

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

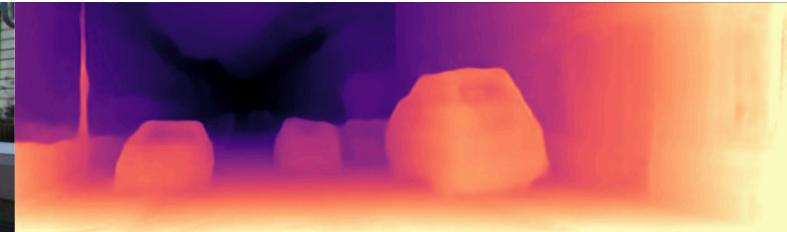
*Esame di laurea*

# 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?

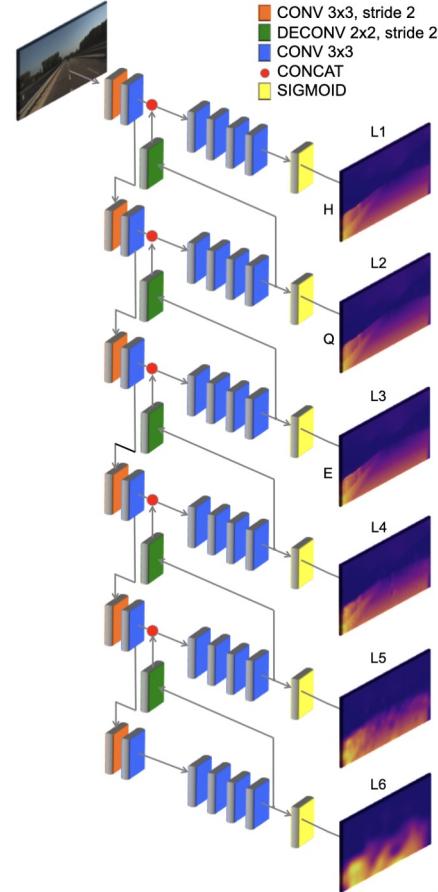
## Monocular Depth Estimation

	Efficacia	Economicità	Efficienza
<b>Sensori</b>	✓	✗	✓
<b>Sistemi MDE</b>	✓	✓	✗
<b>Sistemi MDE embeddable</b>	✓	✓	✓

*Embeddable*: adatto per i sistemi *embedded*

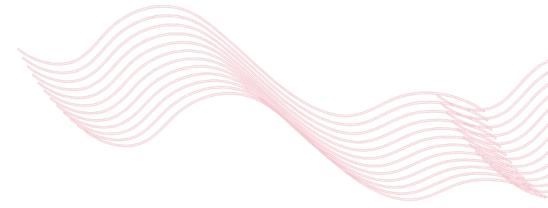
- Poca memoria
- Poca energia
- Poca potenza computazionale

# Il modello



[paper](#)

## PyDNet e migrazione



### ✓ **Efficiente**

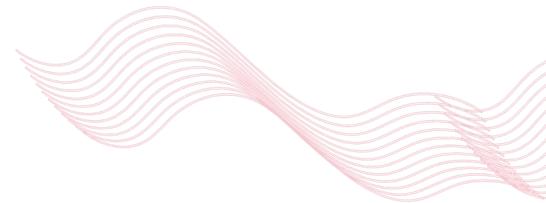
- Modello molto piccolo
- Sfrutta solo operazioni lineari

### ✓ **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



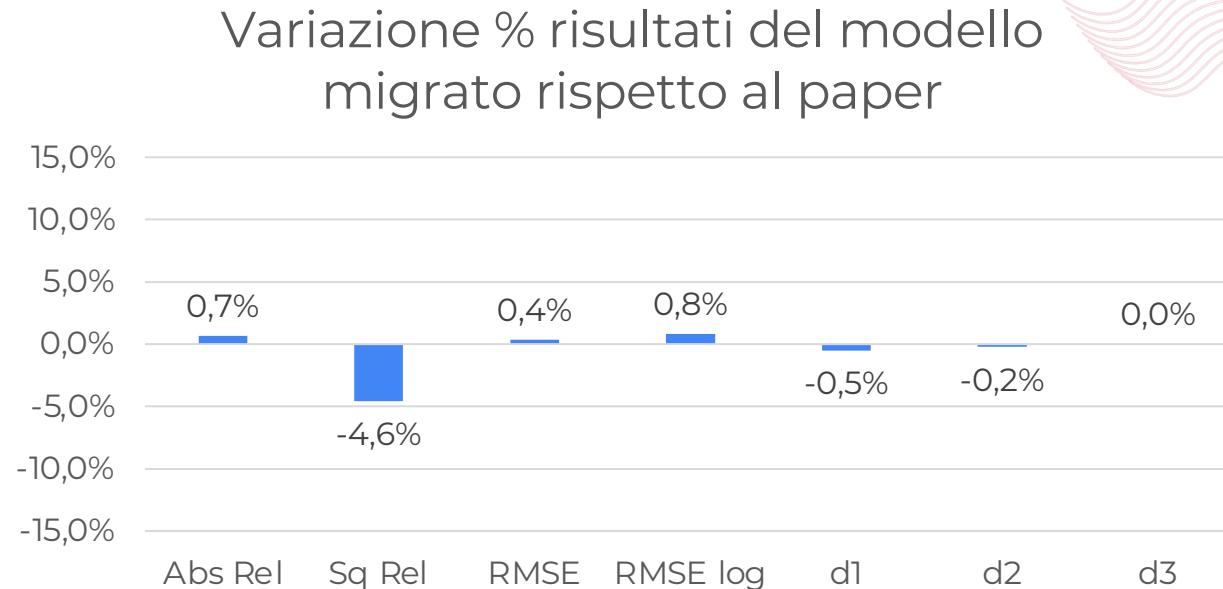
## Utilizzo

- 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

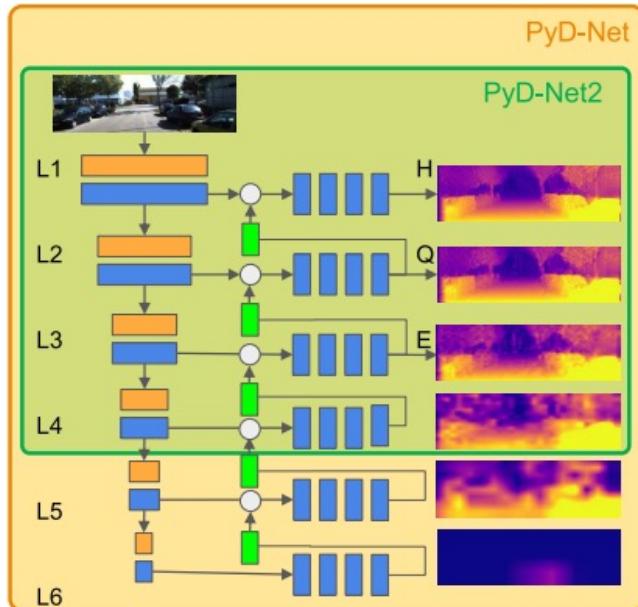


Esperimento:

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

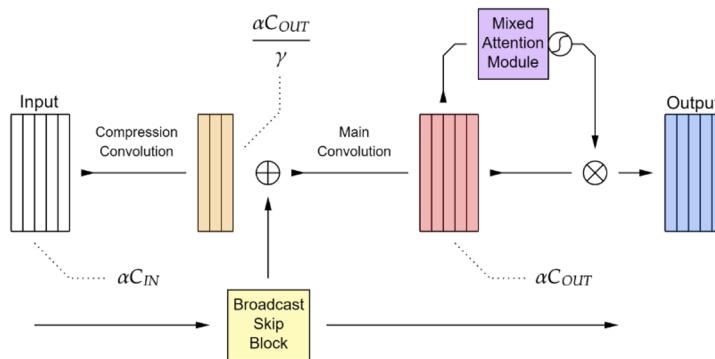
# PyDNet V2

## 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	<b>0.253</b>	0.782	0.916	<b>0.964</b>
PyDNet V2	~700k	0.1	<b>0.157</b>	<b>1.487</b>	<b>6.167</b>	0.254	<b>0.783</b>	<b>0.917</b>	<b>0.964</b>
Miglioramento %	<b>64%</b>	<b>33%</b>	<b>1.9%</b>	<b>2.2%</b>	<b>1%</b>	<b>-0.4%</b>	<b>0.1%</b>	<b>0.1%</b>	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

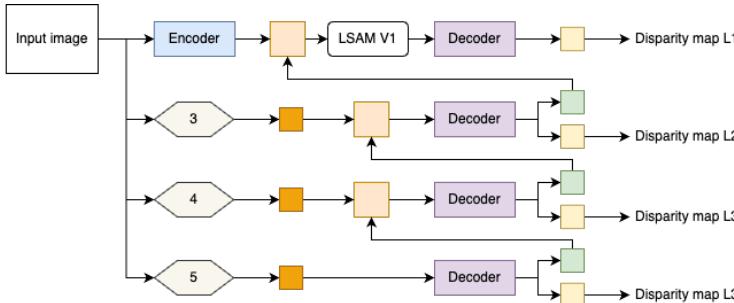
Ricerca e sviluppo

	# pyramid levels	# XiNets layers	Parallelized graph	Self attention	CBAM attention
<b>PyXiNet <math>\alpha</math></b> (2 modelli)	3-4	3	✗	✗	✗
<b>PyXiNet <math>\beta</math></b> (4 modelli)	3-4	3-5	✓	✗	✗
<b>PyXiNet <math>\text{M}</math></b> (4 modelli)	4	3-5	✓	✓	✗
<b>PyXiNet <math>\beta</math>CBAM</b> (2 modelli)	4	3-5	✓	✗	✓
<b>PyDNet CBAM</b> (1 modello)	4	0	✗	✗	✗

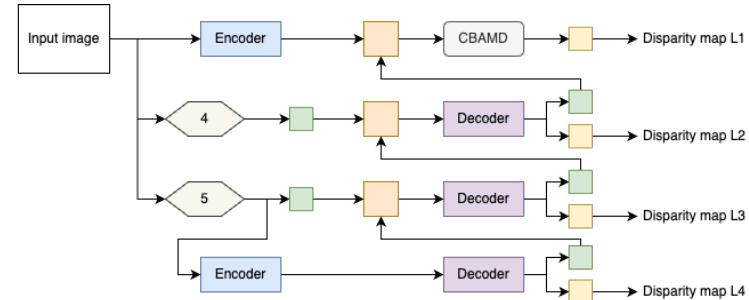
# Modelli sperimentali: i migliori

Ricerca e sviluppo

PyXiNet M II



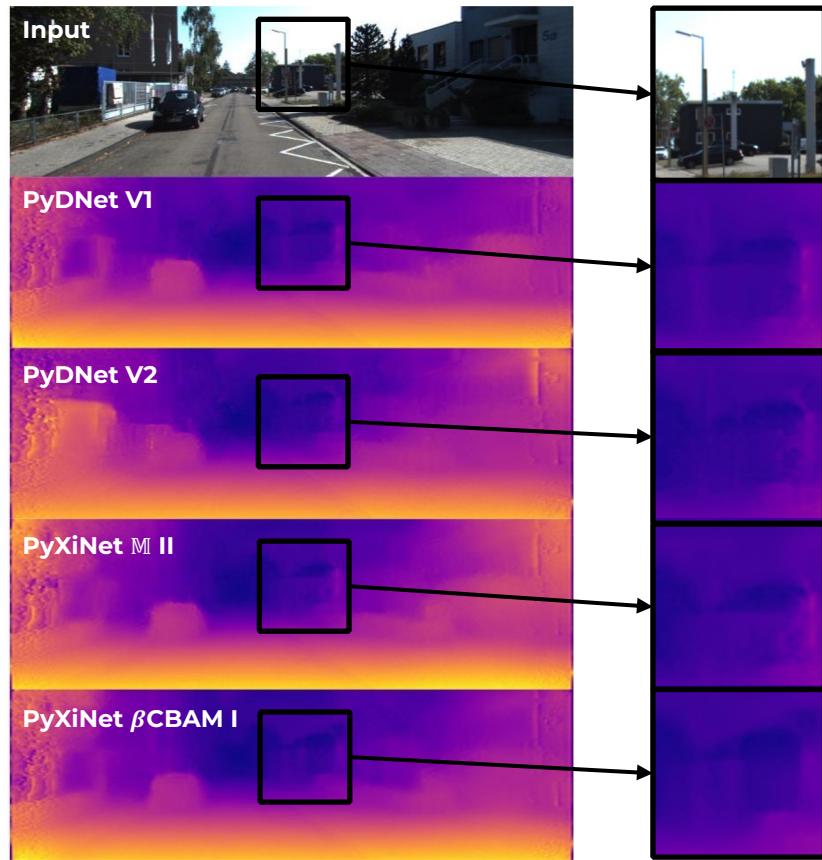
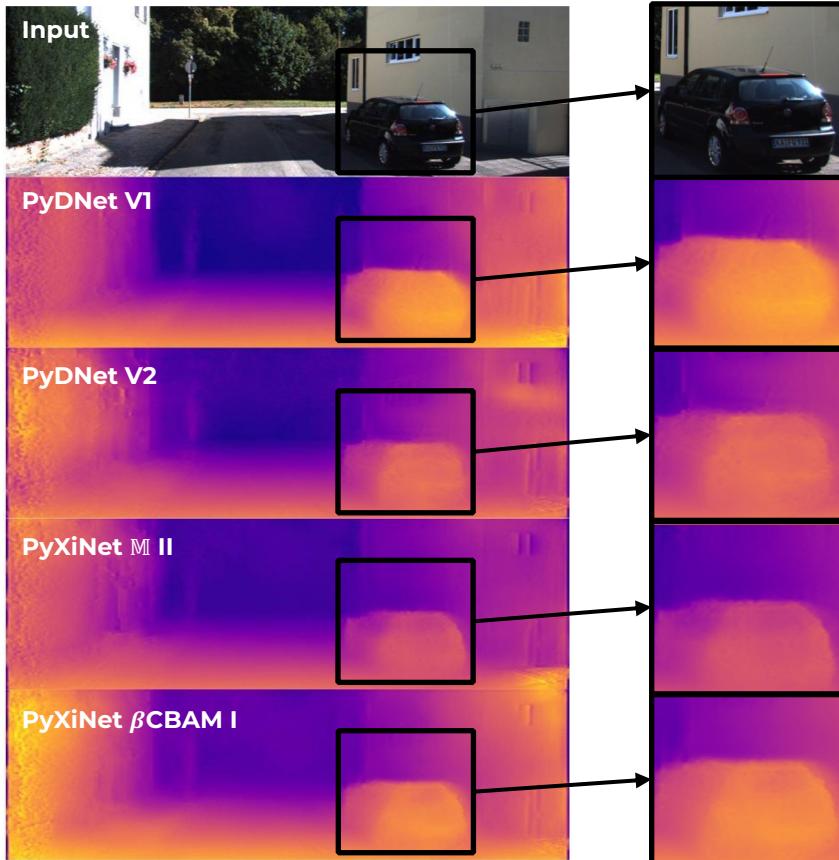
PyXiNet  $\beta$ CBAM I



	Minore è meglio						Maggiore 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	<b>0.14</b>	<b>1.289</b>	<b>5.771</b>	<b>0.234</b>	<b>0.814</b>	<b>0.933</b>	<b>0.969</b>
Miglioramento % M II	<b>-13%</b>	<b>-153%</b>	<b>13%</b>	<b>15%</b>	<b>7%</b>	<b>8%</b>	<b>4%</b>	<b>2%</b>	<b>1%</b>
PyXiNet $\beta$ CBAM I	<b>~1.2M</b>	0.19	0.143	1.296	5.91	0.239	0.805	0.928	0.968
Miglioramento % $\beta$ CBAM I	<b>37%</b>	<b>-27%</b>	<b>11%</b>	<b>15%</b>	<b>5%</b>	<b>6%</b>	<b>3%</b>	<b>1%</b>	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