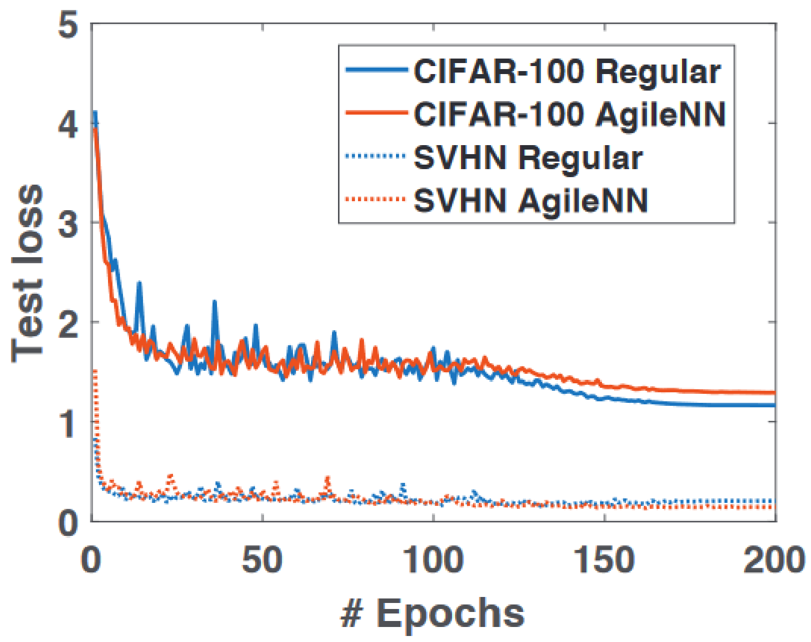
- **Baselines**
  - **MCUNet [1] – NAS to find the best local NN structures – Local NN**
  - **DeepCOD [2] – use a NN-based encoder – Remote NN**
  - **SPINN [3] – early-exit structures NN – NN offloading**
  - **Edge-only Inference – compress and offload raw data**
- **Datasets**
  - **CIFAR-10/100**
  - **SVHN**
  - **ImageNet-200**
- **Reference NN**
  - **EfficientNetV2 CNN**

[1] Lin, Ji, et al. "Mcunet: Tiny deep learning on iot devices." *Advances in Neural Information Processing Systems* 33 (2020): 11711-11722.
[2] Yao, Shuochao, et al. "Deep Compressive Offloading: Speeding Up Edge Offloading for AI Services." GetMobile: Mobile Computing and Communications 25.1 (2021): 39-42.
[3] Laskaridis, Stefanos, et al. "SPINN: synergistic progressive inference of neural networks over device and cloud." Proceedings of the 26th annual international conference on mobile computing and networking. 2020.

Evaluation - Training Convergence and Cost



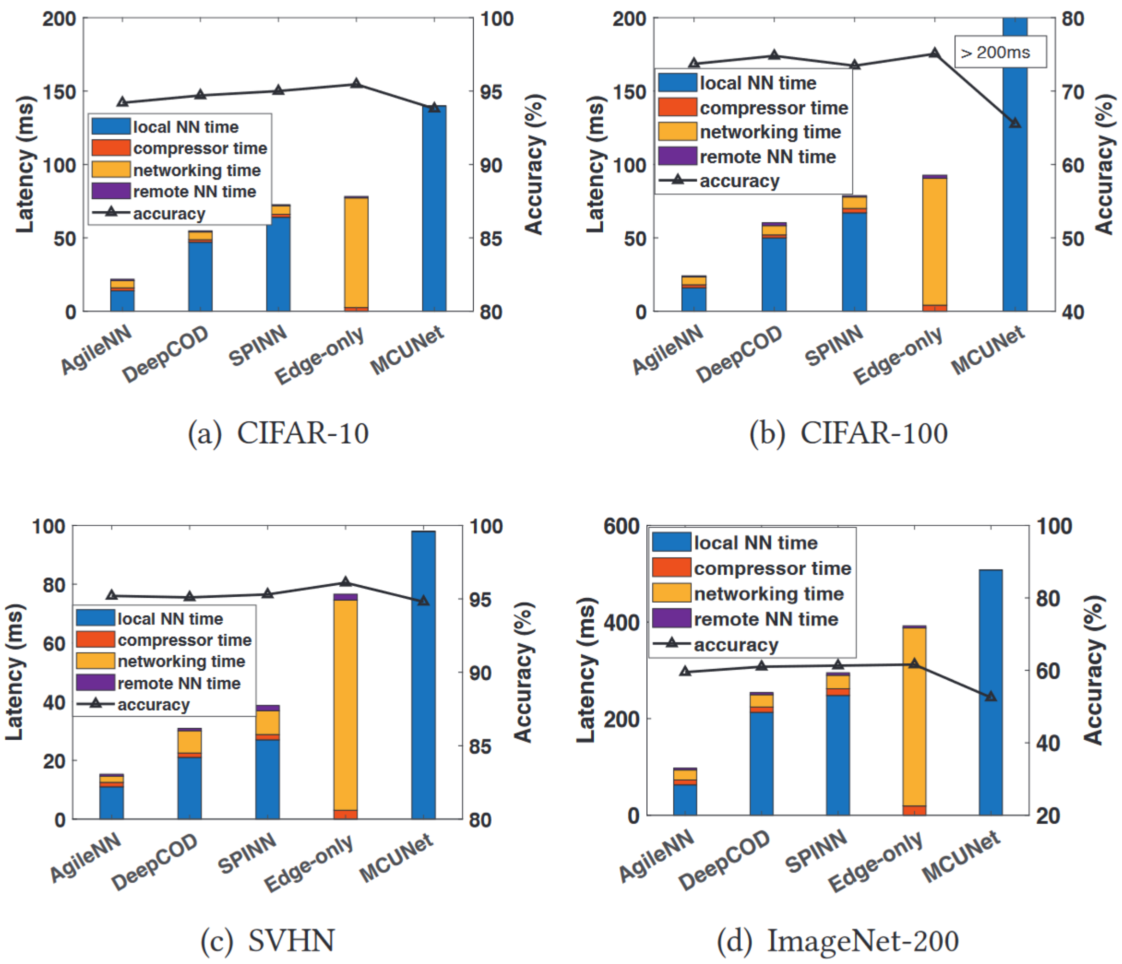
(a) Test accuracy



(b) Test loss

Evaluation - Accuracy and Latency

- *AgileNN* reduces end-to-end latency by 2x – 2.5x

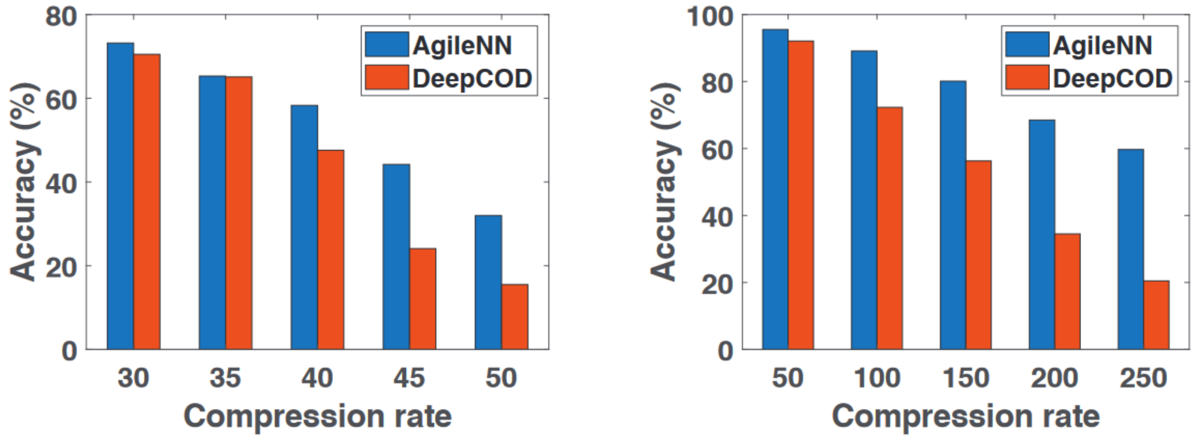


Dataset	CIFAR-10	CIFAR-100	SVHN	ImageNet
Reduction	43.7%	15.8%	72.3%	20.8%

Table 2: Reduction of transmitted data size, compared to DeepCOD [65]

Evaluation - Accuracy and Latency

- *AgileNN* reduces end-to-end latency by 2x – 2.5x



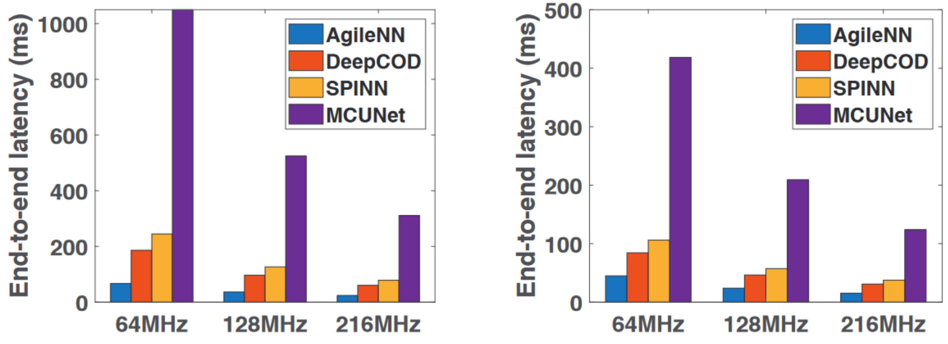
(a) CIFAR-100 (b) SVHN

Figure 17: Accuracy with different compression rates

Evaluation – Different System Settings

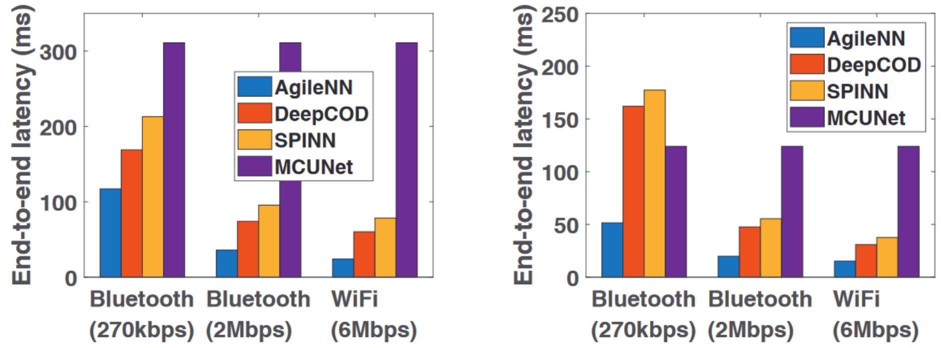
- **Impact of Local CPU Frequency**
  - **64MHz – 216MHz**
  - **Reduces latency by 2.1x to 2.5x**

- **Impact of Network Bandwidth**
  - **Bluetooth 270kbps, 2Mbps**
  - **WiFi 6Mbps**



(a) CIFAR-100 (b) SVHN

Figure 22: The impact of different CPU Frequencies

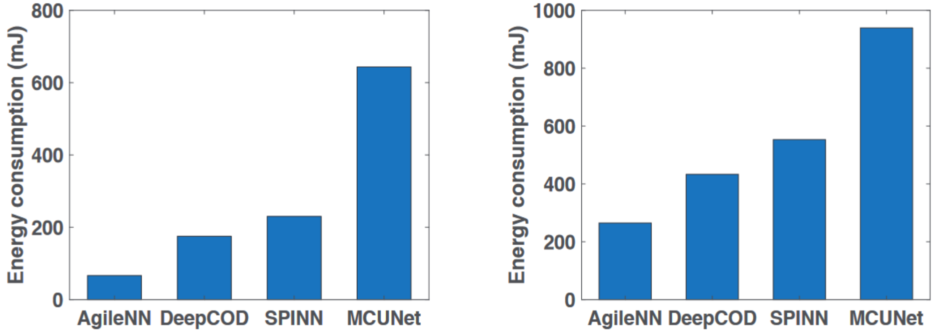


(a) CIFAR-100 (b) SVHN

Figure 23: The impact of different wireless bandwidths

Evaluation – Local Resources Consumption

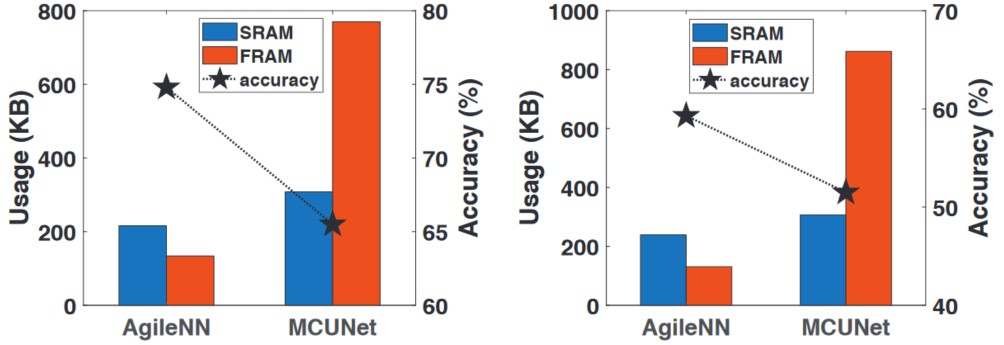
- **Local energy consumption**
  - **1.6x – 2.5x more efficient than DeepCOD**
  - **8x more efficient than MCUNet**



(a) CIFAR-100 (b) ImageNet-200

Figure 19: Local energy consumption per NN inference run

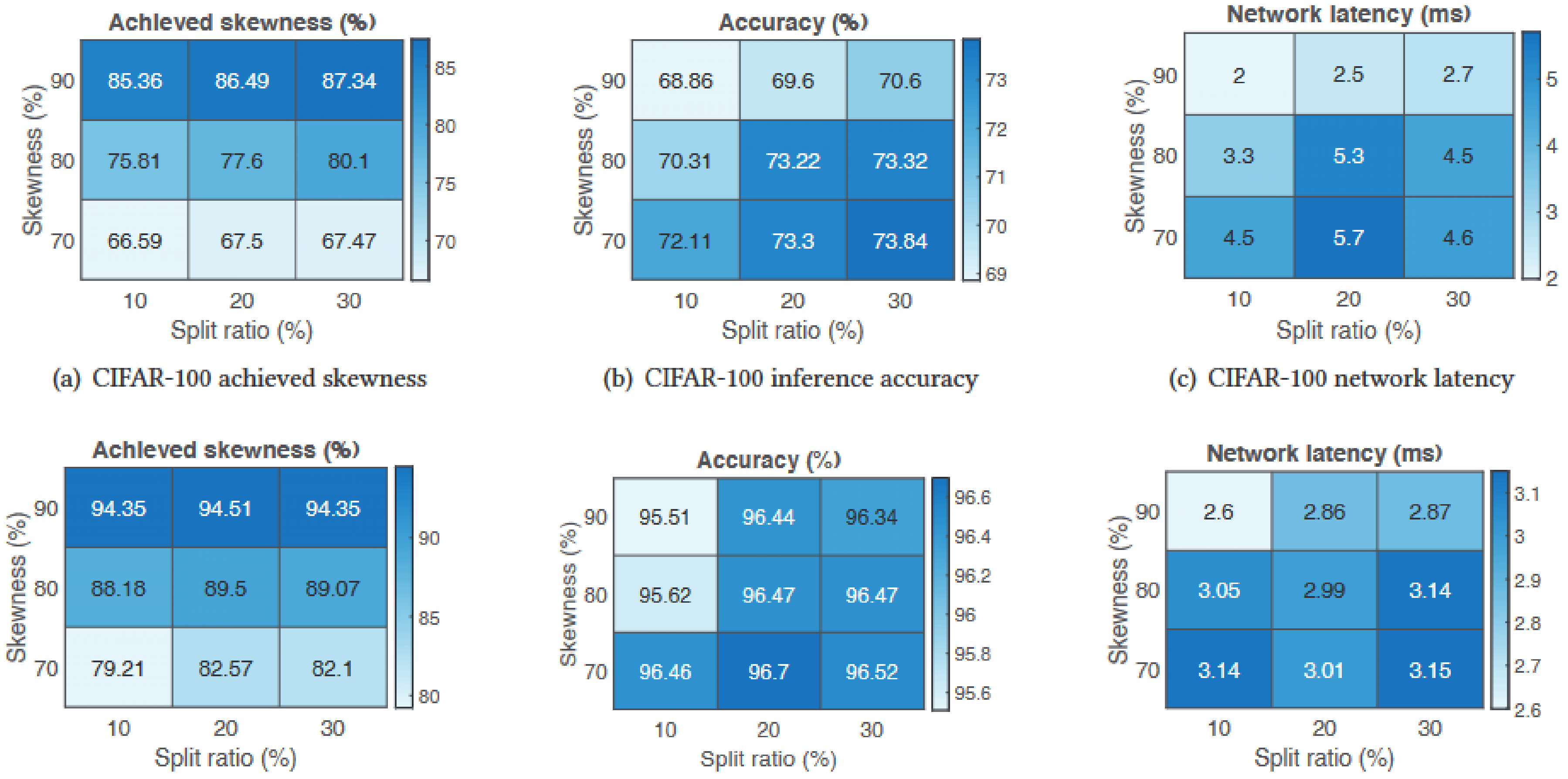
- **Memory and storage usage**
  - **SRAM to Memory**
  - **FRAM to Storage**
  - **Higher accuracy, and save 40%-50% memory and >50% storage**



(a) CIFAR-100 (b) ImageNet-200

Figure 20: Memory and storage usage

Evaluation – Effectiveness of Skewness Manipulation



(a) CIFAR-100 achieved skewness (b) CIFAR-100 inference accuracy (c) CIFAR-100 network latency

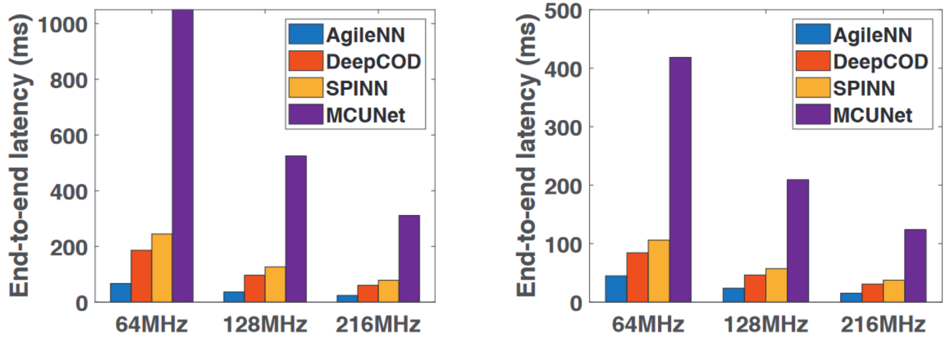
(d) SVHN achieved skewness (e) SVHN inference accuracy (f) SVHN network latency

Figure 21: Effectiveness of skewness manipulation with different requirements of feature importance skewness

Evaluation – Different System Settings

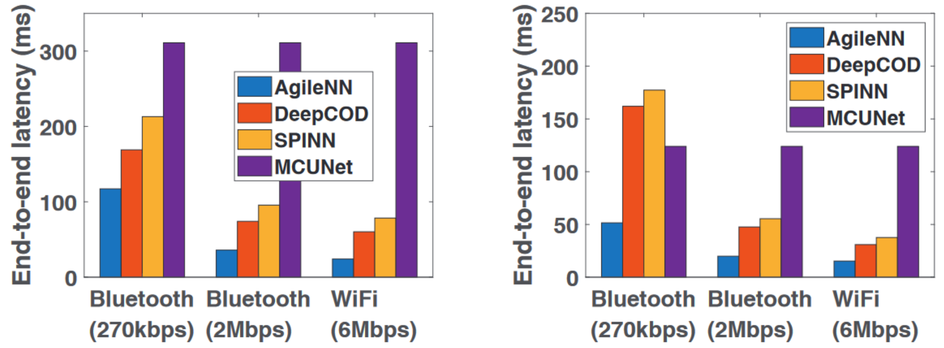
- **Impact of Local CPU Frequency**
  - 64MHz – 216MHz
  - Reduces latency by 2.1x to 2.5x

- **Impact of Network Bandwidth**
  - Bluetooth 270kbps, 2Mbps
  - WiFi 6Mbps



(a) CIFAR-100 (b) SVHN

Figure 22: The impact of different CPU Frequencies



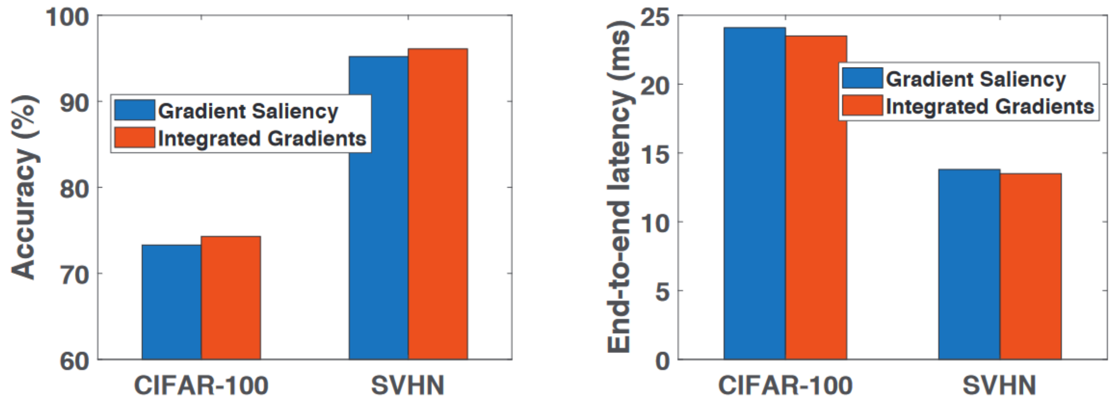
(a) CIFAR-100 (b) SVHN

Figure 23: The impact of different wireless bandwidths

Evaluation - Choices of XAI techniques

- **Gradient Saliency**
  - **evaluate each feature's importance by injecting noise.**

- **Integrated Gradients**
  - **It aggregates more interpolations of NN outputs' gradients**
  - **IG is more computationally.**



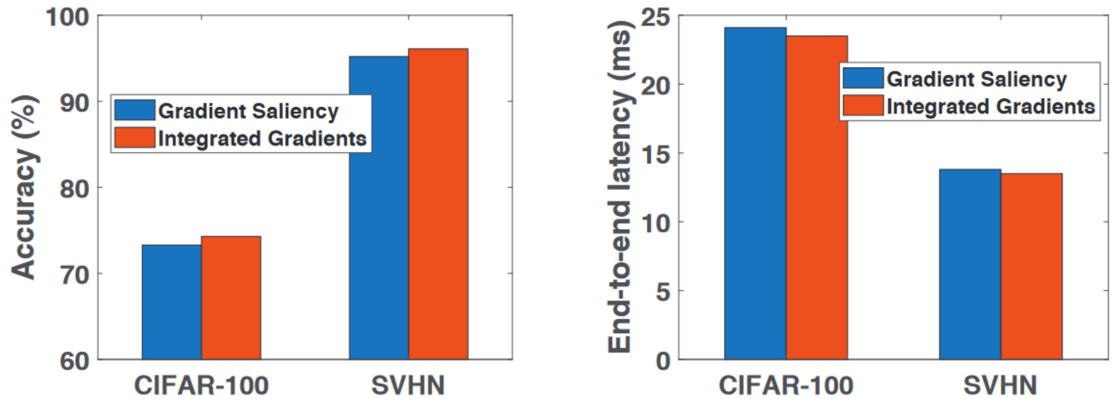
(a) Inference accuracy (b) End-to-end latency

Figure 24: Different XAI techniques

Evaluation - Choices of XAI techniques

- **Gradient Saliency**
  - **evaluate each feature's importance by injecting noise.**

- **Integrated Gradients**
  - **It aggregates more interpolations of NN outputs' gradients**
  - **IG is more computationally.**



(a) Inference accuracy (b) End-to-end latency

Figure 24: Different XAI techniques

Advantage

- First to integrate XAI techniques in to NN offloading systems
- Shifts the rationale of offloading from fixed to data-centric & agile
- >6x lower latency and >8x resources consumption for extremely weak devices

Disadvantage

- XAI tools do not consider the relationship between features.
- Hard for static NN models to adopt to new data and different application scenarios.