ALMA MATER STUDIORUM - UNIVERSITY OF BOLOGNA
Scuola di Ingegneria e Architettura

Compression of Convolutional Neural Networks using Tensor Decomposition

Candidato:
Ali Alessio SALMAN

Relatore:
Prof. Stefano MATTOCCIA

Correlatori:
Dott. Matteo POGGI
Dott. Fabio TOSI

Convolutional Neural Networks (CNN)

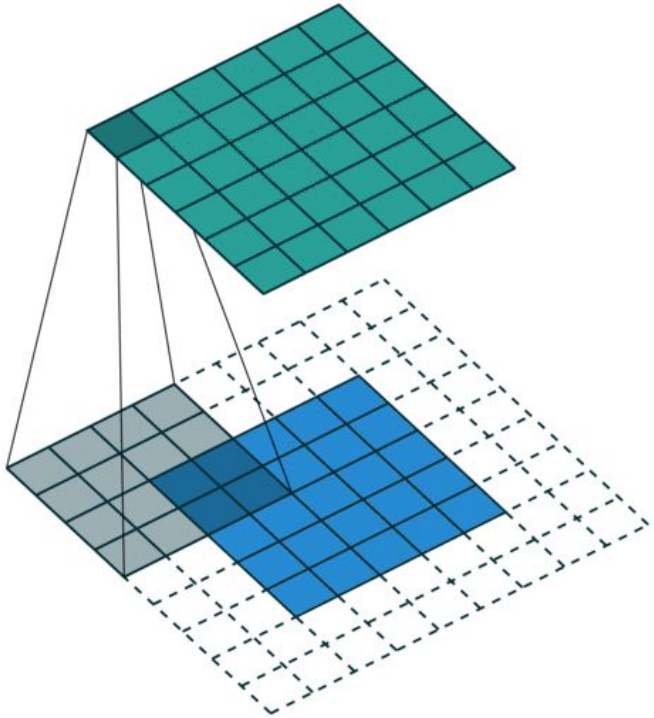
Rete neurale con un’architettura particolarmente vantaggiosa per compiti di tipo visivo

Combina:

- Rete profonda
- Operazione di convoluzione

Stato dell'arte in:

- Classificazione
- Object Detection
- Riconoscimento facciale
- …



Perché comprimere una CNN?

- Numero di parametri elevatissimo (ad es. VGG-16 500MB)
- Notevoli capacità computazionali necessarie → training solo su GPU, lento
- Accelerazione difficile su dispositivi low-end (e.g. FPGA, ASIC, ...)
- Ampie possibilità di applicazione (Mobile devices, Drones, self-driving cars)

Tecniche di riduzione di dimensionalità

PRO:

- Velocità di training e d'inferenza
- Dimensioni molto ridotte

CONS:

Potenziale perdita di accuratezza

Convolutional Layer – Tensore 4D

Un layer di convoluzione può essere visto come tensore a 4 dimensioni:

$$W = [S \times T \times d \times d]$$

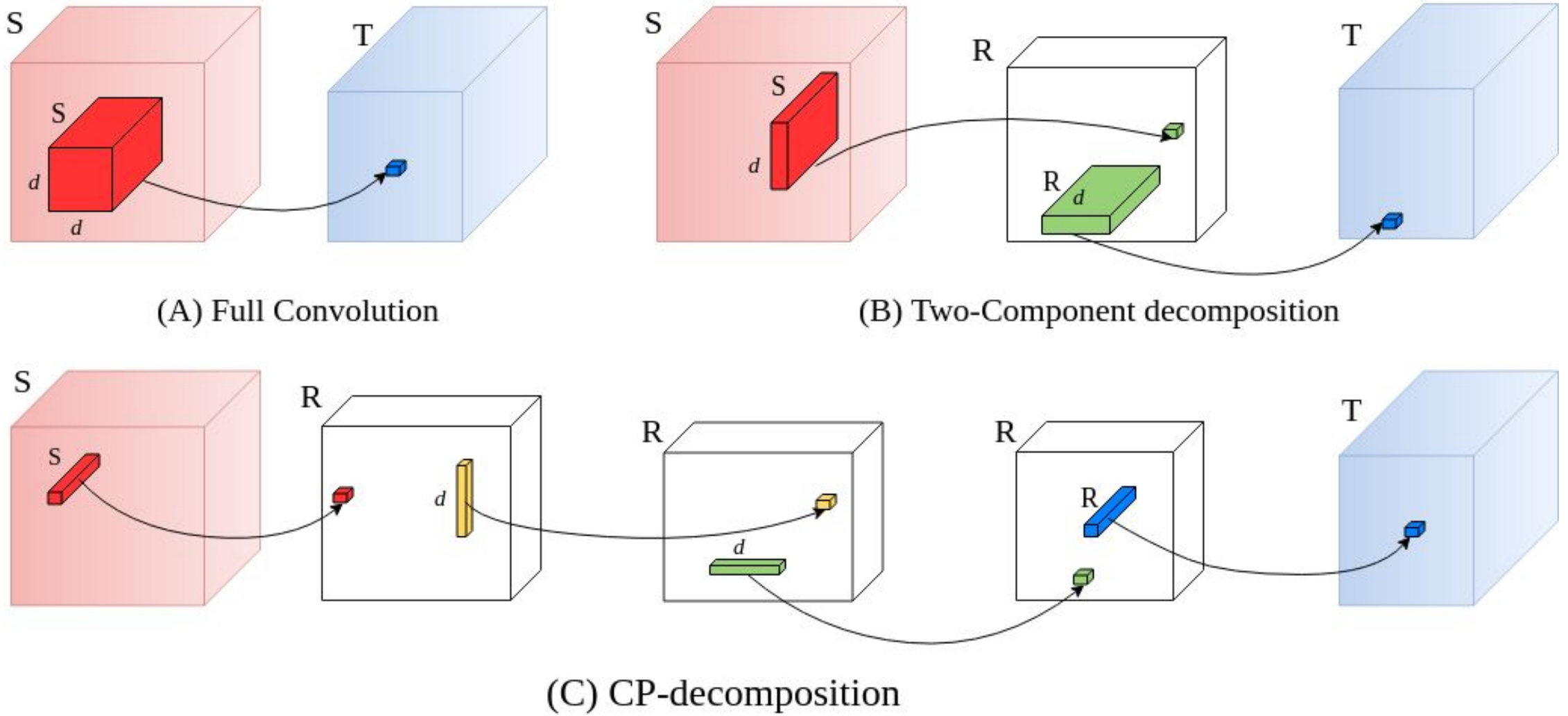
- S = numero di feature maps in ingresso
- T = numero di feature maps in uscita
- [d, d] è la dimensione del filtro di convoluzione quadrato

⇒ Decomposizione tensoriale può essere applicata a **W**

Obiettivo:

1) In ingresso una CNN classica con numero di parametri pari ad X
2) Applicazione di tecniche di decomposizione tensoriale
3) In uscita: CNN con Y parametri, con Y << X

CPD - layer di convoluzione



Micro-architettura

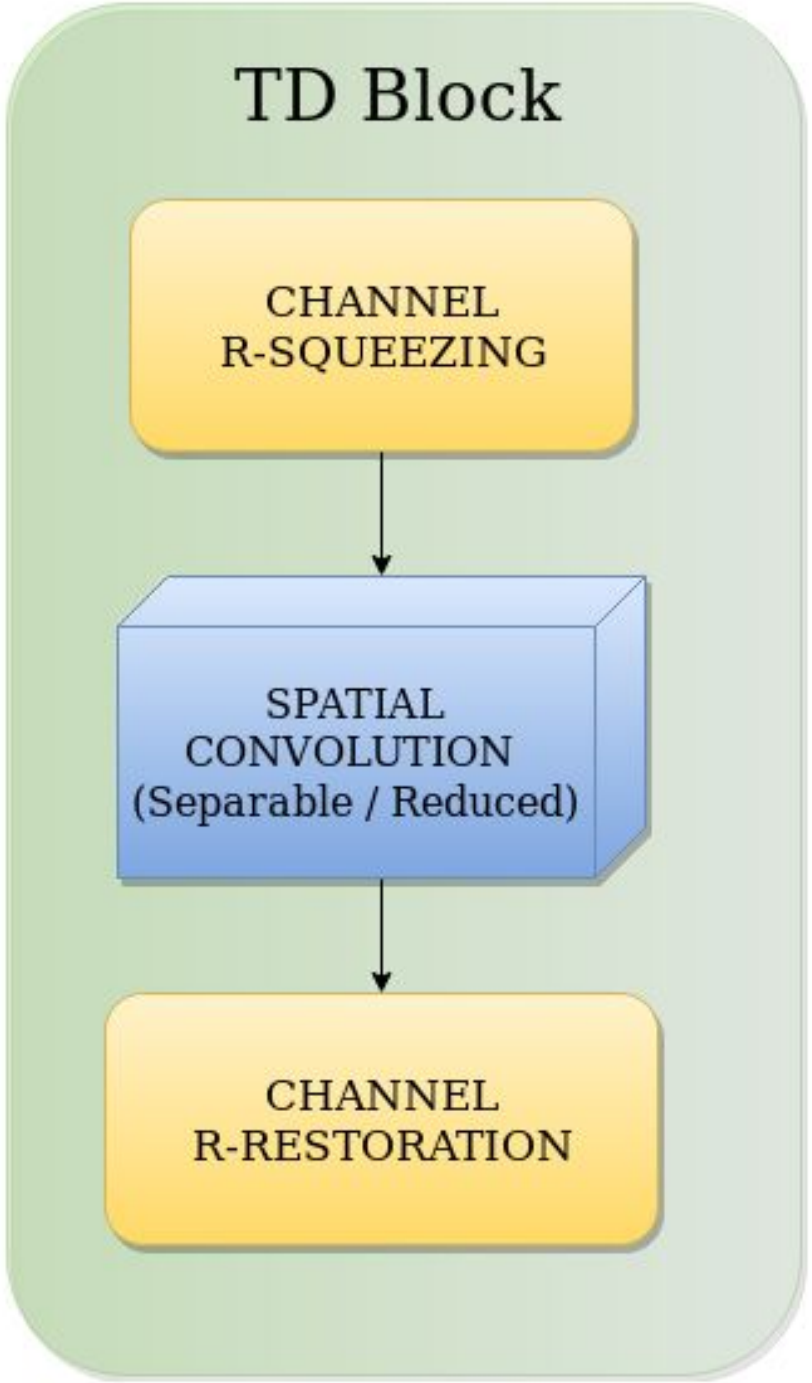
TD Block: un nuovo design

Parametri iniziali = $S \times T \times d^2$

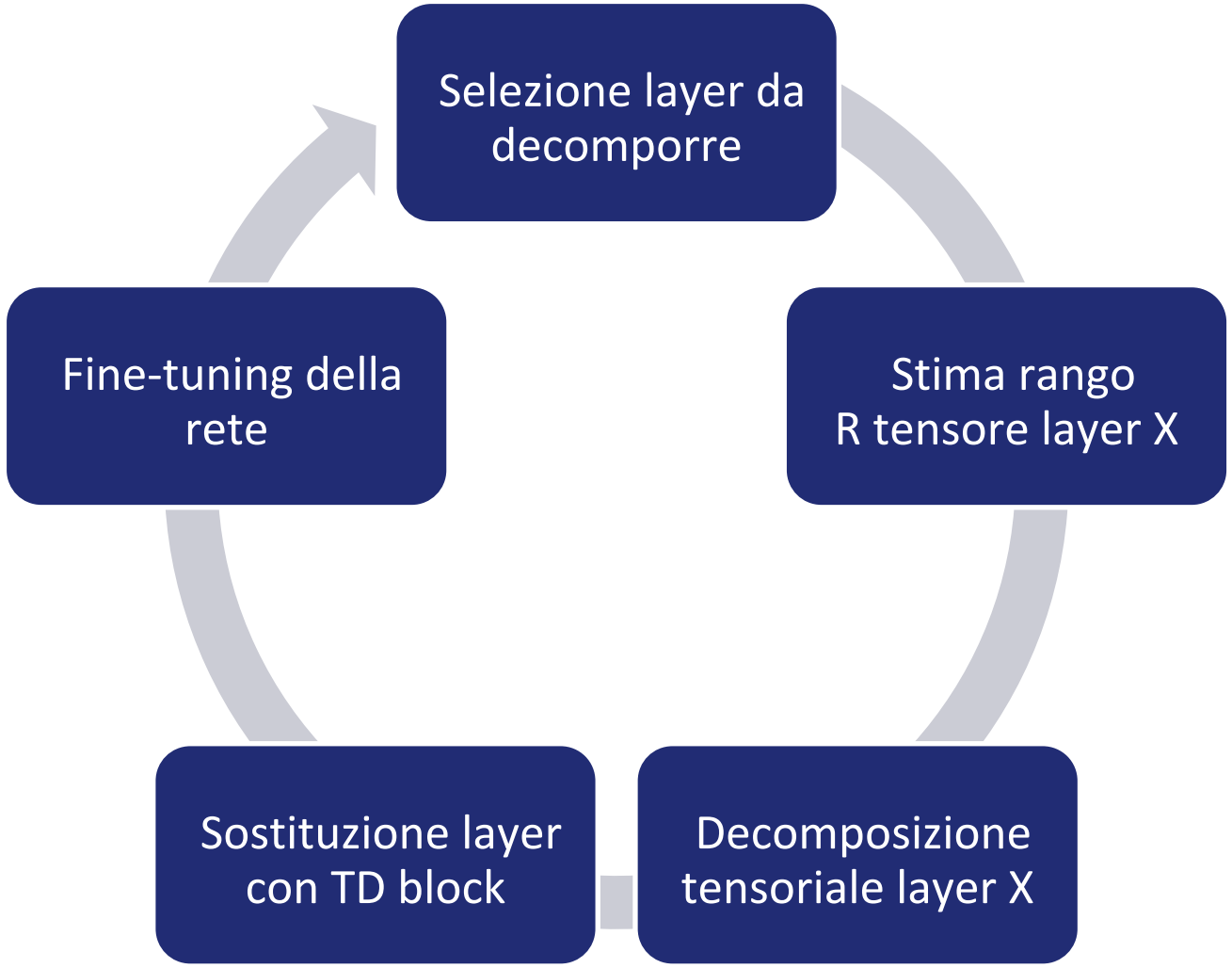
Parametri **dopo** la decomposizione

CPD	$R(S + 2d + T)$
Tucker	$R_3 + R_3 R_4 d^2 + R_4 T$

$$\mathcal{C}_r = \frac{STd^2}{R(S + 2d + T)}$$



Pipeline di decomposizione di una CNN



TD Block: Model Design

- **LeNet** 4 Conv layer + 2 FC

Dataset: CIFAR10: 60K immagini a colori divise in 10 categorie

- **CCNN** - 7 Conv layer

Dataset: KITTI 193 paia di immagini per algoritmi di matching stereo

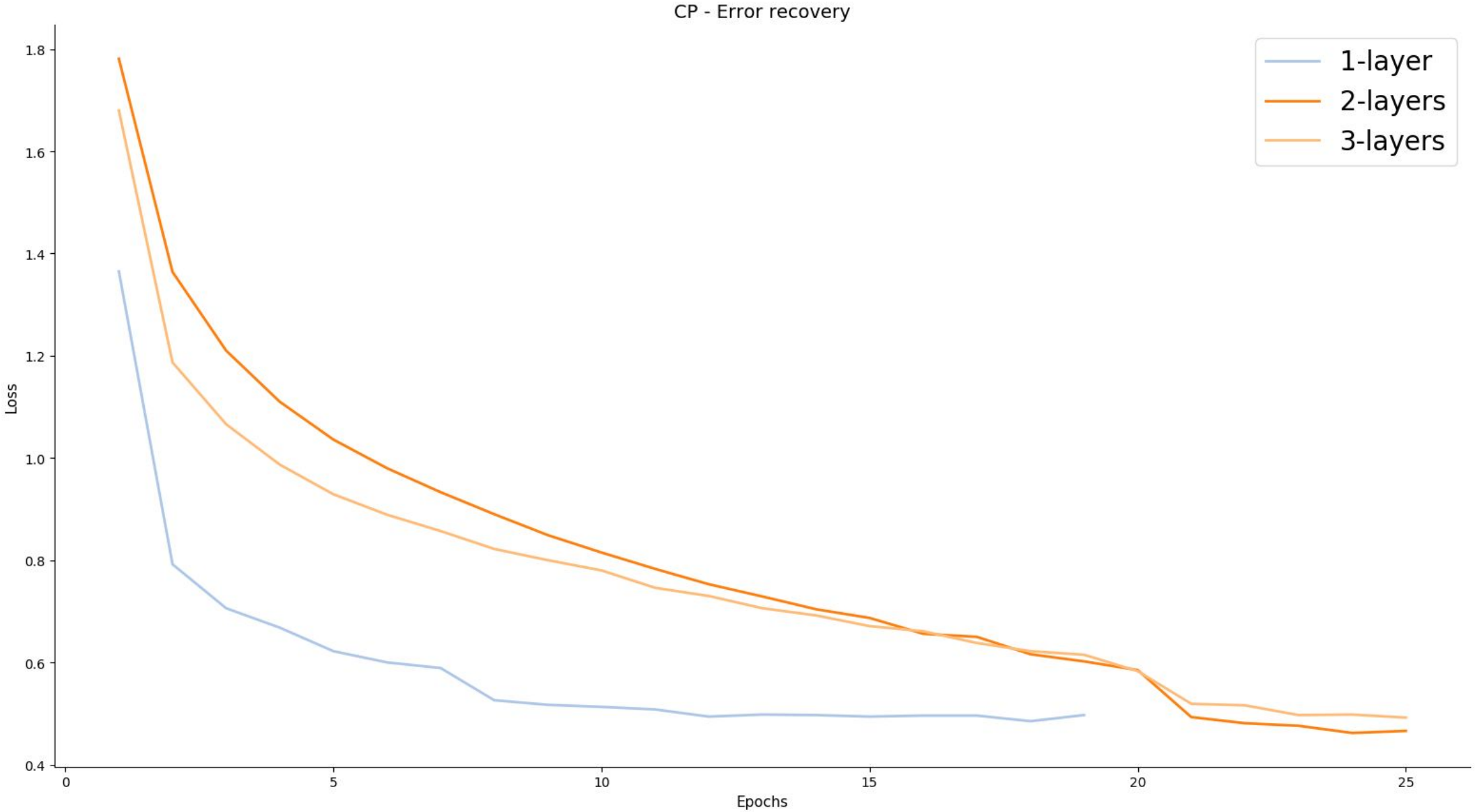
TD Block: Model Compression

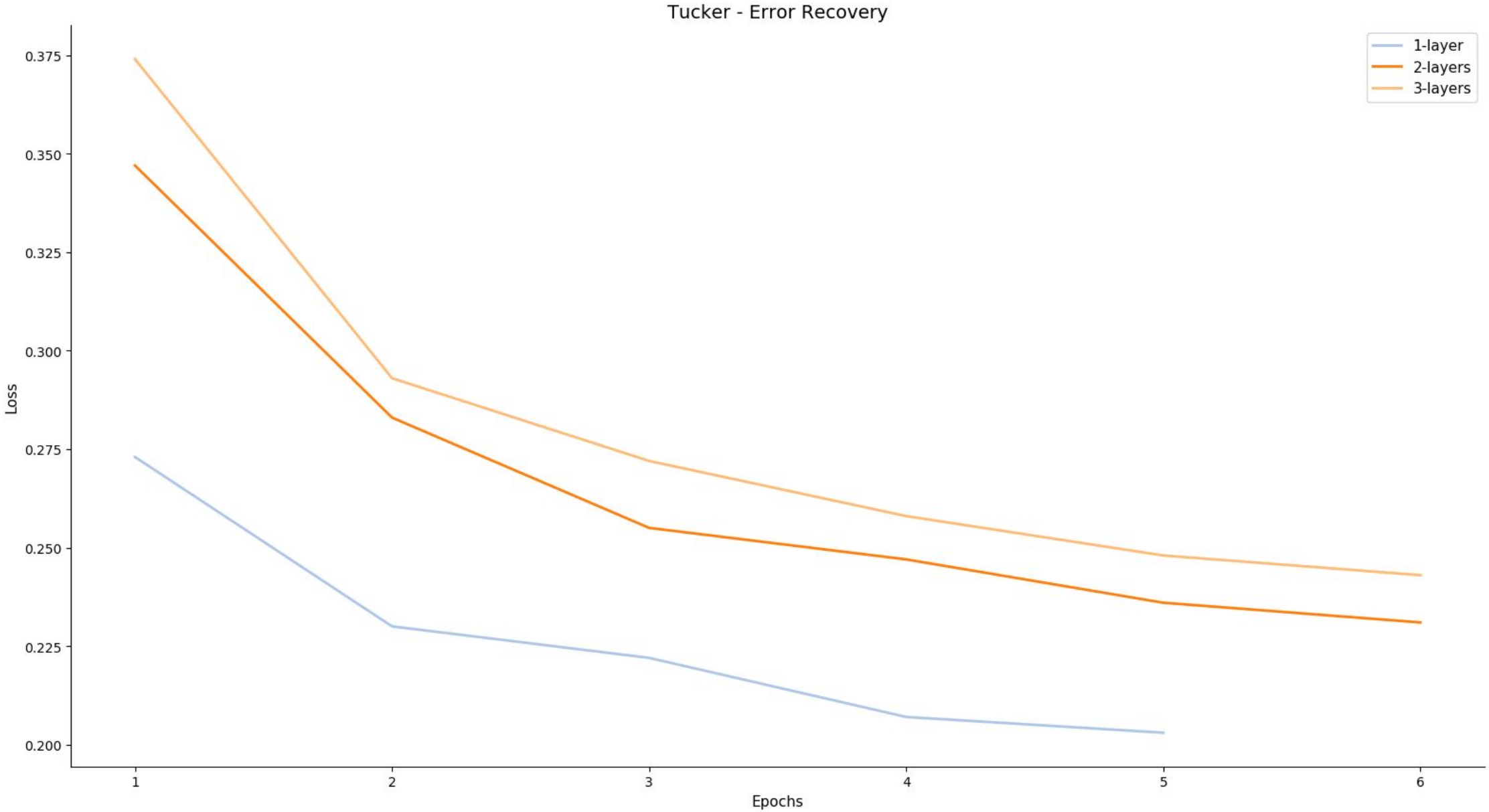
- **LeNet 3 Conv layer + 3 FC**

Dataset: CIFAR10

- **Network-In-Network: All Conv**

Dataset: CIFAR10


CP - Error recovery
1-layer
2-layers
3-layers
Loss
Epochs
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
5
10
15
20
25


Tucker - Error Recovery
1-layer
2-layers
3-layers
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.200
Loss
1
2
3
4
5
6
Epochs

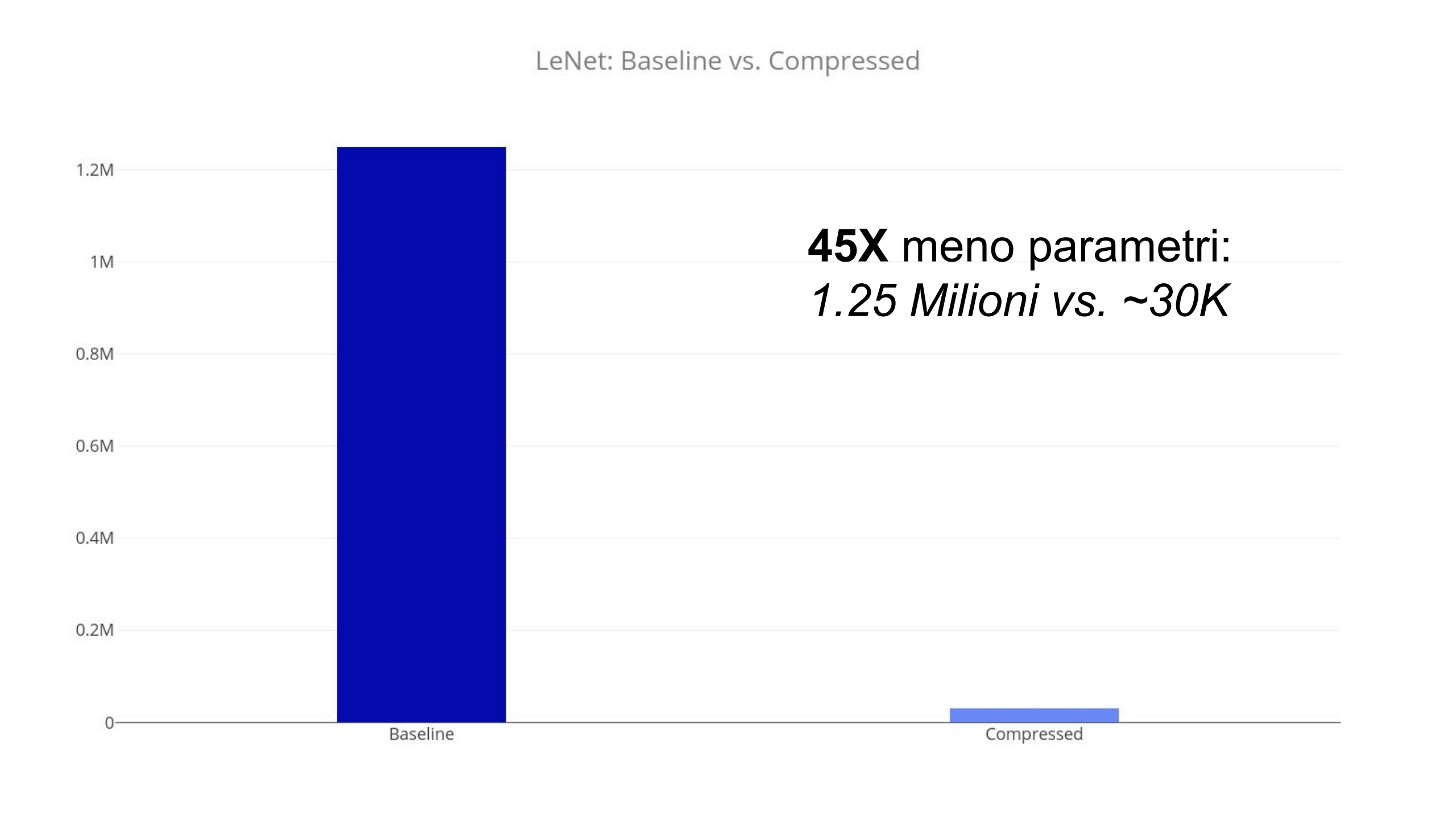
Risultati sperimentali

LAYER	***METODO***	***ACCURACY***	***COMPRESSION RATIO***
CONV1	CPD	80% (+6%)	4X
CONV2	CPD	82% (+8%)	31X
CONV3 R=85	CPD	80% (+6%)	25X
CONV3 R=39	**CPD**	**77 (+3%)**	**52X (Improves Zhang et al.)**
OVERALL	**CPD**	**81% (+7%)**	**4.5X**
---	---	---	---
CONV1	TUCKER	75% (+1%)	0.77X
CONV2	TUCKER	79% (+8%)	6.3X
CONV3	TUCKER	80% (+6%)	15X
CONV3 R3=6, R4=41	TUCKER	**76% (+2%)**	**63X (Improves Zhang et al.)**
OVERALL	**TUCKER**	**80% (+6%)**	**3.5X**

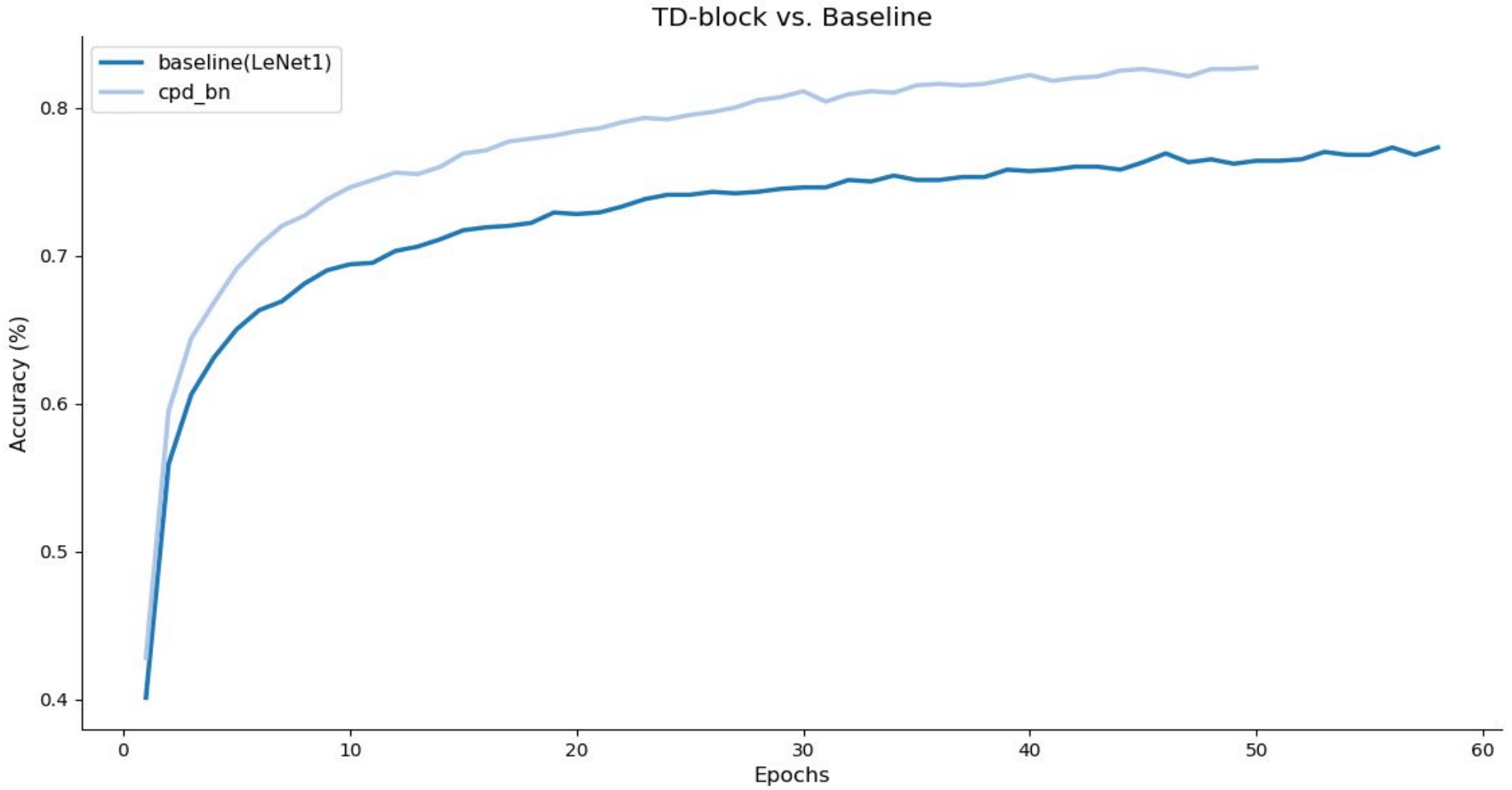
LeNet: Architettura TD Block

LAYER	CHARACTERISTICS
CONV	32 filters 3×3, padding=1, stride=1
ReLU	-
CONV	32 filters 3 × 3, padding=0 stride=1
ReLU	-
POOL	pool size [2, 2] stride=2
Dropout	p=0.25
CONV	64 filters 3 × 3, padding=1 stride=1
ReLU	-
CONV	64 filters 3 × 3, padding=0 stride=1
ReLU	-
POOL	pool size [2, 2] stride=2
Dropout	p=0.25
FC	512 units
FC	#Classes (=10) units

LAYER	CHARACTERISTICS
TD-Block	$32 \times [1,1] + R \times 3 + 3 \times R + 32 \times [1,1]$
ReLU	-
TD-Block	$32 \times [1,1] + R \times 3 + 3 \times R + 32 \times [1,1]$
ReLU	-
POOL	pool size [2, 2] stride=2
Dropout	p=0.25
TD-Block	$64 \times [1,1] + R \times 3 + 3 \times R + 64 \times [1,1]$
ReLU	-
TD-Block	$64 \times [1,1] + R \times 3 + 3 \times R + 64 \times [1,1]$
ReLU	-
POOL	pool size [2, 2] stride=2
Dropout	p=0.25
TD-Block	$64 \times [1,1] + R \times 6 + 6 \times R + 512 \times [1,1]$
TD-Block	$512 \times [1,1] + R \times 1 + 1 \times R + 10 \times [1,1]$

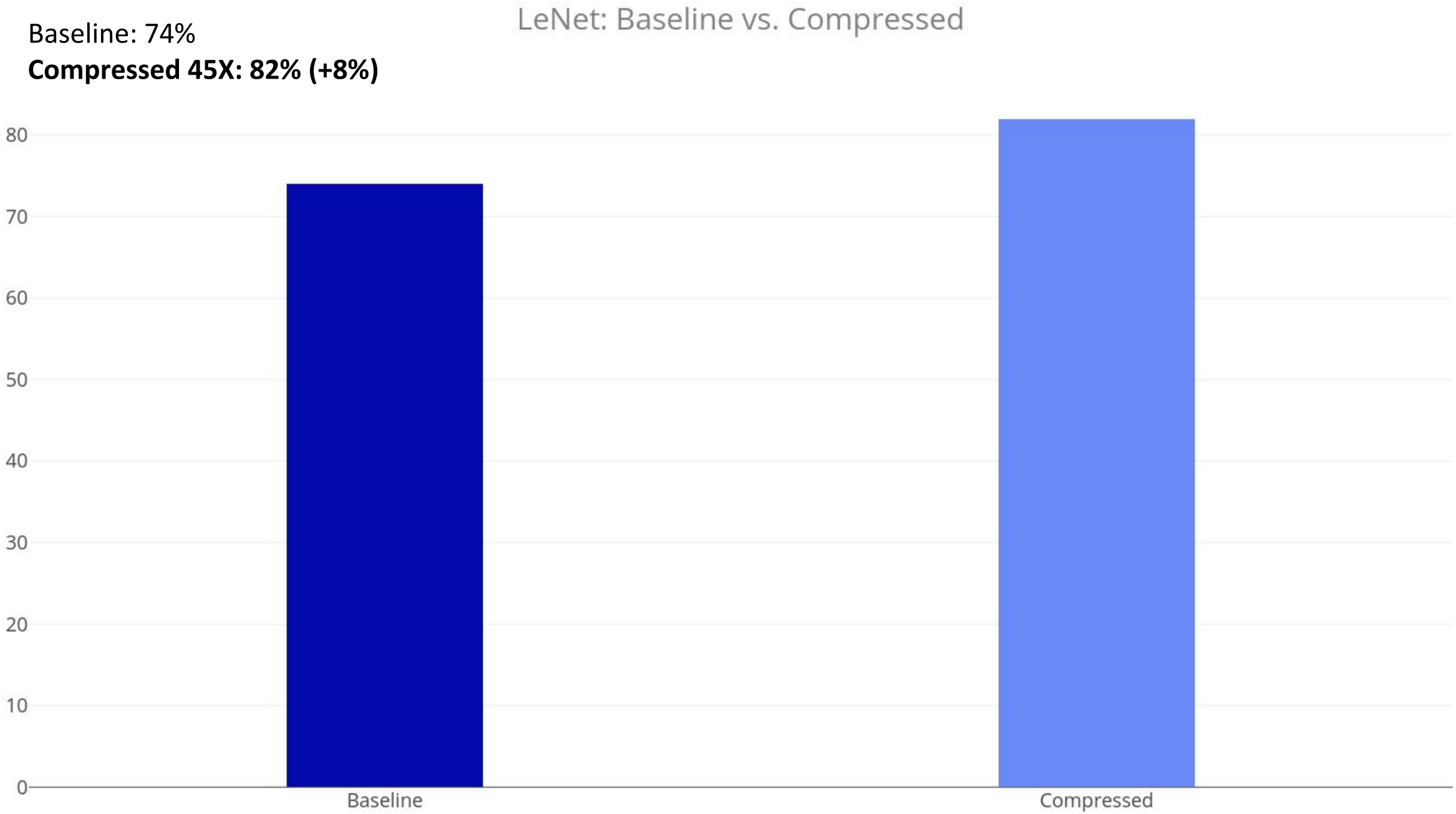

LeNet: Baseline vs. Compressed
1.2M
1M
0.8M
0.6M
0.4M
0.2M
0
Baseline
Compressed
45X meno parametri:
1.25 Milioni vs. ~30K

Confronto con l’originale

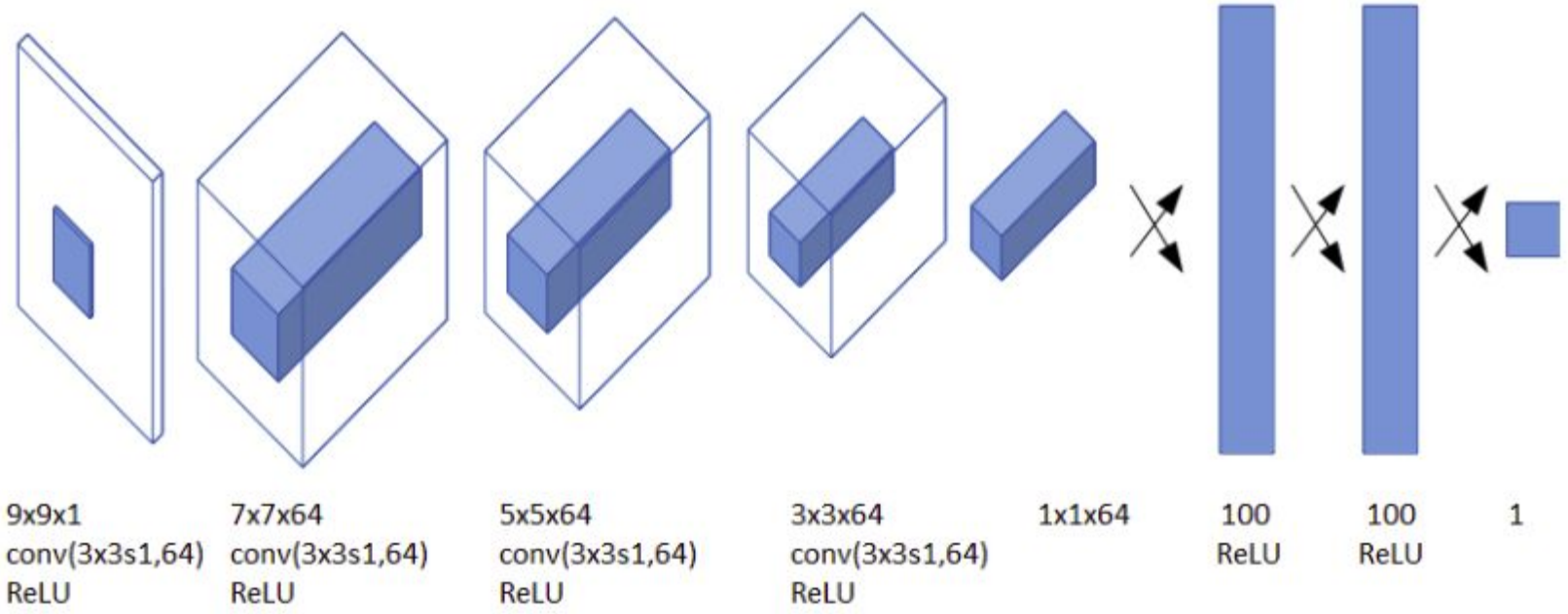


Baseline: 74%

Compressed 45X: 82% (+8%)



CCNN - Confidence CNN



Banco di prova interessante:

- **Stato dell'arte**
- Rete snella, solo 128K parametri

M. Poggi, S. Mattoccia, "Learning from scratch a confidence measure", (BMVC 2016)

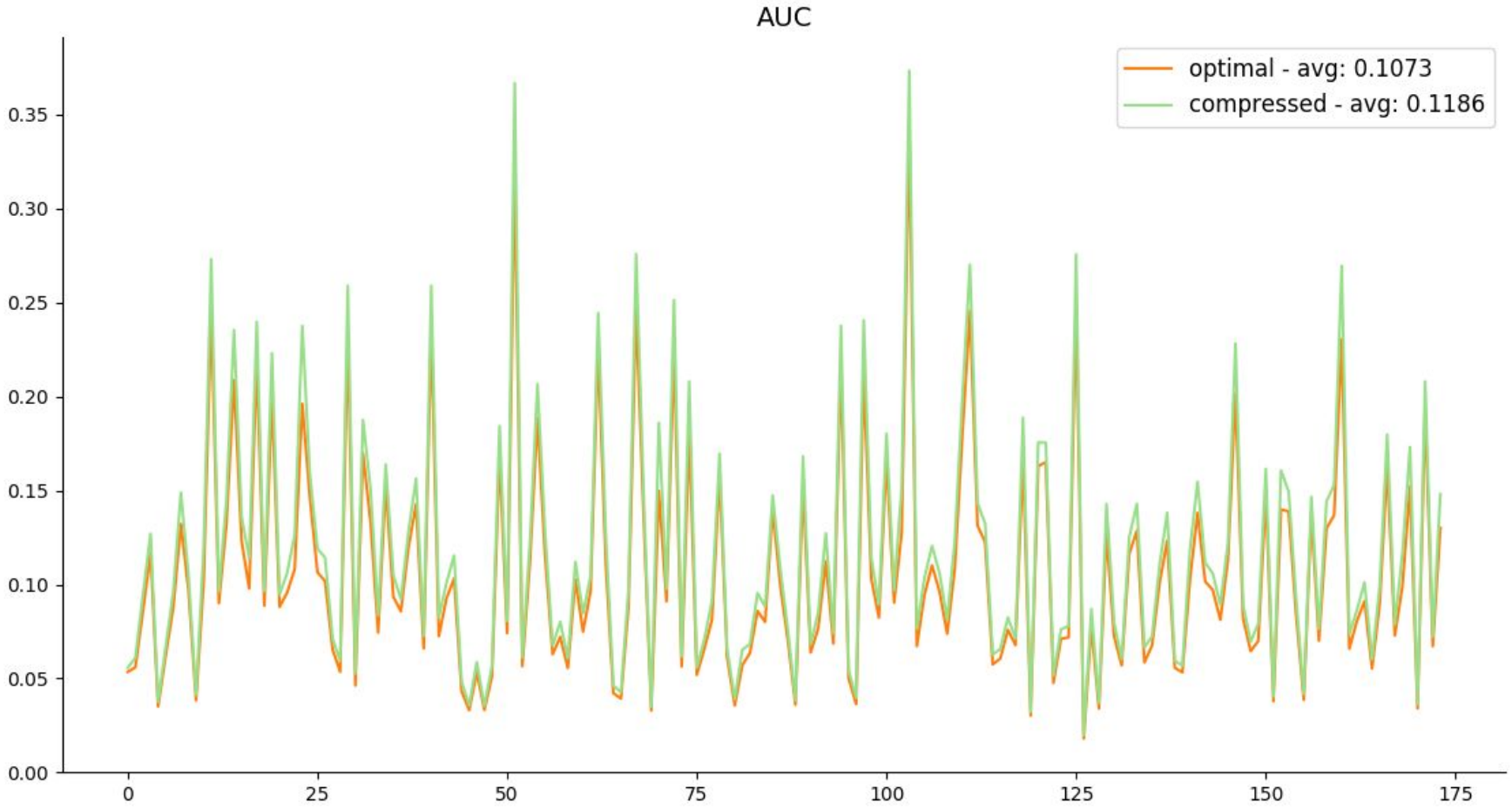
CCNN - Modelli Compressi

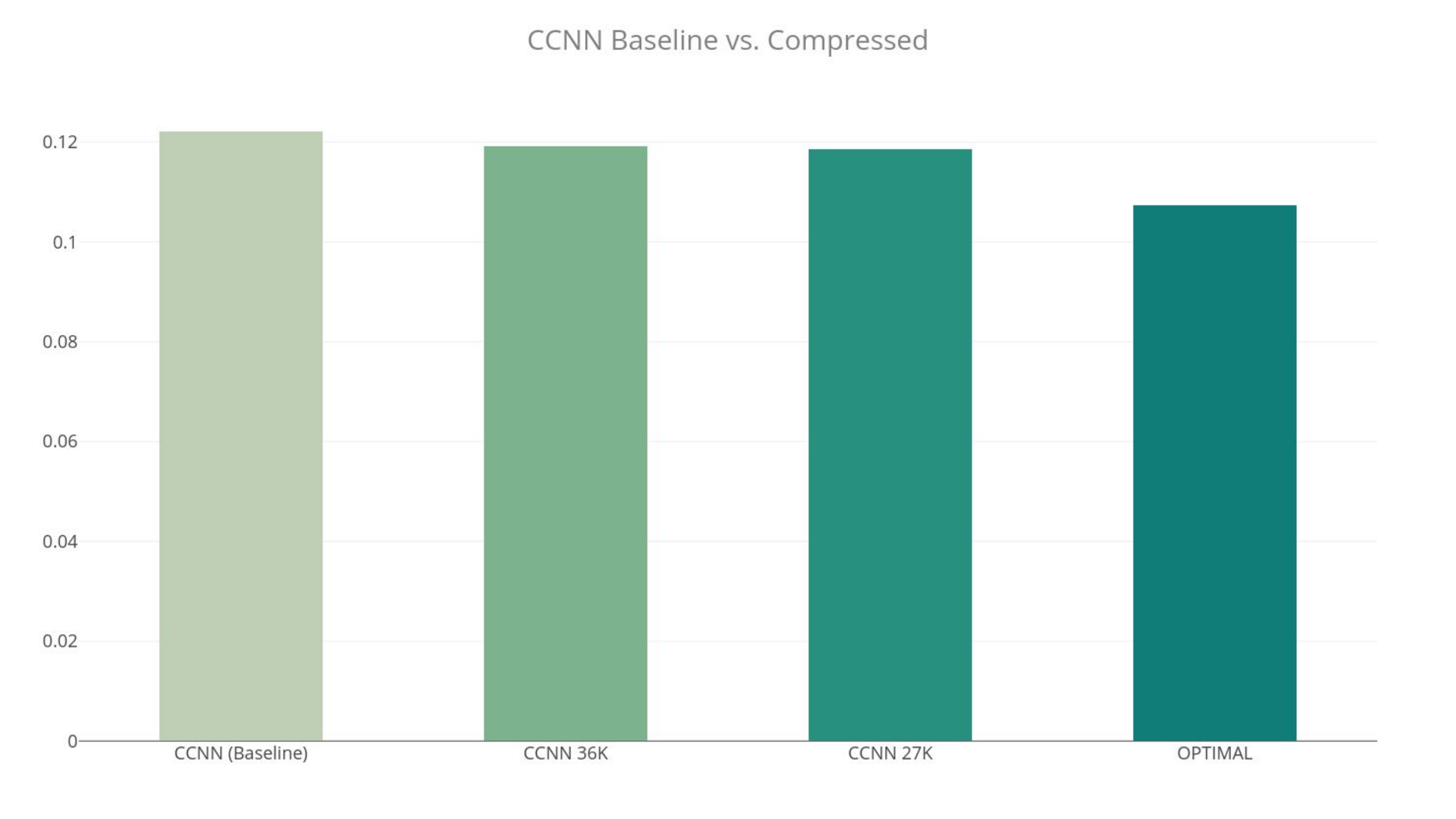
Sono stati proposti 2 modelli basati sul TD block a 4 layer:

- CCNN-36K: decomposti solo i layer di convoluzione
- CCNN-27K: decomposti **anche** i layer FC 1x1

Entrambi i modelli migliorano lo stato dell'arte.

Risultati – Compressed CCNN




CCNN Baseline vs. Compressed
0.12
0.1
0.08
0.06
0.04
0.02
0
CCNN (Baseline)
CCNN 36K
CCNN 27K
OPTIMAL

RIEPILOGO

TD Block: model design

- LeNet **45X** più piccola migliora l'accuratezza della baseline di 8%
- Migliorato **lo stato dell'arte** di CCNN con ⅕ dei parametri, dataset diverso da quelli classici
- Testate diverse configurazioni del TD Block

TD Block: model compression

- Tucker e CPD sono efficaci
- Risultati consistenti su diversi modelli
- Compressione fino a ~50X senza perdita di accuratezza
- Pipeline di decomposizione END-to-END

Conclusioni e sviluppi futuri

- Decomposizione tensoriale risulta efficace
- Richiede diverse iterazioni per recuperare l'accuracy
- Tucker più conservativo e stabile
- CPD più aggressivo sulla compressione ma richiede più attenzione

- Applicazione su altre CNN per applicazioni in tempo reale come MonoDepth
- Metodi di compressione misti Tucker/CPD
- Nuovi design per il TD Block (e.g. residual learning)
- La decomposizione non tiene conto della ottimizzazione globale della rete
  - ⇒ servono metodi "*learning to learn*" per ottimizzare la decomposizione

Grazie per l’attenzione!

Approfondimenti: rango R di un tensore

- Problema NP-difficile
- Metodi iterativi per stimare il trade-off tra efficienza e precisione dell'approssimazione; Molto onerosi (8 ore per la stima del rango di un layer di convoluzione)
- Metodi probabilistici: VBMF ⇒ risolvono il problema riportandolo nel dominio delle matrici e poi risolvendolo con una soluzione globale analitica. Facilmente inseribili nella pipeline
- Altri: metodi basati sull'apprendimento: VAEs, Reinforcement Learning, …