

Università degli Studi di Padova
Dipartimento di Matematica "Tullio Levi-Civita"
Corso di Laurea in Informatica

Predizione della Profondità con Deep Learning da Immagini di Telecamera Monoculare

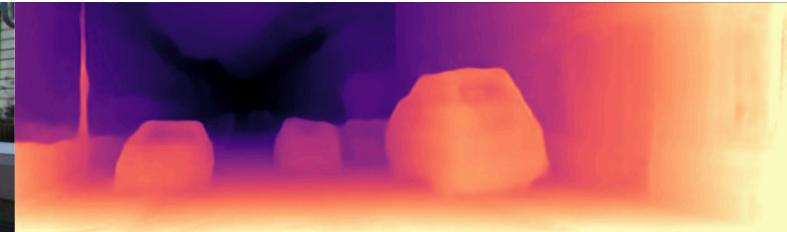
Esame di laurea

Indice

- *Monocular Depth Estimation*
- PyDNet e migrazione
- XiNet e l'attention
- Ricerca e sviluppo

Monocular Depth Estimation (MDE):

Il compito di stimare la distanza relativa, dalla telecamera, di ciascun pixel data una singola immagine, mediante *machine learning*.



Perché non usare dei sensori?

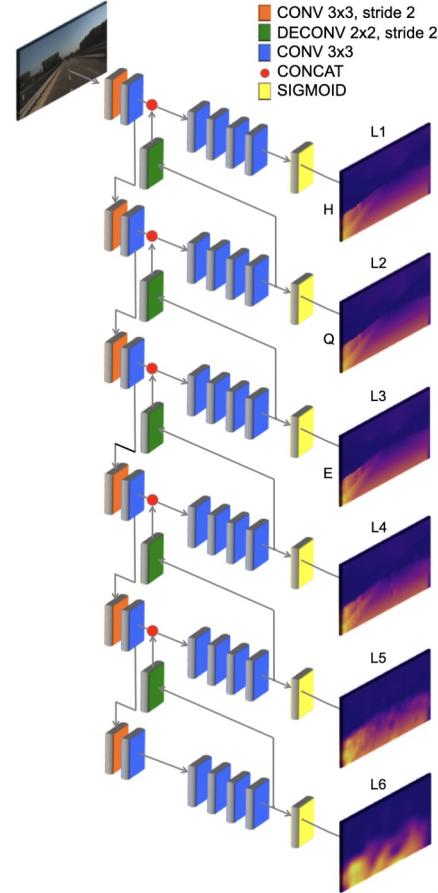
Monocular Depth Estimation

	Efficacia	Economicità	Efficienza
Sensori	✓	✗	✓
Sistemi MDE	✓	✓	✗
Sistemi MDE embeddable	✓	✓	✓

Embeddable: adatto per i sistemi *embedded*

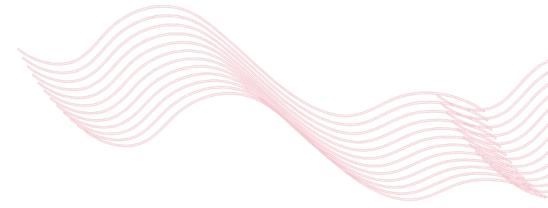
- Poca memoria
- Poca energia
- Poca potenza computazionale

Il modello



[paper](#)

PyDNet e migrazione



✓ Efficiente

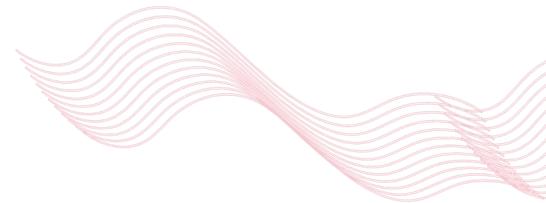
- Modello molto piccolo
- Sfrutta solo operazioni lineari

✓ Buona efficacia

- Buoni risultati qualitativi
- Risultati quantitativi vicini allo stato dell'arte

✗ Implementato in TensorFlow 1.8

- Versione deprecata del framework
- Incompatibile con versioni di CUDA attuali



Dataset

- Unità che approvvigiona i dati
- **Reso più veloce**
- Righe di codice: 217



Modelli

- Modelli effettivi da allenare
- **Reso più leggibile**
- *Righe di codice: 362*



Configurazione

- Unità che configura modelli e procedure
- **Migliorata l'esperienza di sviluppo**
- Righe di codice: 303



Allenamento

- Procedura di allenamento
- **Migliorata la leggibilità**
- **Migliorato il salvataggio** dei modelli
- Righe di codice: 518

La migrazione (3/3)

PyDNet e migrazione



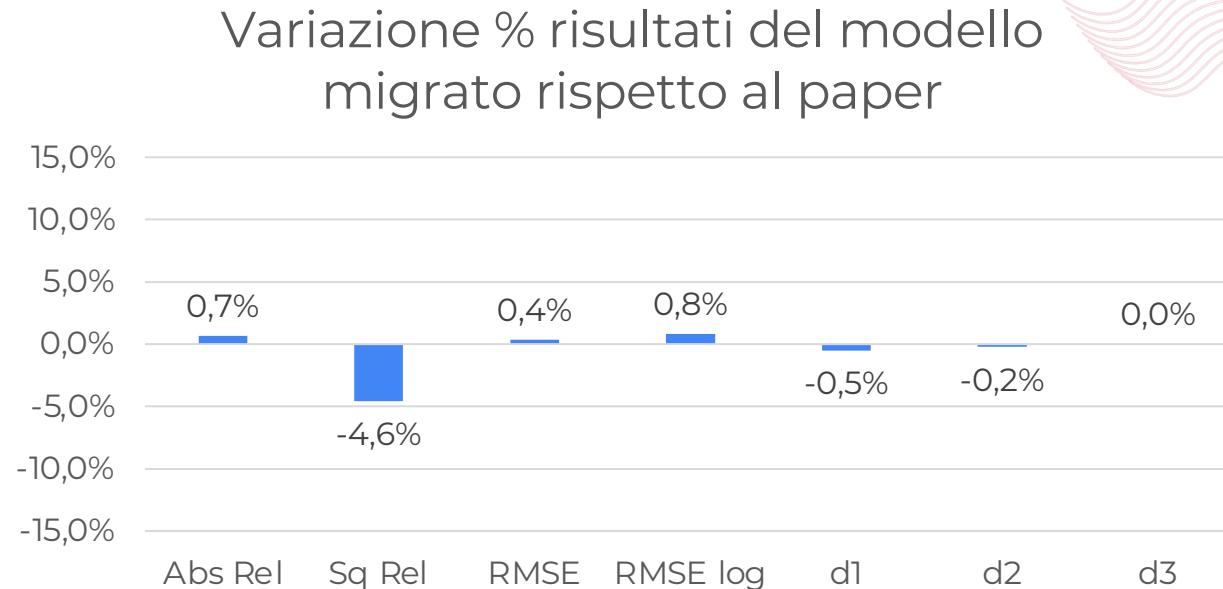
Utilizzo

- Procedura di utilizzo del modello
- Migrato l'utilizzo mediante terminale
- Migrato l'utilizzo mediante *live webcam*
- **Implementato l'utilizzo mediante modulo importabile**
- Righe di codice: 370



Valutazione

- Procedura di *testing* e validazione
- Riscrittura della procedura di *testing retrocompatibile* con la procedura di validazione originale
- Righe di codice: 134

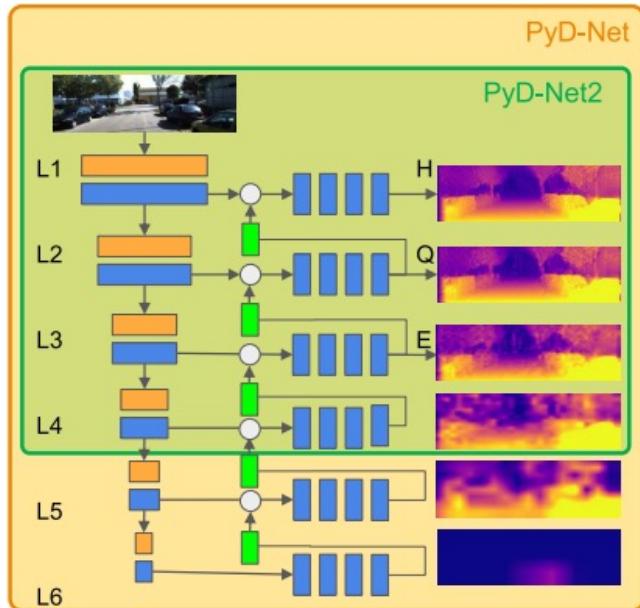


Esperimento:

- Allenamento *dataset CityScapes*
- *Fine-tuning dataset KITTI*

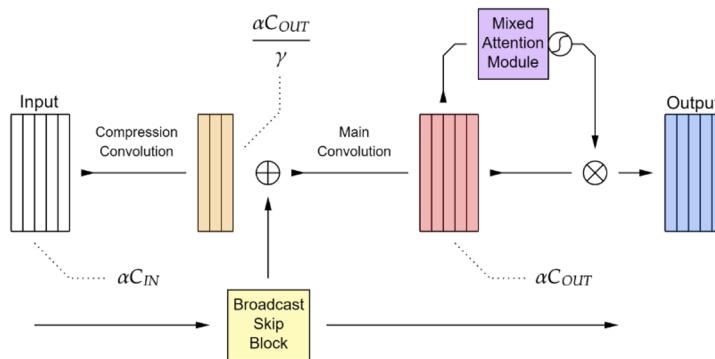
PyDNet V2

PyDNet e migrazione



[paper](#)

	Minore è meglio						Maggiore è meglio		
	#p	Inf Time	Abs Rel	Sq Rel	RMSE	RMSE log	d1	d2	d3
PyDNet V1	~1.9M	0.15	0.16	1.52	6.229	0.253	0.782	0.916	0.964
PyDNet V2	~700k	0.1	0.157	1.487	6.167	0.254	0.783	0.917	0.964
Miglioramento %	64%	33%	1.9%	2.2%	1%	-0.4%	0.1%	0.1%	0%



XiConv

- Riduce l'impatto energetico rispetto alle convoluzioni
- Ottimi *benchmark* di valutazione
- Progettato per l'uso in sistemi *embedded*
- [paper](#)



Self attention

- Complessità quadratica nella dimensione dell'input
- Pesa ogni “pixel” usando le correlazioni con tutti gli altri
- Molto costoso ma molto efficace
- [paper](#)

Convolutional Block Attention Module

- Complessità lineare nella dimensione dell'input
- Sfrutta convoluzioni e pooling
- Meno costoso ma discretamente efficace
- [paper](#)

Modelli sperimentali

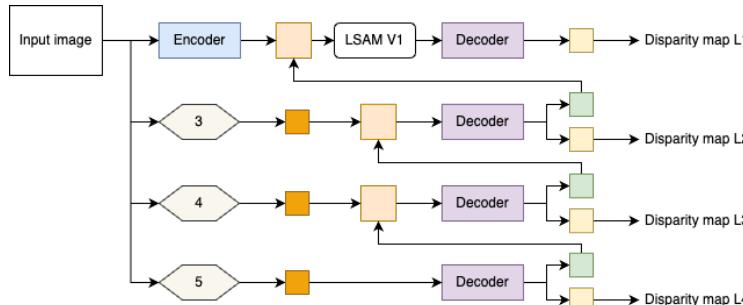
Ricerca e sviluppo

	# pyramid levels	# XiNets layers	Parallelized graph	Self attention	CBAM attention
PyXiNet α (2 modelli)	3-4	3	✗	✗	✗
PyXiNet β (4 modelli)	3-4	3-5	✓	✗	✗
PyXiNet M (4 modelli)	4	3-5	✓	✓	✗
PyXiNet βCBAM (2 modelli)	4	3-5	✓	✗	✓
PyDNet CBAM (1 modello)	4	0	✗	✗	✗

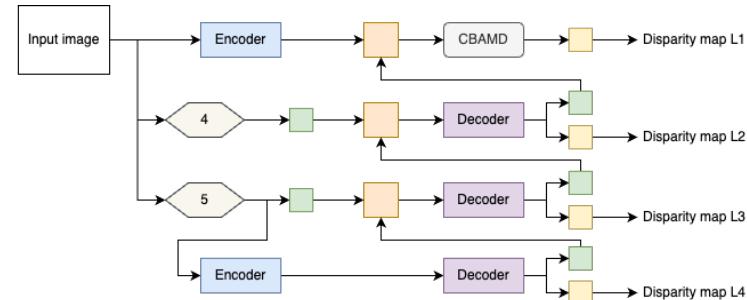
Modelli sperimentali: i migliori

Ricerca e sviluppo

PyXiNet M II



PyXiNet β CBAM I

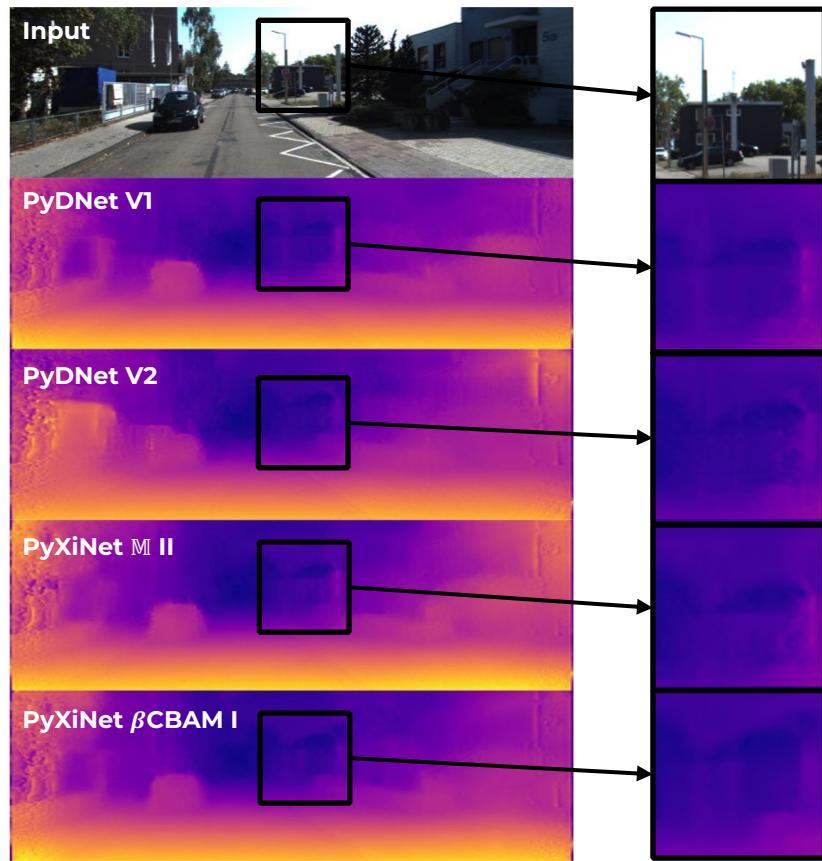
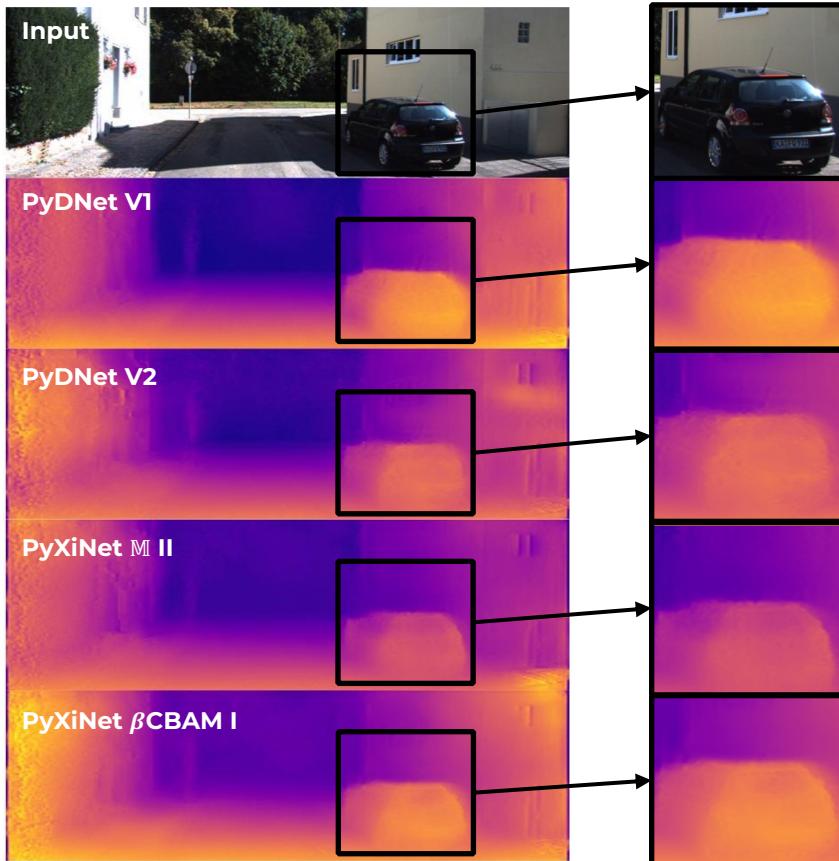


Minore è meglio

	#p	Inf Time	Abs Rel	Sq Rel	RMSE	RMSE log	d1	d2	d3
PyDNet V1	<u>~1.9M</u>	<u>0.15</u>	<u>0.16</u>	<u>1.52</u>	<u>6.229</u>	<u>0.253</u>	<u>0.782</u>	<u>0.916</u>	<u>0.964</u>
PyXiNet M II	~2.2M	0.38	0.14	1.289	5.771	0.234	0.814	0.933	0.969
Miglioramento % M II	-13%	-153%	13%	15%	7%	8%	4%	2%	1%
PyXiNet β CBAM I	~1.2M	0.19	0.143	1.296	5.91	0.239	0.805	0.928	0.968
Miglioramento % β CBAM I	37%	-27%	11%	15%	5%	6%	3%	1%	0%

Confronti qualitativi

Ricerca e sviluppo



Report quantitativo

320 ore di lavoro

- 80 ore di studio (*framework e paper*)
- 80 ore migrazione PyDNet
- 40 ore di sperimentazione con PyDNet
- 120 ore di *R&D*

3906 righe di codice

- 1904 (48%) in migrazione
- 2002 (52%) in *R&D*

15 modelli implementati

- PyDNet V1 & V2
- 13 modelli sperimentali

Documentazione

- Manuale utente per i modelli migrati e sperimentali