Optimizing implementation of CNN inferences:

change the model or the architecture?

Alexandre Honorat

April 8, 2021

INSA Rennes/CNRS, IETR - UMR 6164

Plan

The problem of CNN hyperparameters

What architecture to consider for CNN?

What if CNN have dynamic parts?

Is it better to change the model or the architecture?

The problem of CNN hyperparameters

Tons of parameters...

AlexNet: ≈ 1.4 millions of parameters

Parameters coming from neuron weights, deduced from hyperparameters (#layers and their type, #neurons, conv. sizes)

More parameters $\rightarrow$ more computations

- because training might converge slowly or overfit
- **because more data must be stored and more Mult-Add are needed**

Ref: Understanding deep learning requires rethinking generalization, 2017, quoted 2659 (Google) times

Tons of parameters...

AlexNet: ≈ 1.4 millions of parameters

Parameters coming from neuron weights, deduced from hyperparameters (#layers and their type, #neurons, conv. sizes)

More parameters $\rightarrow$ more computations

- because training might converge slowly or overfit
- **because more data must be stored and more Mult-Add are needed**

Ref: Understanding deep learning requires rethinking generalization, 2017, quoted 2659 (Google) times

Tons of parameters...

AlexNet: ≈ 1.4 **millions of parameters**

Parameters coming from neuron weights, deduced from hyperparameters (#layers and their type, #neurons, conv. sizes)

More parameters $\rightarrow$ more computations

- because training might converge slowly or overfit
- **because more data must be stored and more Mult-Add are needed**

Ref: Understanding deep learning requires rethinking generalization, 2017, quoted 2659 (Google) times

How to reduce the number of parameters? Pruning!

Pruning what?

Q1. Prune connections, neurons, layers?

Q2. With what objective?

$\Rightarrow$ very long if pruning during learning

Ref: Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks, 2021
Ref: Automated Design of Deep Neural Networks: A Survey and Unified Taxonomy, ACM Comput. Surv. 2021
Ref: Gradient-Based Hyperparameter Optimization through Reversible Learning, ICML 2015, quoted 27 (ACM) or 483 (Google) times

How to reduce the number of parameters? Pruning!

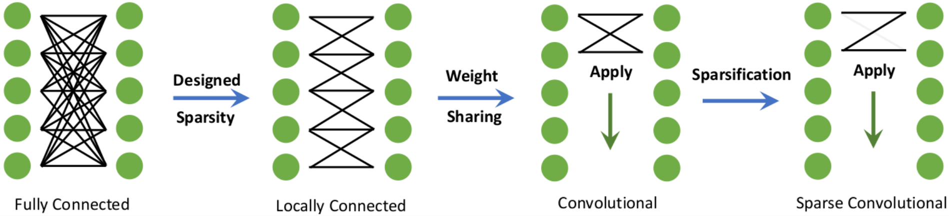
Pruning what?

Q1. Prune connections, neurons, layers?

Q2. With what objective?

⇒ very long if pruning during learning

Convolutions already are reducing the number of parameters!



©Sparsity in Deep Learning(...), Figure 3

Ref: Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks, 2021
Ref: Automated Design of Deep Neural Networks: A Survey and Unified Taxonomy, ACM Comput. Surv. 2021
Ref: Gradient-Based Hyperparameter Optimization through Reversible Learning, ICML 2015, quoted 27 (ACM) or 483 (Google) times

Pruning the parameters or their related values?

Quantization and weight sharing

Select numerical representation with less bits (after training).

e.g. `char` instead of `long`

Ref: Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding, ICLR 2016, quoted 4963 (Google) times

Pruning the parameters or their related values?

Quantization and weight sharing

Select numerical representation with less bits (after training).

e.g. `char` instead of `long`

Network	Top-1 Error	Top-5 Error	Parameters	Compress Rate
LeNet-300-100 Ref	1.64%	-	1070 KB	
LeNet-300-100 Compressed	1.58%	-	**27 KB**	**40**×
LeNet-5 Ref	0.80%	-	1720 KB	
LeNet-5 Compressed	0.74%	-	**44 KB**	**39**×
AlexNet Ref	42.78%	19.73%	240 MB	
AlexNet Compressed	42.78%	19.70%	**6.9 MB**	**35**×
VGG-16 Ref	31.50%	11.32%	552 MB	
VGG-16 Compressed	31.17%	10.91%	**11.3 MB**	**49**×

©Deep Compression(...), Table 1

Ref: Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding, ICLR 2016, quoted 4963 (Google) times

Do not forget hyperparameters and network topology!

Reorder the computations of convolution layers

Reduce data movements by splitting the convolutions.

Ref: Accelerating Very Deep Convolutional Networks for Classification and Detection, IEEE Trans. PAMI 2016, quoted 211 (IEEE) or 467 (Google) times

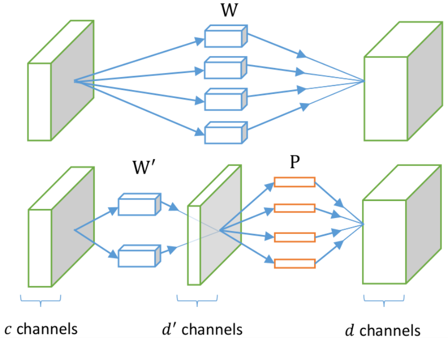
Ref: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, rejected at ICLR 2017, quoted 3719 (Google) times

Ref: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017 quoted 7791 (Google) times

Do not forget hyperparameters and network topology!

Reorder the computations of convolution layers

Reduce data movements by splitting the convolutions.



©Accelerating Very Deep Convolution Networks(...), Figure 1

Examples

- SqueezeNet
- MobiletNets

Problem

Requires linear separability.

Ref: Accelerating Very Deep Convolutional Networks for Classification and Detection, IEEE Trans. PAMI 2016, quoted 211 (IEEE) or 467 (Google) times

Ref: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, rejected at ICLR 2017, quoted 3719 (Google) times

Ref: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017 quoted 7791 (Google) times

What architecture to consider for CNN?

What architectures?

CNN are heavily data-parallel, slightly task-parallel

The number of pixels in images usually exceeds the number of processing elements.
If feed-forward, CNN can be considered layer by layer.

Architectures

- CPU ≈ (task parallelism, not enough cores)
- GPU ++ (data parallelism, perfect)
- **FPGA** + (streamed data parallelism + task parallelism, but not enough memory)
- ASIC ++ (whatever you want, but costly)
- MPSoC+ (quantity of CPU/GPU/... is predefined)

Ref: Combining Task- and Data-Level Parallelism for High-Throughput CNN Inference on Embedded CPUs-GPUs MPSoCs, SAMOS 2020
Ref: AMAIX: A Generic Analytical Model for Deep Learning Accelerators, SAMOS 2020

What architectures?

CNN are heavily data-parallel, slightly task-parallel

The number of pixels in images usually exceeds the number of processing elements.
If feed-forward, CNN can be considered layer by layer.

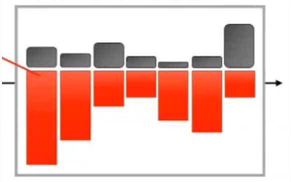
Architectures

- CPU ≈ (task parallelism, not enough cores)
- GPU ++ (data parallelism, perfect)
- **FPGA** + (streamed data parallelism + task parallelism, but not enough memory)
- ASIC ++ (whatever you want, but costly)
- MPSoC+ (quantity of CPU/GPU/... is predefined)

Ref: Combining Task- and Data-Level Parallelism for High-Throughput CNN Inference on Embedded CPUs-GPUs MPSoCs, SAMOS 2020
Ref: AMAIX: A Generic Analytical Model for Deep Learning Accelerators, SAMOS 2020

The case of FPGA: why not so easy?

Easier to pipeline if layers have balanced computations.
(otherwise we might merge them)



©Tutorial ISFPGA'21: Neural Network Accelerator Co-Design with FINN (2 min 06 sec)

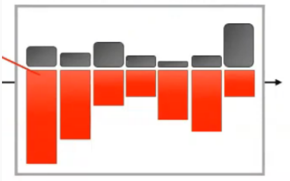
FINN-R framework of Xilinx

- claim higher throughput than GPU (not higher energy efficiency?)
- claim easy reconfiguration (scaling) when changing of FPGA
- intrinsic quantization

Ref: FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks, ACM Trans. Reconfigurable Technol. Syst. 2018, quoted 32 (ACM) or 83 (Google) times

The case of FPGA: why not so easy?

Easier to pipeline if layers have balanced computations.
(otherwise we might merge them)



©Tutorial ISFPGA'21: Neural Network Accelerator Co-Design with FINN (2 min 06 sec)

FINN-R framework of Xilinx

- claim higher throughput than GPU (not higher energy efficiency?)
- claim easy reconfiguration (scaling) when changing of FPGA
- intrinsic quantization

Ref: FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks, ACM Trans. Reconfigurable Technol. Syst. 2018, quoted 32 (ACM) or 83 (Google) times

The case of ~~FPGA~~: two examples

Extreme quantization

Binary Neural Networks have all weights on 1 bit.

Extreme (down)scaling

MobileNets can be easily adapted to constrained architectures by sacrificing accuracy.

Ref: FracBNN: Accurate and FPGA-Efficient Binary Neural Networks with Fractional Activations, FPGA 2021
Ref: Multi-Objective Autotuning of MobileNets across the Full Software/Hardware Stack, ReQuEST 2018

The case of ~~FPGA~~: data ordering

Matrix multiplication is already highly optimized...

...but it requires to copy and reorder the data (im2col)

$$\begin{pmatrix} k1 & k2 & 0 & k3 & k4 & 0 & 0 & 0 & 0 \\ 0 & k1 & k2 & 0 & k3 & k4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k1 & k2 & 0 & k3 & k4 & 0 \\ 0 & 0 & 0 & 0 & k1 & k2 & 0 & k3 & k4 \end{pmatrix} \cdot \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \\ x9 \end{pmatrix}$$

©StackOverflow: 2-D convolution as a matrix-matrix multiplication

Best reordering (tested on CPU)

kn2row seems to have better data locality.

Does it depend on the convolution shape? (symmetry, size)

Ref: Parallel Multi Channel convolution using General Matrix Multiplication, ASAP 2017, quoted 22 (IEEE) or 66 (Google) times

The case of ~~FPGA~~: data ordering

Matrix multiplication is already highly optimized...

...but it requires to copy and reorder the data (im2col)

$$\begin{pmatrix} k1 & k2 & 0 & k3 & k4 & 0 & 0 & 0 & 0 \\ 0 & k1 & k2 & 0 & k3 & k4 & 0 & 0 & 0 \\ 0 & 0 & 0 & k1 & k2 & 0 & k3 & k4 & 0 \\ 0 & 0 & 0 & 0 & k1 & k2 & 0 & k3 & k4 \end{pmatrix} \cdot \begin{pmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \\ x6 \\ x7 \\ x8 \\ x9 \end{pmatrix}$$

©StackOverflow: 2-D convolution as a matrix-matrix multiplication

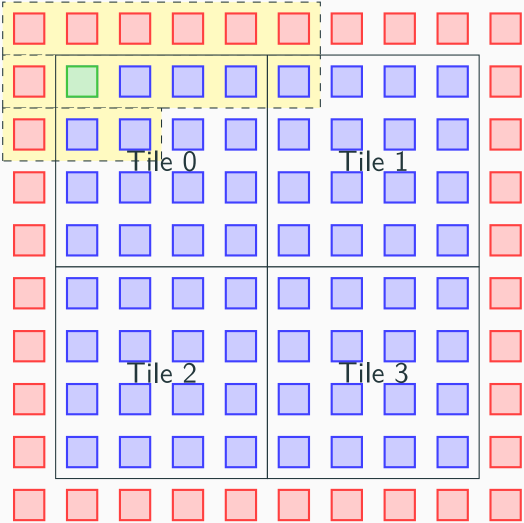
Best reordering (tested on CPU)

kn2row seems to have better data locality.

Does it depend on the convolution shape? (symmetry, size)

Ref: Parallel Multi Channel convolution using General Matrix Multiplication, ASAP 2017, quoted 22 (IEEE) or 66 (Google) times

The case of ~~FPGA~~: tiling + streamed convolution

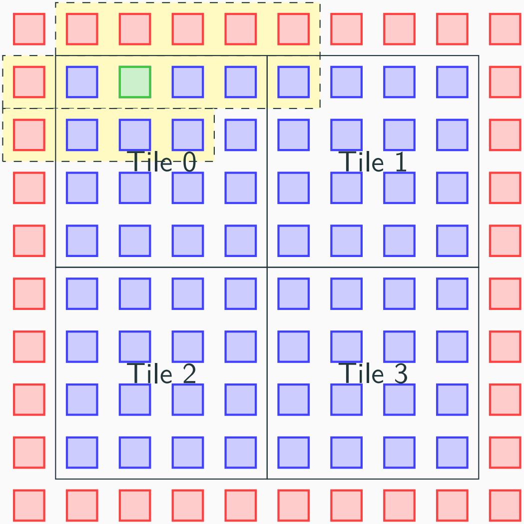


Input image is streamed and a 3x3 convolution is applied per tile.

Requires previous reorder but reduce FIFO sizes.

Ref: Algorithm Level Timing Speculation for Convolutional Neural Network Accelerators, FPT 2018

The case of ~~FPGA~~: tiling + streamed convolution

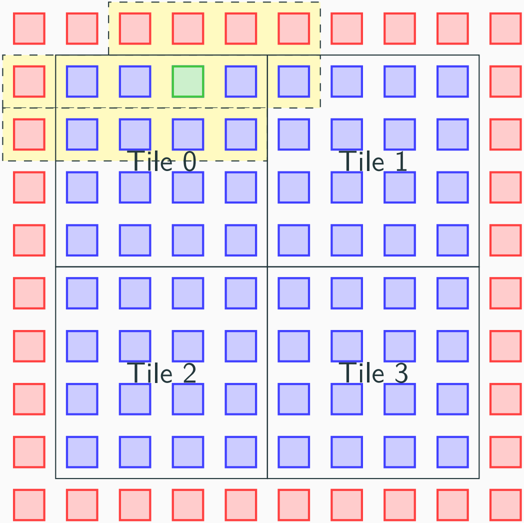


Input image is streamed and a 3x3 convolution is applied per tile.

Requires previous reorder but reduce FIFO sizes.

Ref: Algorithm Level Timing Speculation for Convolutional Neural Network Accelerators, FPT 2018

The case of ~~FPGA~~: tiling + streamed convolution

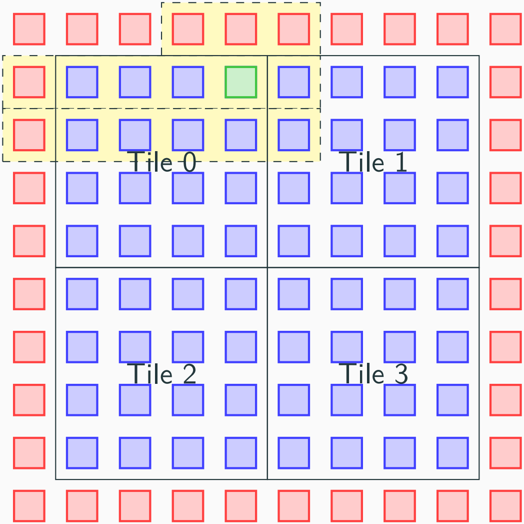


Input image is streamed and a 3x3 convolution is applied per tile.

Requires previous reorder but reduce FIFO sizes.

Ref: Algorithm Level Timing Speculation for Convolutional Neural Network Accelerators, FPT 2018

The case of ~~FPGA~~: tiling + streamed convolution

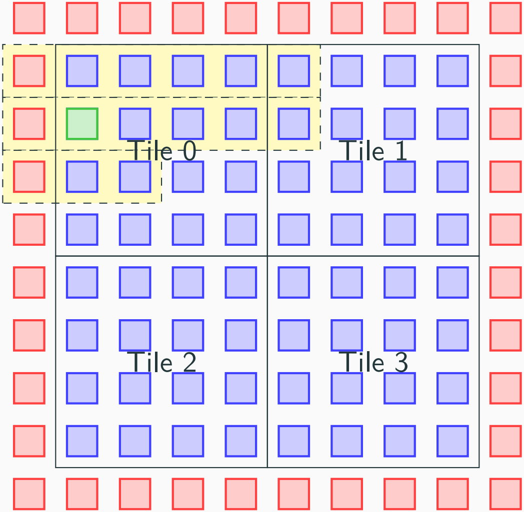


Input image is streamed and a 3x3 convolution is applied per tile.

Requires previous reorder but reduce FIFO sizes.

Ref: Algorithm Level Timing Speculation for Convolutional Neural Network Accelerators, FPT 2018

The case of ~~FPGA~~: tiling + streamed convolution



Input image is streamed and a 3x3 convolution is applied per tile.

Requires previous reorder but reduce FIFO sizes.

Ref: Algorithm Level Timing Speculation for Convolutional Neural Network Accelerators, FPT 2018

The case of ~~FPGA~~: specialized operators

Vendors know their circuits better

Even on FPGA, some circuits are hard-wired (as full-adder).

Specialized per layer type?

Proposed by FINN-R, probably not for full-connected layers nor wide convolutions.

Ref: The Challenge of Multi-Operand Adders in CNNs on FPGAs: How Not to Solve It!, SAMOS 2018

The case of ~~FPGA~~: specialized operators

Vendors know their circuits better

Even on FPGA, some circuits are hard-wired (as full-adder).

Specialized per layer type?

Proposed by FINN-R, probably not for full-connected layers nor wide convolutions.

Ref: The Challenge of Multi-Operand Adders in CNNs on FPGAs: How Not to Solve It!, SAMOS 2018

What if CNN have dynamic parts?

Dynamic filter pruning

Prune some filters during inference

Remove redundant ones or select the ones with more saliency.

Problem

P1. Modify the balance of computations between processing elements.

Ref: A dynamic CNN pruning method based on matrix similarity, Signal Image Video Process. 2021
Ref: Dynamic Channel Pruning: Feature Boosting and Suppression, ICLR 2019, quoted 88 (Google) times

Dynamic filter pruning

Prune some filters during inference

Remove redundant ones or select the ones with more saliency.

Problem

P1. Modify the balance of computations between processing elements.

Ref: A dynamic CNN pruning method based on matrix similarity, Signal Image Video Process. 2021
Ref: Dynamic Channel Pruning: Feature Boosting and Suppression, ICLR 2019, quoted 88 (Google) times

Dynamic filter selection

Select some filters during inference

Possibly even dependent on location in the image.

Problem

P1. Modify the balance of computations between processing elements.

P2. Suitable for GPU?

Ref: Dynamic Filter Networks, NIPS 2016, quoted 15 (ACM) or 156+345 (Google) times
Ref: DynCNN: An Effective Dynamic Architecture on Convolutional Neural Network for Surveillance Videos, rejected at ICLR'19

Dynamic filter selection

Select some filters during inference

Possibly even dependent on location in the image.

Problem

P1. Modify the balance of computations between processing elements.

P2. Suitable for GPU?

Ref: Dynamic Filter Networks, NIPS 2016, quoted 15 (ACM) or 156+345 (Google) times

Ref: DynCNN: An Effective Dynamic Architecture on Convolutional Neural Network for Surveillance Videos, rejected at ICLR'19

Pyramidal and dynamic pooling

Dynamic image size before k-max pooling

Input image size may vary for convolutions, but k is constant and keep spatial properties.

Dynamic k-max pooling

Sentence size varies and k depends on the input.

Problem

P1. Modify the balance of computations between processing elements.

Ref: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE Trans. PAMI 2015, quoted 2004 (IEEE) or 5884 (Google) times
Ref: A Convolutional Neural Network for Modelling Sentences, ACL 2014, quoted 3109 (Google) times

Pyramidal and dynamic pooling

Dynamic image size before k-max pooling

Input image size may vary for convolutions, but k is constant and keep spatial properties.

Dynamic k-max pooling

Sentence size varies and k depends on the input.

Problem

P1. Modify the balance of computations between processing elements.

Ref: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE Trans. PAMI 2015, quoted 2004 (IEEE) or 5884 (Google) times

Ref: A Convolutional Neural Network for Modelling Sentences, ACL 2014, quoted 3109 (Google) times

Pyramidal and dynamic pooling

Dynamic image size before k-max pooling

Input image size may vary for convolutions, but k is constant and keep spatial properties.

Dynamic k-max pooling

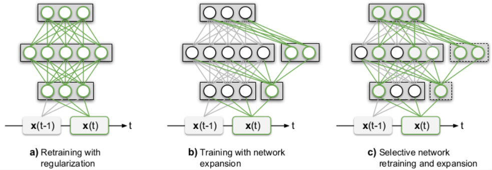
Sentence size varies and k depends on the input.

Problem

P1. Modify the balance of computations between processing elements.

Ref: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, IEEE Trans. PAMI 2015, quoted 2004 (IEEE) or 5884 (Google) times
Ref: A Convolutional Neural Network for Modelling Sentences, ACL 2014, quoted 3109 (Google) times

What if the network topology is dynamic?



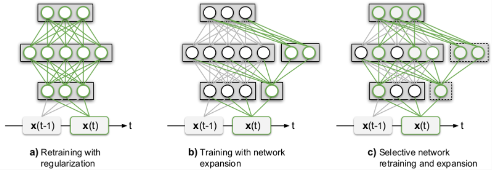
©Continual lifelong learning(...), Figure 2

Problem

Nothing is constant anymore...

Ref: Continual lifelong learning with neural networks: A review, NN 2019, quoted 210 (Elsevier) or 693 (Google) times

What if the network topology is dynamic?



©Continual lifelong learning(...), Figure 2

Problem

Nothing is constant anymore...

Ref: Continual lifelong learning with neural networks: A review, NN 2019, quoted 210 (Elsevier) or 693 (Google) times

Is it better to change the model or the architecture?

Conclusion: better to change model and architecture

ML project management

Architecture selection comes at stage 5 out of 10.

Options to not change the hardware

- everything is static and data parallel
- narrow convolutions even if adding more layers

Otherwise

Test and retry! (probably layer per layer)

Ref: Technology Readiness Levels for Machine Learning Systems, 2021

Conclusion: better to change model and architecture

ML project management

Architecture selection comes at stage 5 out of 10.

Options to not change the hardware

- everything is static and data parallel
- narrow convolutions even if adding more layers

Otherwise

Test and retry! (probably layer per layer)

Ref: Technology Readiness Levels for Machine Learning Systems, 2021

Conclusion: better to change model and architecture

ML project management

Architecture selection comes at stage 5 out of 10.

Options to not change the hardware

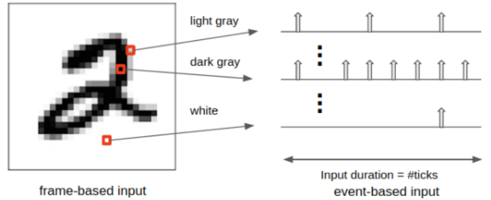
- everything is static and data parallel
- narrow convolutions even if adding more layers

Otherwise

Test and retry! (probably layer per layer)

Ref: Technology Readiness Levels for Machine Learning Systems, 2021

Although being time dependent, SNN support convolutions!



©S2N2(...), Figure 1

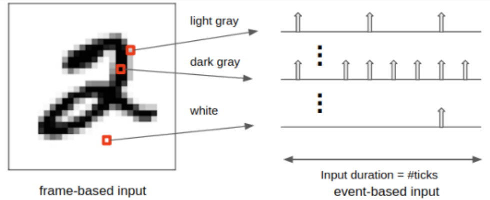
For what architecture?

Initially ASICs, and now FPGA (using FINN-R).

Ref: Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition, Int. J. Comput. Vis. 2015, quoted 233 (Springer) or 356 (ResearchGate) times
Ref: S2N2: A FPGA Accelerator for Streaming Spiking Neural Networks, FPGA 2021

Opening 1: Spiking Neural Networks

Although being time dependent, SNN support convolutions!



©S2N2(...), Figure 1

For what architecture?

Initially ASICs, and now FPGA (using FINN-R).

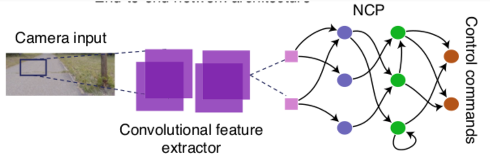
Ref: Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition, Int. J. Comput. Vis. 2015, quoted 233 (Springer) or 356 (ResearchGate) times
Ref: S2N2: A FPGA Accelerator for Streaming Spiking Neural Networks, FPGA 2021

Opening 2: NN with Ordinary Differential Equations

NCP are still time dependent, but after a regular CNN!

$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)A. \quad (1)$$

©Liquid Time-constant Networks, Equation 1



©Neural circuit policies(...), Figure 3

For what architecture?

Not specified (uses Tesla Hydranet), but maybe CNN on GPU and NCP elsewhere?

Ref: Liquid Time-constant Networks, 2020

Ref: Neural circuit policies enabling auditable autonomy, Nature Machine Intelligence 2020

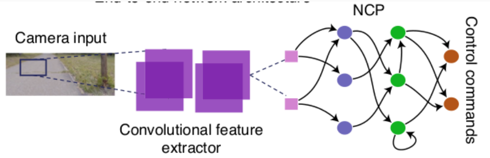
Ref: HydraNets: Specialized Dynamic Architectures for Efficient Inference, CVPR 2018, quoted 47 (Google) times

Opening 2: NN with Ordinary Differential Equations

NCP are still time dependent, but after a regular CNN!

$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), t, \theta)A. \tag{1}$$

©Liquid Time-constant Networks, Equation 1



©Neural circuit policies(...), Figure 3

For what architecture?

Not specified (uses Tesla Hydranet), but maybe CNN on GPU and NCP elsewhere?

Ref: Liquid Time-constant Networks, 2020
Ref: Neural circuit policies enabling auditable autonomy, Nature Machine Intelligence 2020
Ref: HydraNets: Specialized Dynamic Architectures for Efficient Inference, CVPR 2018, quoted 47 (Google) times

Questions?