Institut de Robòtica
i Informàtica Industrial

CSIC

UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems

ICPR 2020 paper #1294

Javier García López.
javier.jarcia.l@upc.edu
Francesc Moreno-Noguer
fmoreno@iri.upc.edu
Antonio Agudo
antonio.agudo@upc.edu

Paper presented at the 25th. International Conference on Pattern Recognition
Milano, IT. Jan 10-15, 2021.

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems
ICPR 2020 #1294

Current problem:

Current manual neural network design approaches prevent the usage of Deep Learning-based application on many fields since it is a hard and time-consuming.

Newly Neural Architecture Search (NAS) methods provide good response to this problem, however they are still very time consuming and not very suited for big datasets such as ImageNet or COCO.

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems
ICPR 2020 #1294

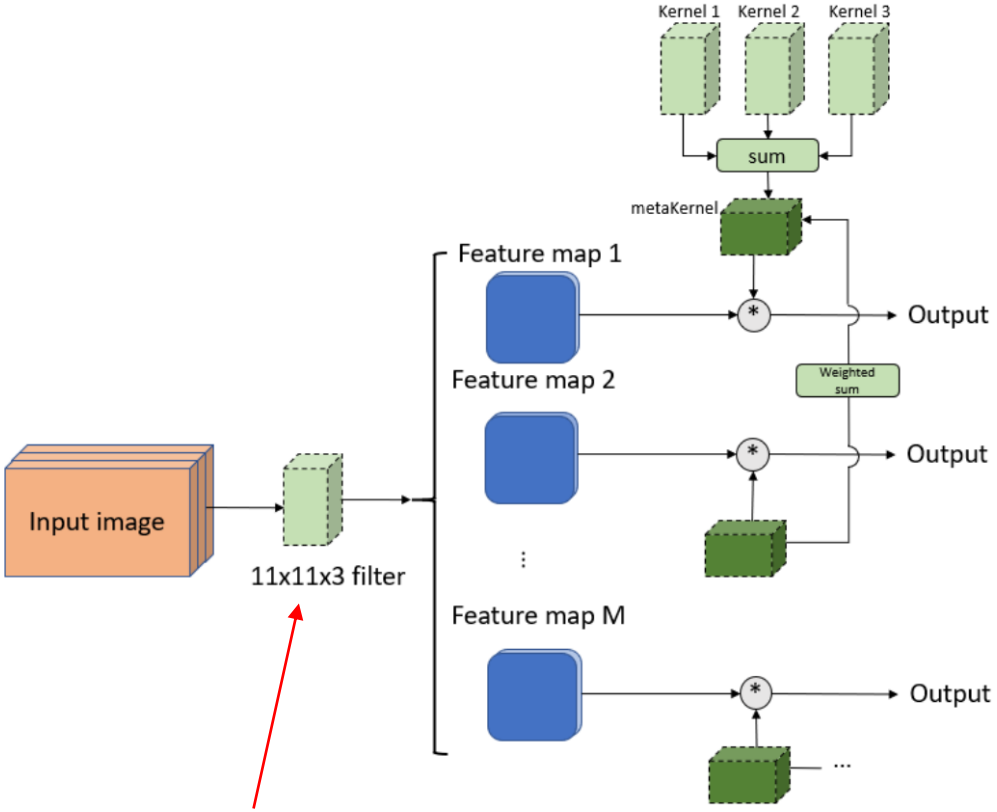
Solution:

Convert the design process into an optimization problem that can be minimized through SGD (Stochastic Gradient Descent). Optimization function attends to two main aspects: Latency and FLOPs of the candidate architecture.

Number of operations is minimized through the usage of Meta-Kernels that merge several convolutional operations → Additivity Property

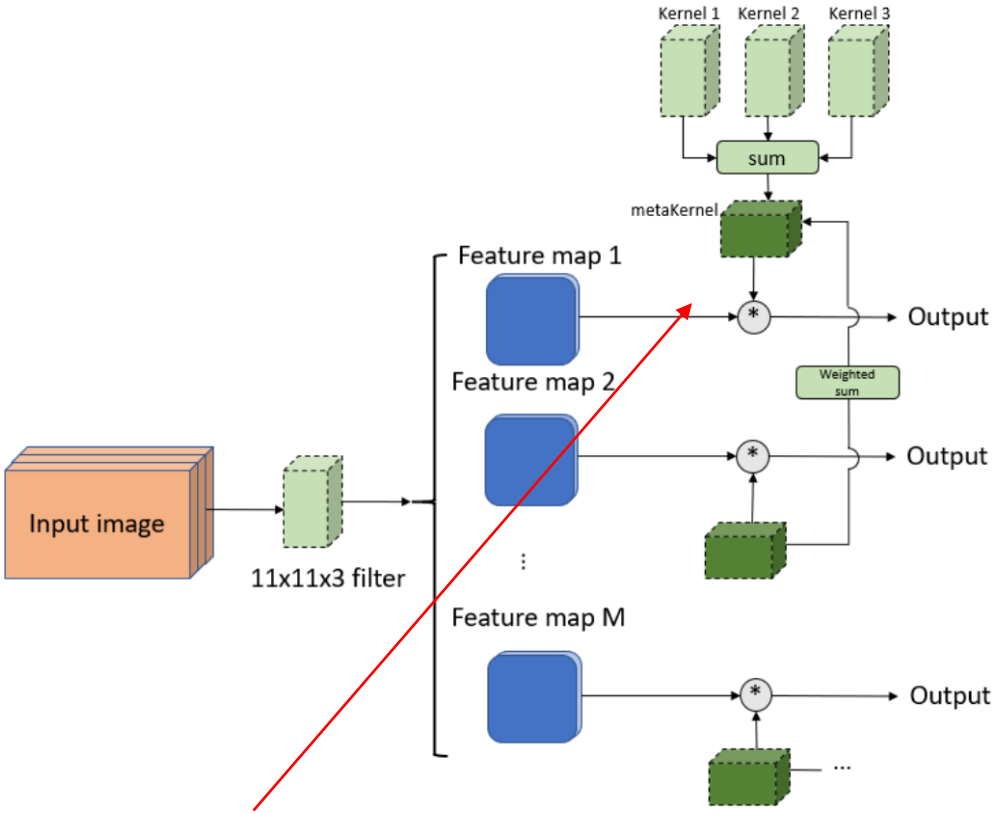
$$K_1 * I + K_2 * I = (K_1 + K_2) * I$$

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems
ICPR 2020 #1294



First step: separable depthwise convolution with a 11 x 11 kernel applied on the input image that leads to a reduced parameter size and computational cost

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems
ICPR 2020 #1294



Second step: Meta-kernels as a sum of 3 x 3, 5 x 5 and 7 x 7 filters applied on the output feature maps of the first step (Additivity property of the convolution)

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems
ICPR 2020 #1294

The search space:

we define the search space of each output $x^{(j)}$ as the combination of operations $o^{(i,j)}$ applied on inputs $x^{(i)}$, assuming the inputs as the outputs of the previous two layers

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)})$$

$o^{(i,j)}$ is the set of candidate operations (e.g. pooling, dilated convolutions, etc.). To make the search space differentiable, we approximate each operation and formulate it as:

$$\overline{o}^{(i,j)}(x) = \sum_{o \in O} \frac{exp(\alpha_o{}^{(i,j)})}{\sum_{o' \in O} exp(\alpha_{o'}{}^{(i,j)})} o(x)$$

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems
ICPR 2020 #1294

Through this, and taking into consideration that any operation is a combination of and Input (**I**) and a Kernel (**K**) or filter

$$o(x) = \boldsymbol{I} * \boldsymbol{K}^{(i)}$$

the proposed search network algorithm aims to learn jointly the architecture "α" and the weights "*w*" by optimizing the following loss function.

$$L(a, w_a) = CE(a, w_a) + \beta L_{LAT(a)}$$

where $CE(a;w_a)$ is the cross-entropy loss of the network candidate a with weights w_a and LAT(a) is the measured latency of network candidate a in microseconds

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems
ICPR 2020 #1294

The feedback blocks:

To shorten the search time, we propose the implementation of a feedback-block during training, which is the weighted sum of the learned meta-kernels being trained in parallel.

On each iteration the closer kernel to the "expected" one has more influence.

$$\boldsymbol{K_1}' = \boldsymbol{K_2}' = \beta_1 * \boldsymbol{K_1} + \beta_2 * \boldsymbol{K_2}$$

$$\beta_1 = \frac{\tanh \frac{1}{L_1}}{\tanh \frac{1}{L_1} + \tanh \frac{1}{L_2}}$$

$$\beta_2 = \frac{\tanh \frac{1}{L_2}}{\tanh \frac{1}{L_1} + \tanh \frac{1}{L_2}}$$

$$with \quad \beta_1 + \beta_2 = 1.$$

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems
ICPR 2020 #1294

The search method:

Algorithm 1: The search architecture methodology

Result: Find weights w_i and architecture probability parameters α to optimize the global loss function (13), given a defined search space with a combination of operations $\overline{o}^{(i,j)}$, defined in Eq. (9), a latency budget and an input dataset.

random initialization of α parameters

while *not converge* **do**

- Similar to [10], we generate the kernel candidates.
- Calculate Loss through Eq. (13).
- Calculate $\partial L/\partial w_a$ and $\partial L/\partial \alpha$.
- Update weights and architecture probability parameters α.
- Update Kernels using Eq. 3 and Eq. 5.

end

Extract more optimal architecture from learned α parameters.

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems ICPR 2020 #1294

Experiments:

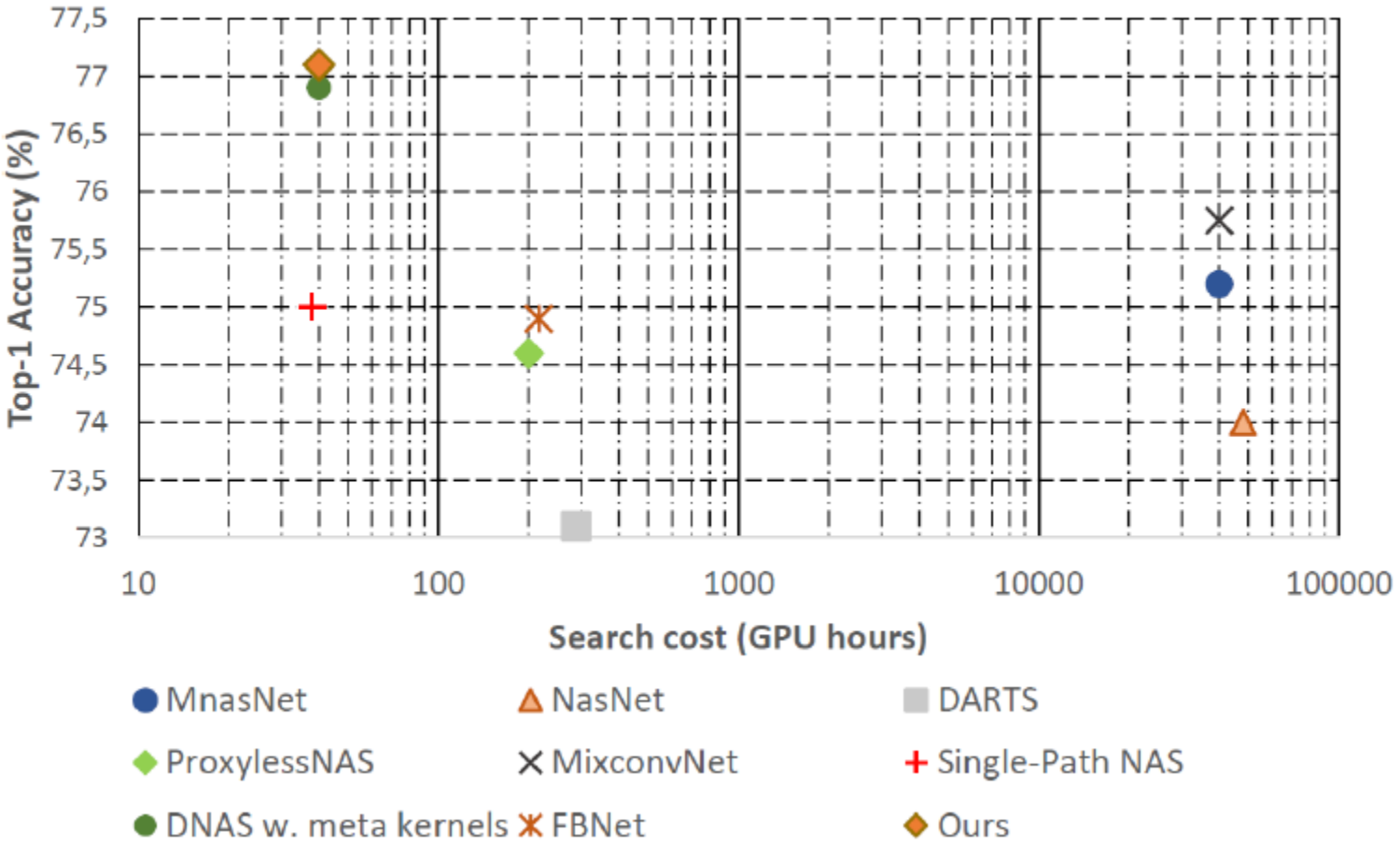
We have conducted experiments on the commonly used ImageNet and PascalVOC benchmarks.

We have trained the models using 8 GPU NVIDIA Tesla V100. Several implementation tricks have been implemented to improve the training process, such as the following:

- Random crop of a rectangular region.
- Normalize RGB channels.
- Randomly sample an image and decode it into 32-bit floating point raw pixel values in [0,255].

E-DNAS: Differentiable Neural Architecture Search for Embedded Systems
ICPR 2020 #1294

Experiments:



E-DNAS: Differentiable Neural Architecture Search for Embedded Systems
ICPR 2020 #1294

Model	# Params (M)	MACs (M)	Time (ms)
MNet [2]	4.2	569	75
NasNet-A [5]	5.3	564	183
Ours	**5.9**	**535**	**38**

Conclusions:

1. We propose a two-step pipeline that learns different meta-kernel sizes, able to treat different resolution patterns to create automatic neural networks that can classify images.
2. One of the main contributions is the proposed **circular feedback** on each iteration to speed up the process
3. We test out method with commercial hardware used in the automotive industry obtaining good results in terms of Accuracy and search time