Hardware Architectures for Embedded and Edge AI

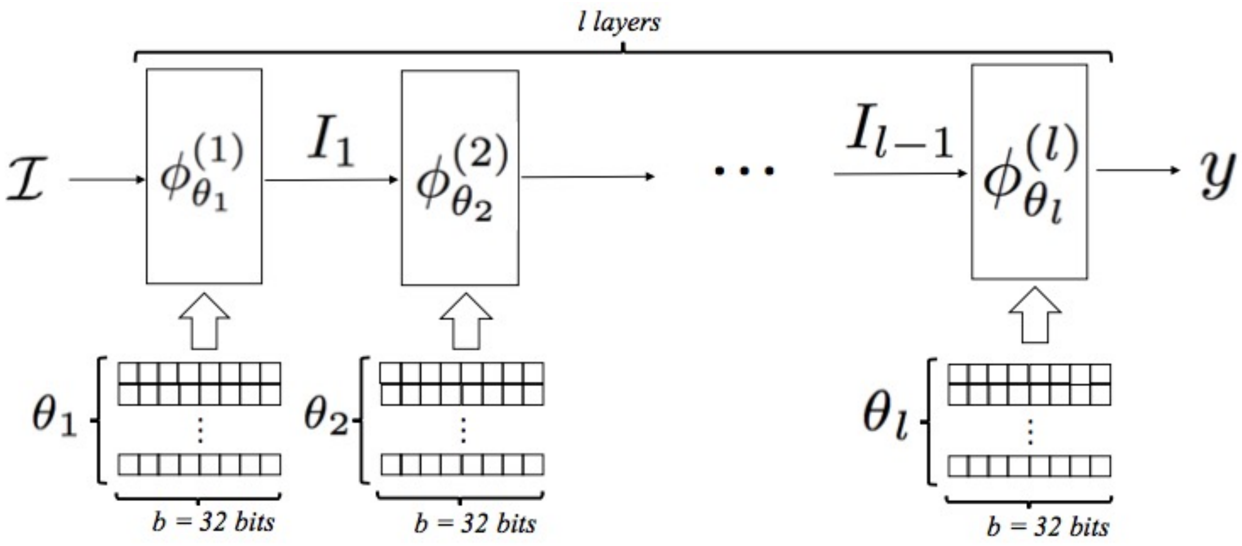
Prof Manuel Roveri – manuel.roveri@polimi.it

Lecture 8 – Early Exit Neural Networks

Introduction to Early exit neural networks

- Deep neural networks (DNNs) are widely used in many different application scenarios such as image classification, and recognition as well as in other forecasting and detection tasks.
- DNNs are generally designed as a stack of layers, in which a result is obtained only after processing the full stack.
- This might lead to drawbacks

The idea of Early Exit Neural Networks

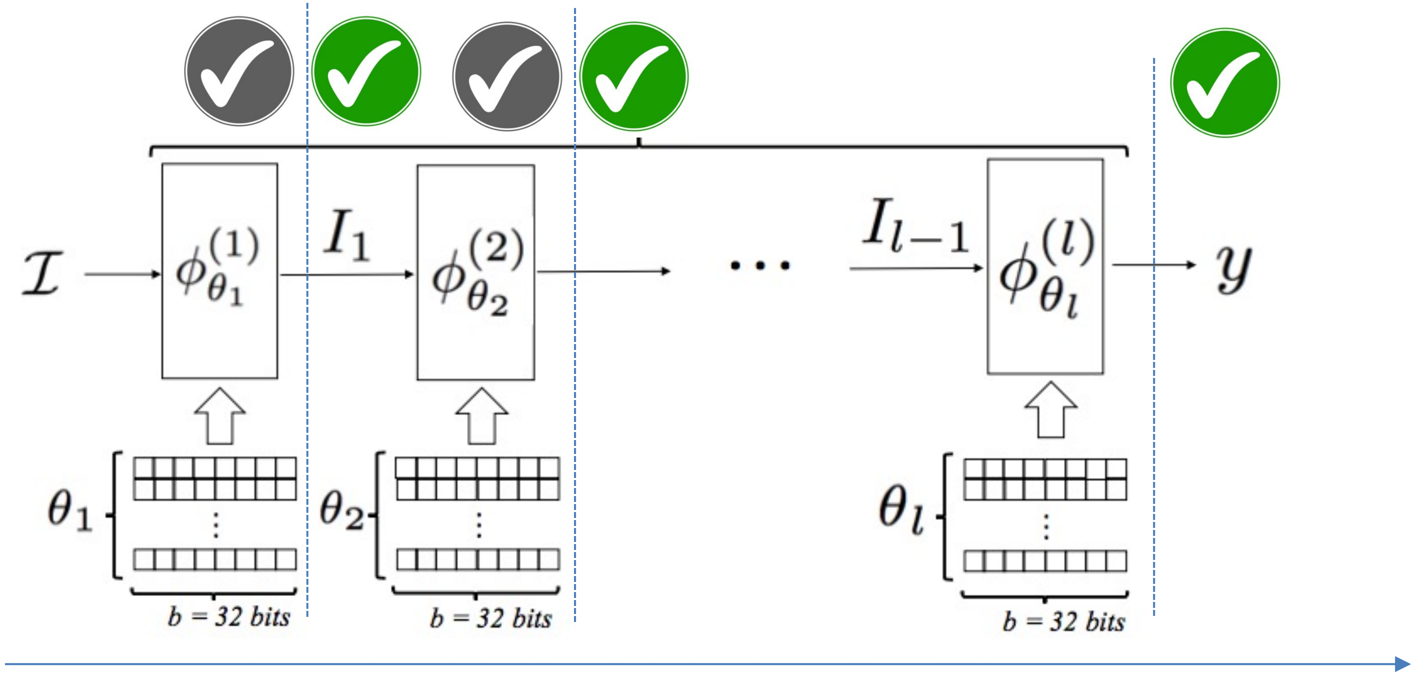


Features characterized by increasing complexity and meaning

The computational load is measured as the number of multiplications required to accomplish the layer goals

$$C^{\Phi}_{conv} = \sum_{i=1}^{N_c} n_{i-1} \cdot s_i^2 \cdot n_i \cdot m_i^2,$$

The idea of Early Exit Neural Networks



Incrementally process the input image through the CNN layers and take a decision as soon as "enough confidence" about the classification is gained

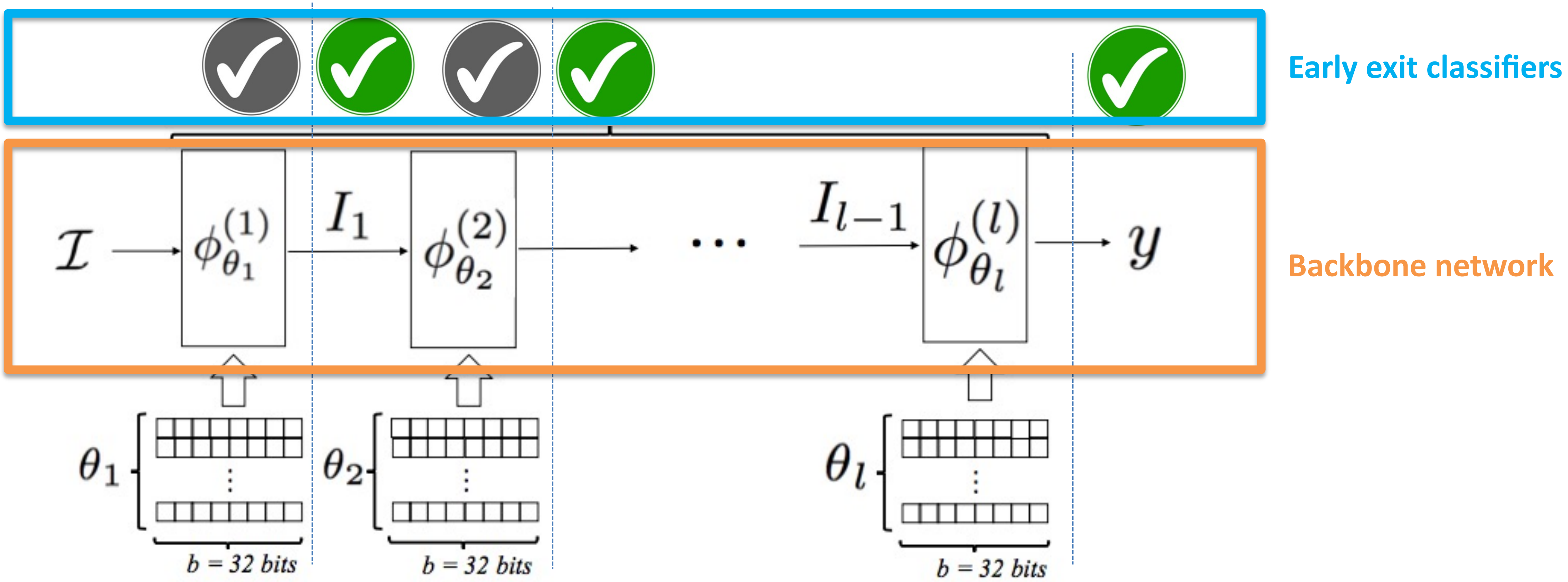
The Early exit classifiers and the Backbone Network

- Early Exit Neural Networks (EENNs) endow network architectures with ***Early Exit Classifiers*** (EECs).

- By adding EECs, EENNs can progressively process the input and make decisions at intermediate points of the network.

- We refer to the original network architecture (i.e., without EECs) as ***Backbone network***.

The Early exit classifiers and the Backbone Network



The «lottery» idea behind EENNs

- The majority of input samples could be classified by resorting to smaller architectures even in very complex datasets such as ImageNet.
- The “lottery Ticket” hypothesis: **bigger is not always better**
- Indeed, most input samples are classified by the EENNs in earlier stages of the neural network.

Properties of EENNs

- Mitigation of many drawbacks of DNNs such as overfitting, vanishing gradient and **overthinking**;
- Significant reduction of the inference time;
- Capability of being distributed over multi-tier computation platforms;
- Adaptiveness to changing environments by achieving a desired trade-off between accuracy and efficiency on the fly;
- Increased interpretability.

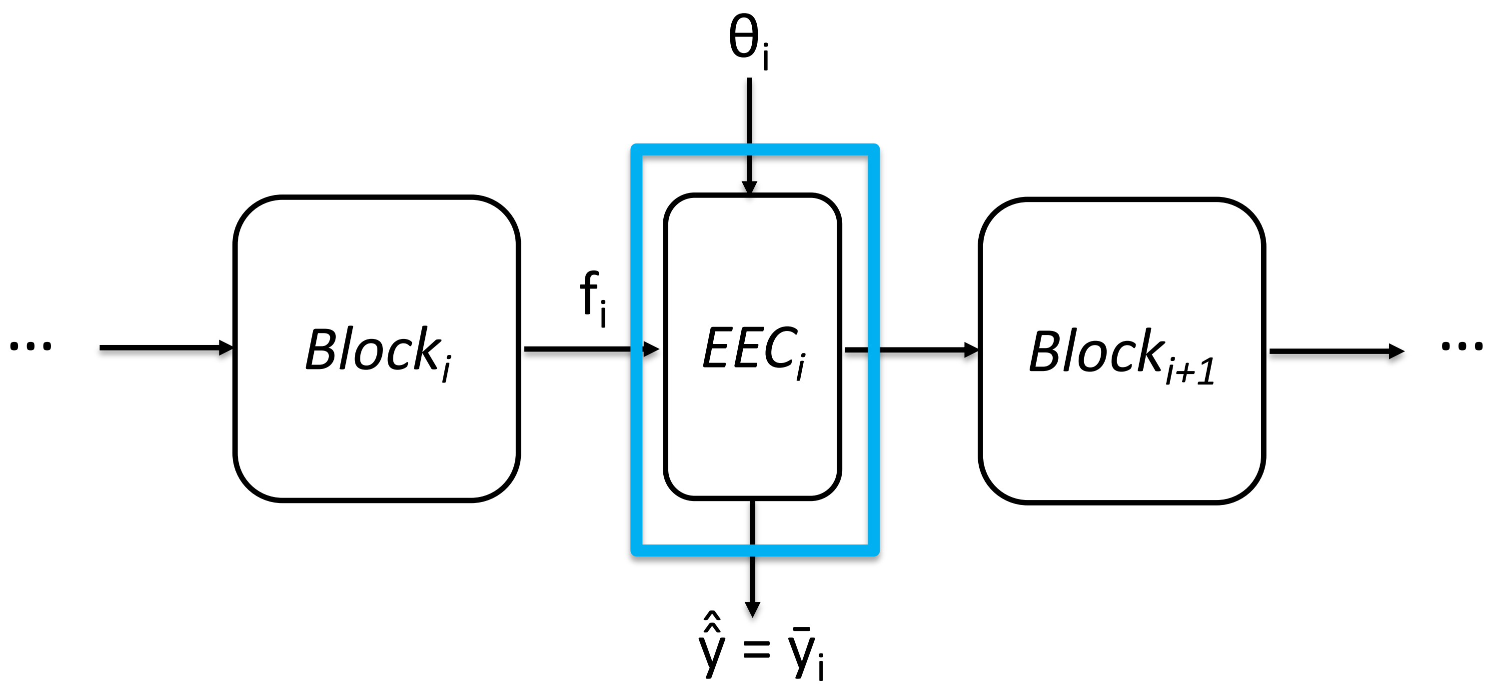
Overthinking: a hidden (but dangerous) behaviour

- Overthinking is a phenomenon that points out that predictions that would be **correct with smaller architectures** become **incorrect with deeper architectures**.

- This means that predictions computed by earlier EECs are not necessarily less accurate than the ones of the next EECs and could be even better!!

Early Exits: the classifier and the selection scheme

A generic EEC in an Early Exit Neural Network



Early Exits: the classifier and the selection scheme

- Denote by $f_i(x)$ the output of an intermediate layer i (or block)
- An **Early Exit Classifier (EEC)** is a small classifer added on top of it:

$$\bar{y}_i = C_i(f_i(x))$$

- Hence, the EENN provides a sequence of predictions $\bar{y}_i$, one for each EEC, potentially with different accuracy, in addition to the final classification $\hat{y}$.

Early Exits: the classifier and the selection scheme

- An input sample exits from an EEC_i when **enough confidence is achieved** *[we will come back on this later on…]*
- In this case, **the sample is not forward propagated** in the network and an intermediate prediction $\bar{y}_i$ is provided by the EEC_i.
- This decision is taken by the so-called ***selection scheme*** which is a set of decision functions, one per EEC.
- Let θ_i be the **hyperparameters** of the decision function corresponding to the i-th EEC

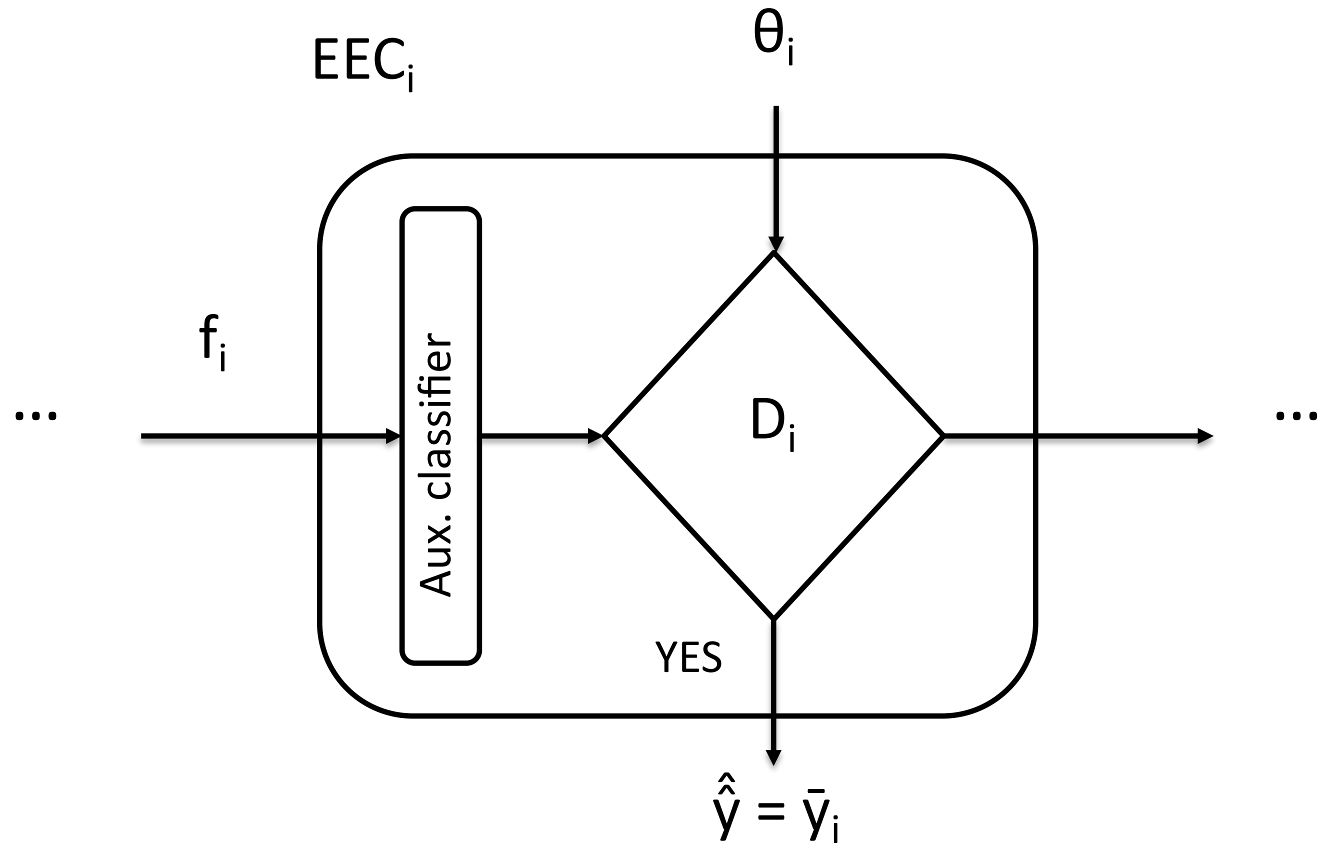
Early Exit - Decision function

- The hyperparameters θ_i of the selection scheme represent **the set of thresholds to decide whether to early exit**.
- For each processed input, the decision function D_i simply **compares the confidence** value C_i **with the corresponding threshold** θ_i of the EEC_i.
- The implementation of a **decision function** D_i is straightforward:

$$D_i(\mathrm{x}) = C_i(x) \geq \theta_i$$

with i=1, .. , N and being N is the number of EECs

Early Exit: Auxiliary Classifier and Decision Function



The training of EENNs

The training of EENNs

The training EENNs can be categorized into three main families:

1. Joint Training (JT)
2. Layer-wise Training (LT)
3. Knowledge Distillation (KD)

Training with Joint Training

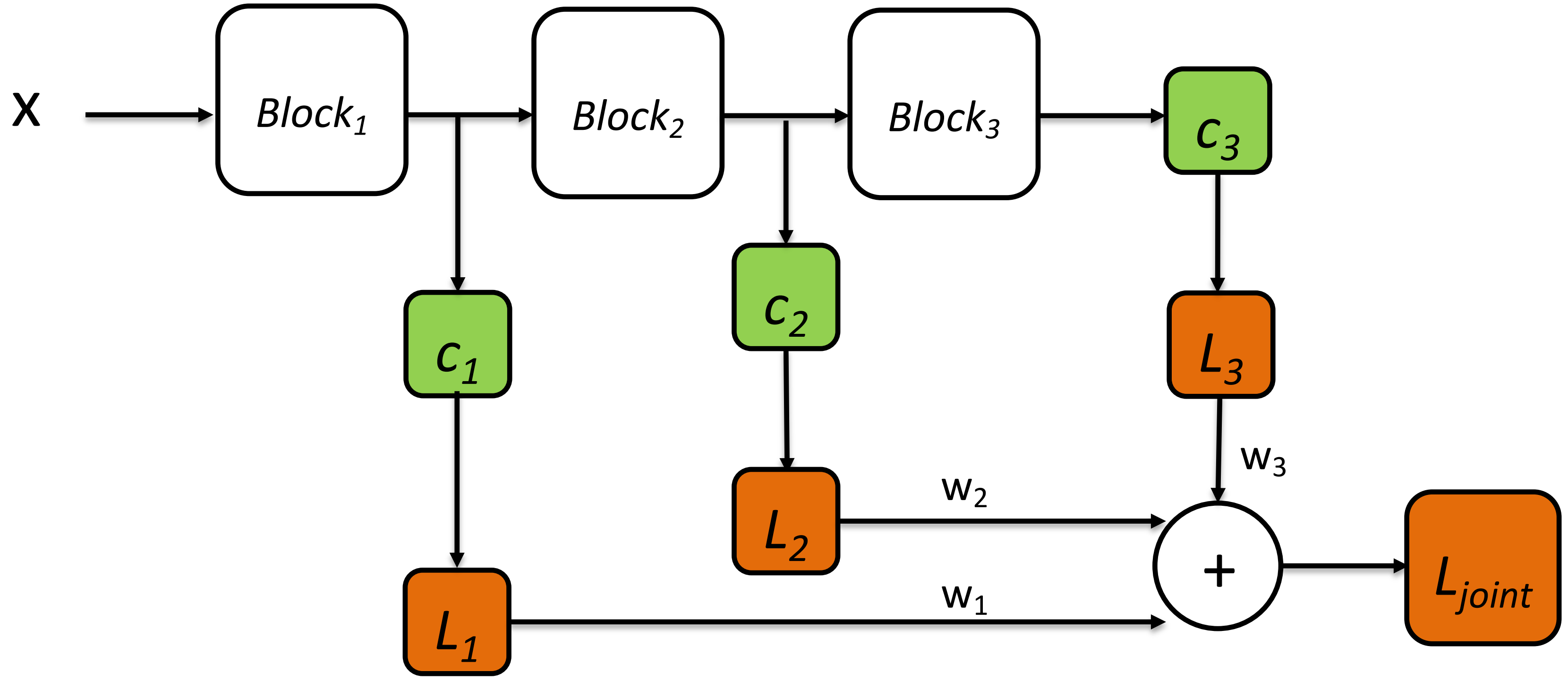
- **Joint Training (JT)**: jointly training all EECs by combining the losses of the classifier.

$$L_{joint} = L(\hat{y}, y) + \sum_{i=1}^{N} w_i * L(\bar{y}_i, y)$$

where L is the standard Cross Entropy loss and N is the number of EECs.

- The weights w_i can be set to 1 or become hyperparameters.

Joint Training: a graphical description



Training with Layer-wise Training and Knowledge Distillation

- **Layer-wise Training (LT)**: train iteratively one block of the backbone network and the corresponding EEC keeping frozen the former pipeline of the neural network.

- **Knowledge Distillation (KD)**: KD initially trains the backbone network (teacher) and, then, the EECs (students) on top of the trained backbone network

The inference of EENNs

Inference: aggregation or early exits

- The most trivial approach is to **aggregate the outputs** of all the EECs so as to provide a joint prediction:
  - “Fake” EENNs

- The most interesting approach is to process the input up to a given EEC and then, **stop the forward propagation**. This is the reference inference scheme for EENNs
  - “True” EENNs

The selection criterion in EENNs

Selection criterion – measure of the confidence

- For classification problems, a popular approach for EENNs relies on ability to **estimate the confidence of the neural network** on its own prediction and **use it to decide whether to early exit** the processed input.

- The **confidence value** can be measured mainly in three ways:
  - ✓ Max Softmax output
  - ✓ Score Margin
  - ✓ Entropy

Selection criterion #1: the Max Softmax output

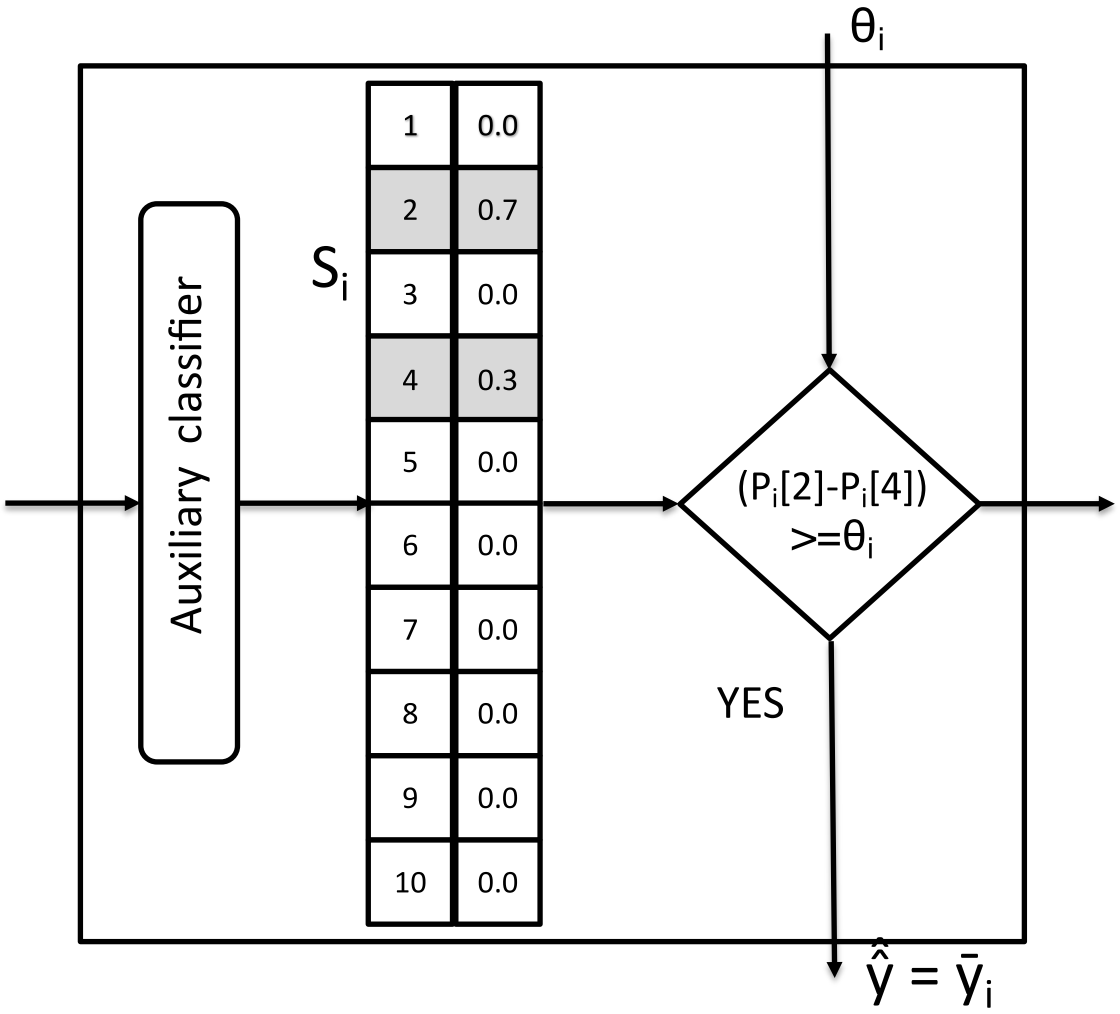
- Denote by $S_i(x)$ the vector of the softmax on the prediction y_i computed by the EEC_i for an input x.
- The **maximum value of $S_i(x)$** can be used as a confidence measure.

Selection criterion #2: the Score Margin

- The **score margin (SM)** is the distance between the largest and the second largest value of $S_i(x)$.

- Intuitively, the higher the “confidence” of the EEC_i about its prediction, the higher the difference between the largest and the second largest value of S_i
  - implying a larger value of SM (since $S_i(x)$’s sum to 1).

Selection criterion #2: Score Margin



Scheme of an EEC using the Score Margin in the Decision Function.

In this example:

- $P_i[2]$ is the largest value of S_i
- $P_i[4]$ is the second largest value of S_i

Selection criterion #3: Entropy

- The entropy H estimates the level of uncertainty on the prediction:

$$\mathrm{H(y)} = -\sum_{i=0}^{N} \mathrm{y_i} * \log(\mathrm{y_i})$$

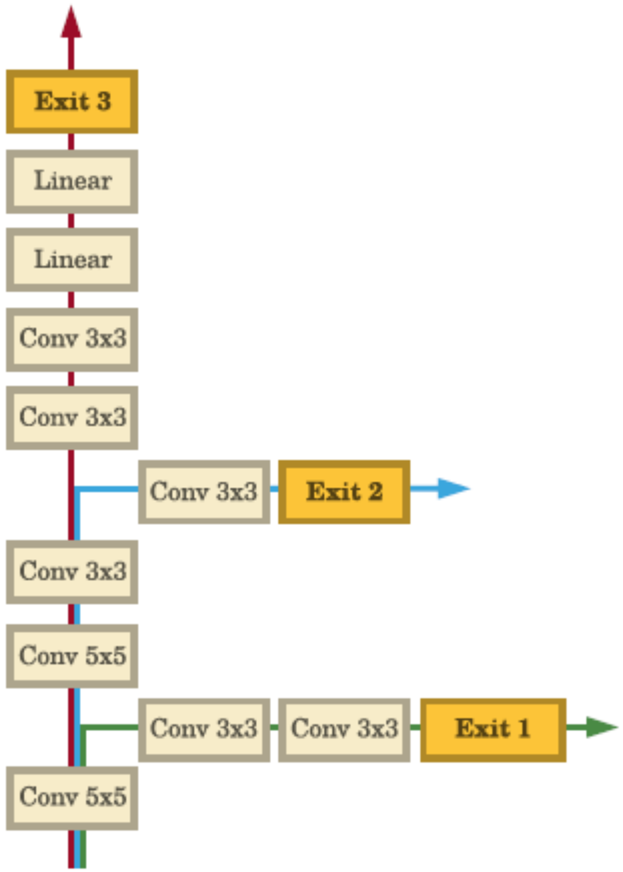
with N the number of EECs.

- The entropy is minimal whenever y equals a one-hot vector, and maximal when it is equal to a uniform distribution over classes.

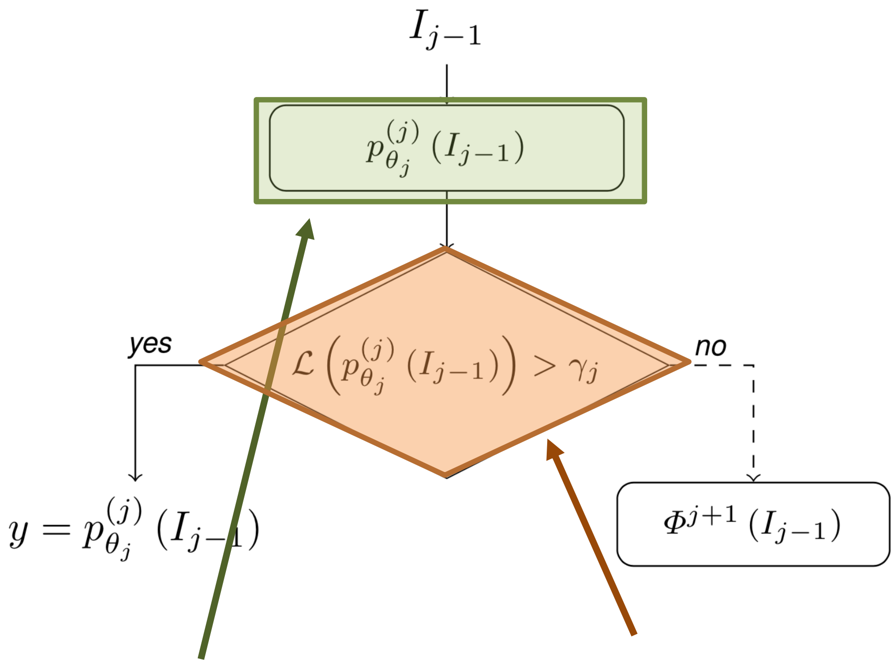
Two examples of EENNs

BranchyNet

- Two EECs added to the backbone (original) AlexNet.
- The training is performed in a JT approach.
- In inference, there is no further computation if a sample exits from an EEC.
- The confidence used is the entropy.

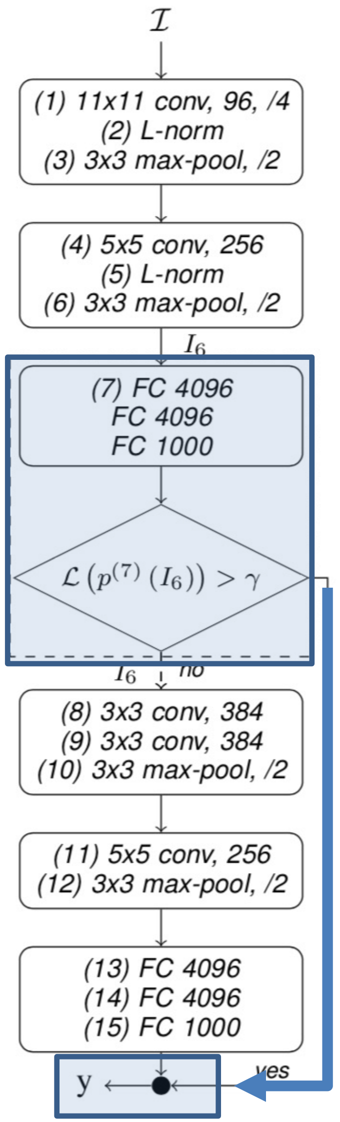


Gate-Classification Neural Networks



- «Confidence» is modelled as the **posterior probability of the classification** of the input image
- The decision is taken as soon as the posterior probability is **larger than automatically-defined thresholds**

A Gate-Classification version of the AlexNet.



References and Acknowledgements

1. J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks"
2. S. Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Why Should We Add Early Exits to Neural Networks?"
3. Han, Yizeng & Huang, Gao & Song, Shiji & Yang, Le & Wang, Honghui & Wang, Yulin. Dynamic Neural Networks: A Survey.
4. S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast Inference via Early Exiting from Deep Neural Networks"

A big thank to Matteo Gambella for the support in the definition of the slides!