Progetto Deep Learning

Obiettivo: semantic segmentation su auto per il riconoscimento di diverse classi

Lucia Ferrari 152999

Elena Zerbin 145302

Step:

01. Visualizzazione dataset

02. Data augmentation

03. Scelta della rete

04. Scelta della metrica

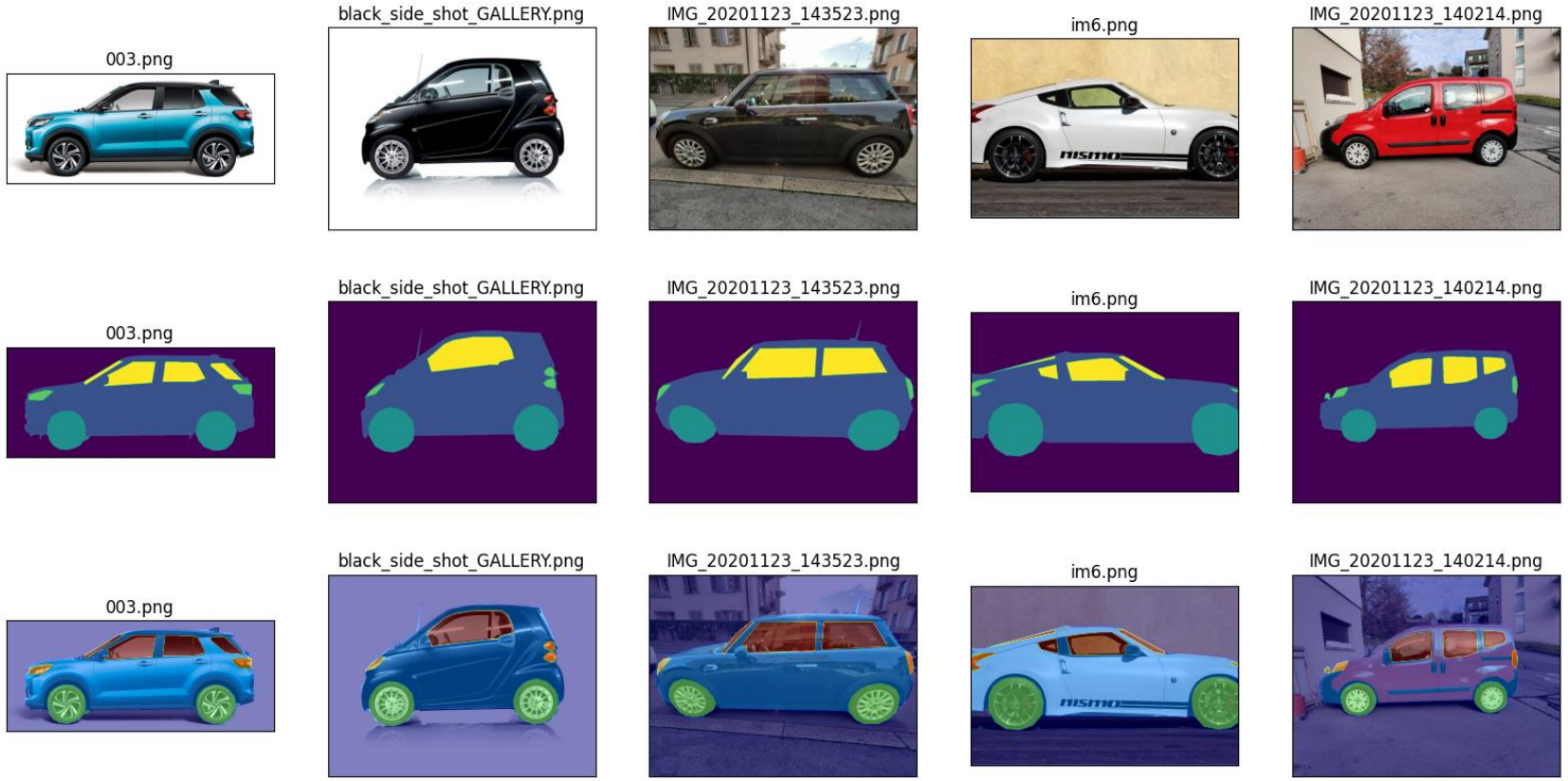
05. Segmentation Library

06. Altri tentativi di miglioramento

1) Visualizzazione del dataset

- Studio della composizione del dataset: 211 immagini di auto di diverse dimensioni e corrispondenti maschere
- Identificazione della 5 classi: carrozzeria, ruote, vetri, luci e background

5 immagini random delle macchine del dataset
Classi:car,wheel,light,windows,background

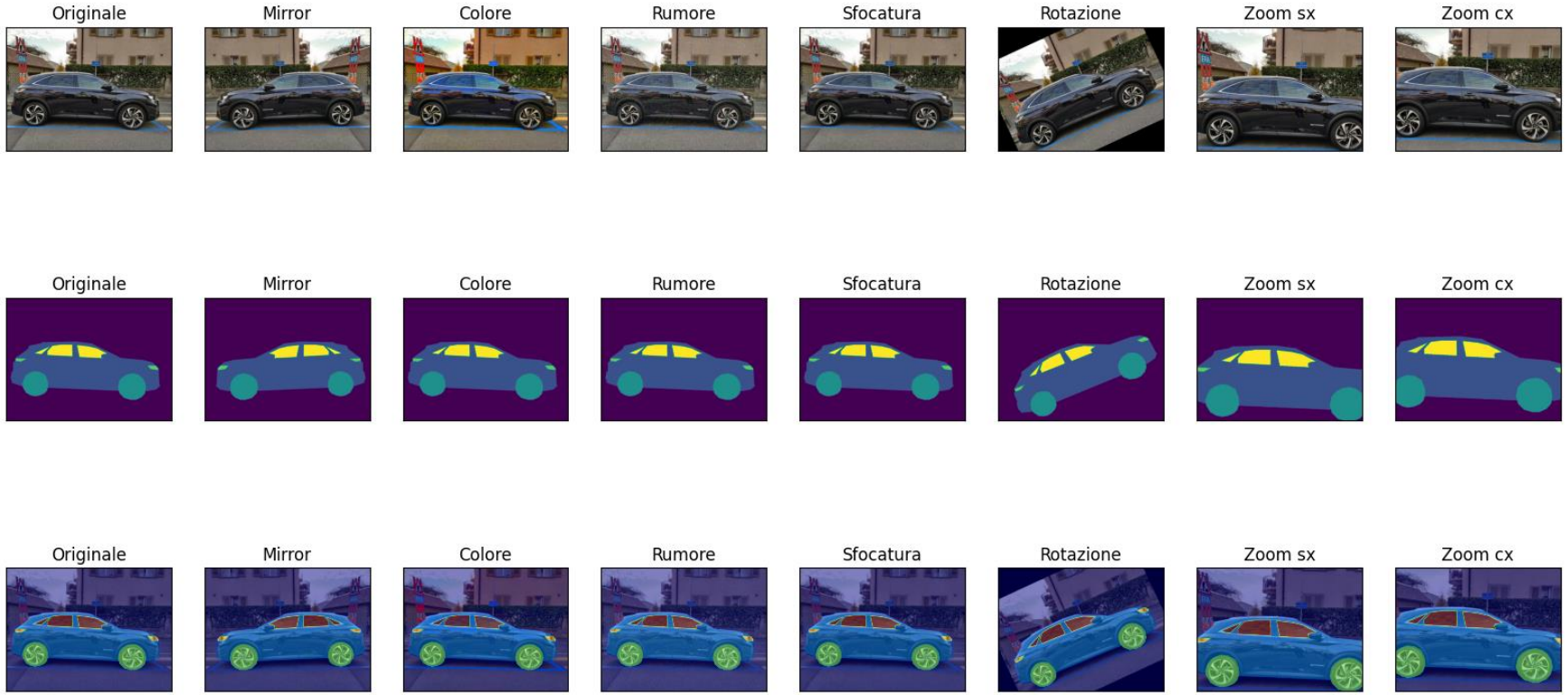


2) Data Augmentation

Abbiamo deciso di eseguire data augmentation portando il dataset da 211 immagini con la corrispondente maschera a 1688.
Tipo di trasformazioni effettuate:

Specchiatura: orizzontale
Colore: randomico
Aggiunta di rumore gaussiano

Sfocatura o definizione: fattore random
Rotazione: random + 25° o -25°
Zoom: a sinistra e centrale di un fattore del 30%



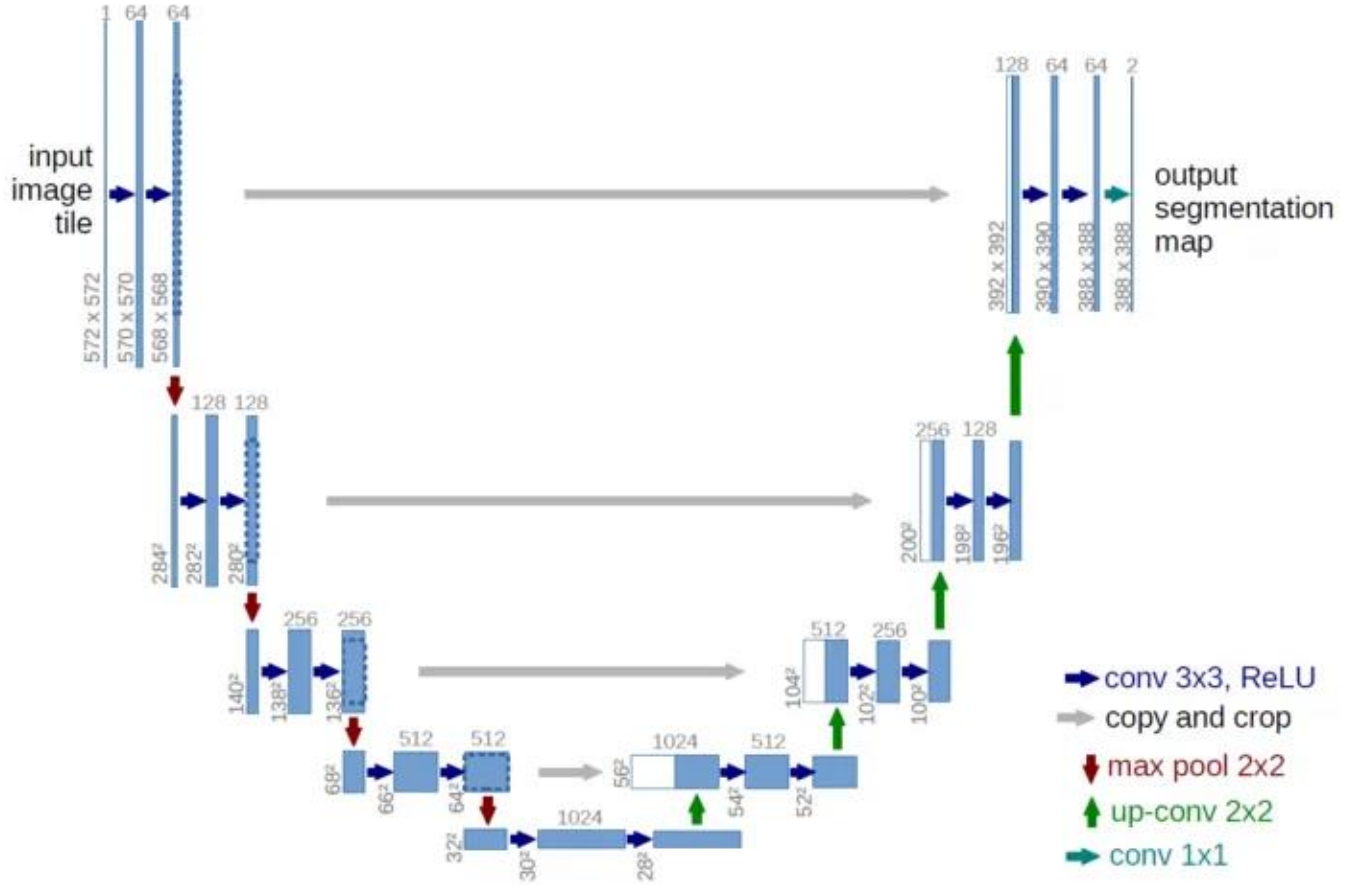
Suddivisione del dataset:
80% Training (di cui un ulteriore 20% per validation set)
20% Test Set

Dimensioni:
Training set: 1080
Validation set: 270
Test set: 338
Batch size: 16

Le proporzioni delle immagini vengono mantenute tramite padding (tranne nella prima rete addestrata)

https://arxiv.org/pdf/1505.04597.pdf

3) Scelta della rete: UNet



La rete UNet è formata da:

- **Encoder**: estrazione delle feature tramite convoluzioni 3x3 e max pooling con ReLU e stride a 2 per effettuare downsampling

- **Decoder**: upsampling e convoluzioni 3x3 con ReLU fino ad ottenere in output la maschera con il necessario numero di classi

Cosa abbiamo utilizzato noi:

- Dimensione di input: 128 x 128 x 3
- Dimensione di output: 128 x 128 x 5
- Aggiunta di Batch Normalization tra convoluzione e attivazione
- Early stopping con pazienza = 5

Risultati rete UNet

Abbiamo eseguito la rete UNet prima su immagini di dimensione 128x128x3 e poi su immagini 256x256x3 utilizzando come metrica di precisione **accuracy,** come funzione di loss **categorical cross entropy** e come ottimizzatore: **adam**

UNet 128x128

Epoche di training: 14 (media di 650s)
Accuracy sul test: 0.9544
Loss sul test: 0.1340
Mean IoU: 0.7540

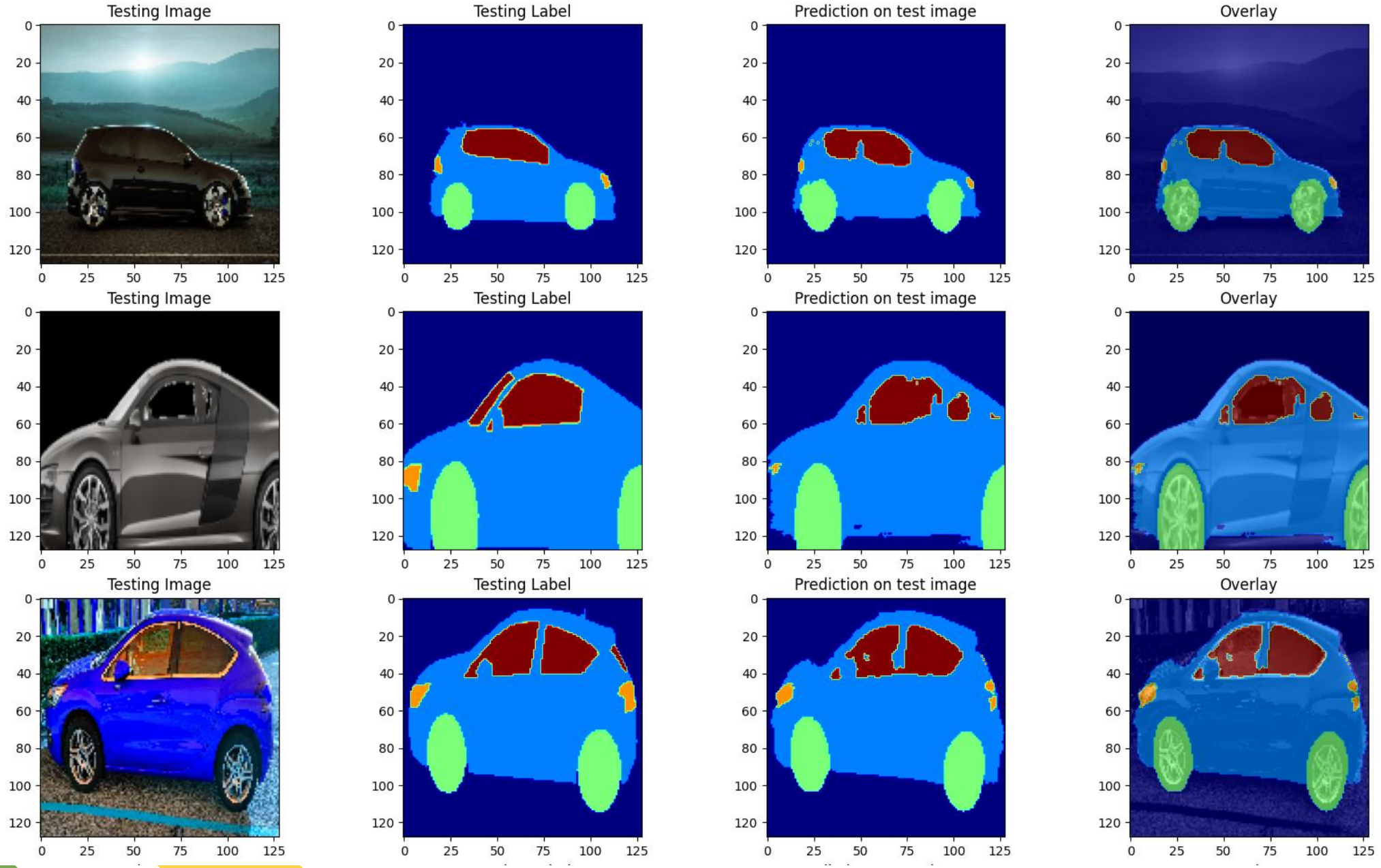
Classe	IoU
Background	0.9632
Car	0.8651
Wheels	0.8850
Lights	0.3289
Windows	0.7281

UNet 256x256

Epoche di training: 35 (media di 3000s)
Accuracy sul test: 0.9716
Loss sul test: 0.0790
Mean IoU: 0.7387

Classe	IoU
Background	0.9811
Car	0.8629
Wheels	0.8650
Lights	0.2610
Windows	0.7234

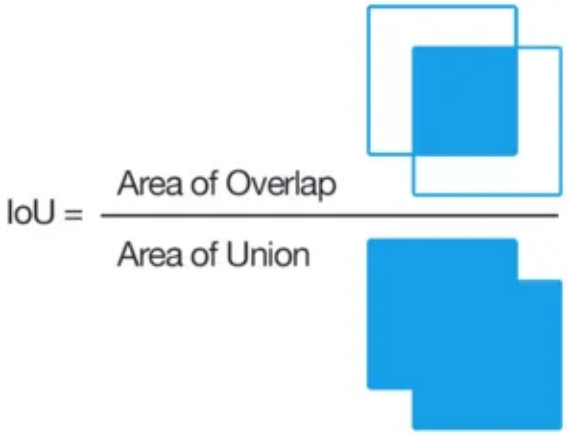
Risultati di bassa qualità (UNet 128x128):



4) Scelta di metriche migliori per il training

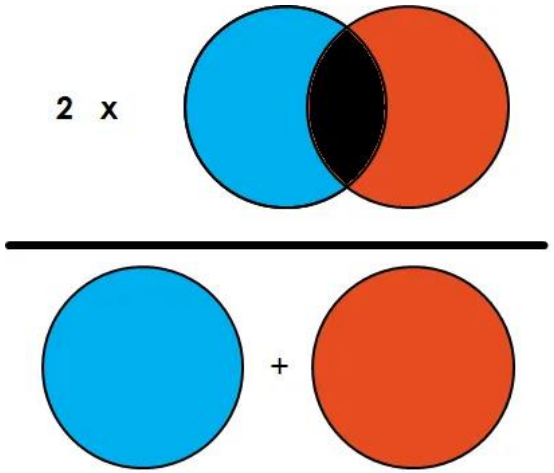
L'accuracy si è rivelata una scelta sbagliata come metrica a causa dello sbilanciamento delle classi nelle immagini (in particolare le luci delle auto che occupano la percentuale minore. Nuove metriche utilizzate:

Intersection Over Union (IOU)



Area sovrapposizione/ area unione

Dice Coefficent (F1-Score)



2 * area sovrapposizione / totale pixel

IoU come metrica di Loss

$$L(A,B) = 1 - \frac{A \cap B}{A \cup B}$$

Libreria utilizzata: Semantic Segmentation

https://github.com/qubvel/segmentation_models

Le caratteristiche principali di questa libreria sono:

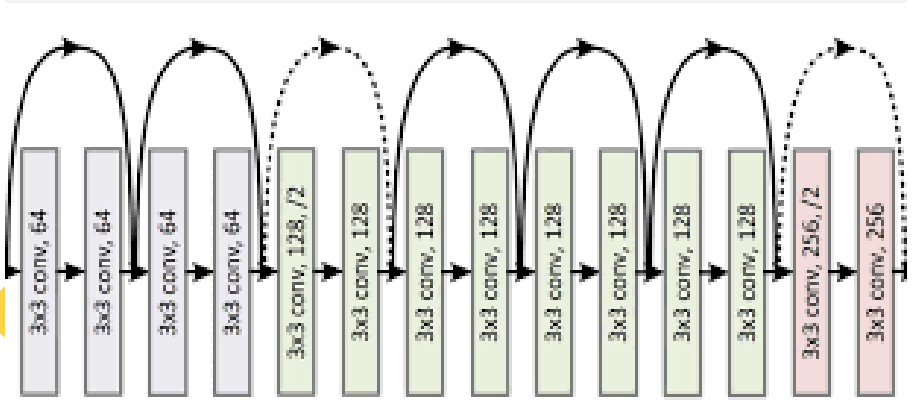
- API di alto livello (solo due righe di codice per creare un modello per la segmentazione)
- 4 architetture di modelli per la segmentazione di immagini binarie e multiclasse (incluso il leggendario Unet)
- 25 backbone disponibili per ogni architettura
- Tutte i backbone hanno pesi pre-addestrati per una convergenza più veloce e migliore
- Funzioni di loss utili (Jaccard, Dice, Focal) e metriche (IoU, F-score)

Noi abbiamo deciso di utilizzare come backbone: **Resnet34** e **VGG16**

Resnet34 vs VGG16

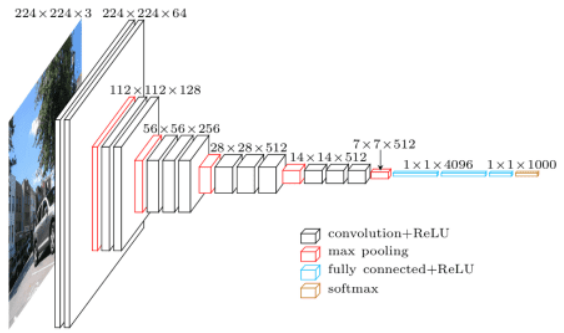
Resnet34

- Costituita da 34 strati di convoluzione
- Introduzione di blocchi residuali: formati da convoluzioni 3x3 e concatenazione delle informazioni iniziali e finali.
- Strategia down-sampling mediante convoluzioni di dimensione 3x3 e stride 2 per ridurre la dimensione del tensore di input.

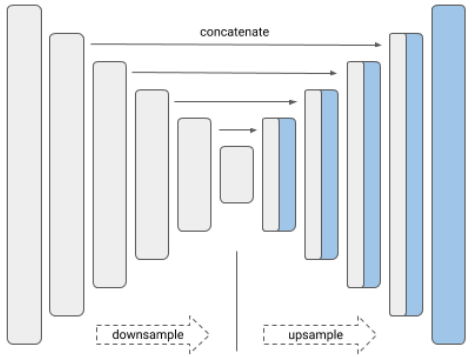


VGG16

- costituita da 16 strati di convoluzione: 13 strati convoluzionali, 5 strati di max pooling e 3 strati fully connected
- Ogni blocco ha una profondità crescente e utilizza convoluzioni di dimensione 3x3 con padding 1 e funzione di attivazione ReLU.
- strategia di down-sampling mediante uno strato di pooling massimo 2x2, che riduce la dimensione della rappresentazione della feature map.



5.1) Risultati UNet



Abbiamo eseguito la rete UNet utilizzando come backbone 'resnet34' e preaddestramer sul dataset ImageNet prima su immagini di dimensione 128x128x3 e poi su immagini 256x256x3 utilizzando come metrica di precisione **IoU Score e F1Score,** come funzione loss **IoU Score,** come ottimizzatore: **adam** e abbiamo aggiunto Batch Normalization

UNet 128x128

Epoche di training: 33 (media di 180s)
Accuracy sul test: 0.9829
Loss sul test: 0.1880
F1 Score: 0.9036
Mean IoU: 0.8358

Classe	IoU
Background	0.9887
Car	0.9155
Wheels	0.9113
Lights	0.5504
Windows	0.8314

UNet 256x256

Epoche di training: 16 (media di 650s)
Accuracy sul test: 0.9851
Loss sul test: 0.1697
F1 Score: 0.9129
Mean IoU: 0.8490

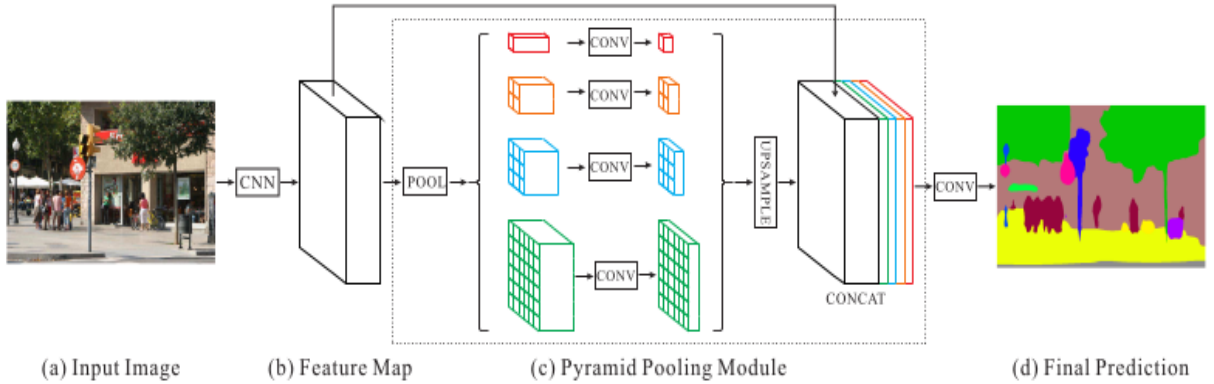
Classe	IoU
Background	0.9900
Car	0.9259
Wheels	0.9214
Lights	0.5775
Windows	0.8536

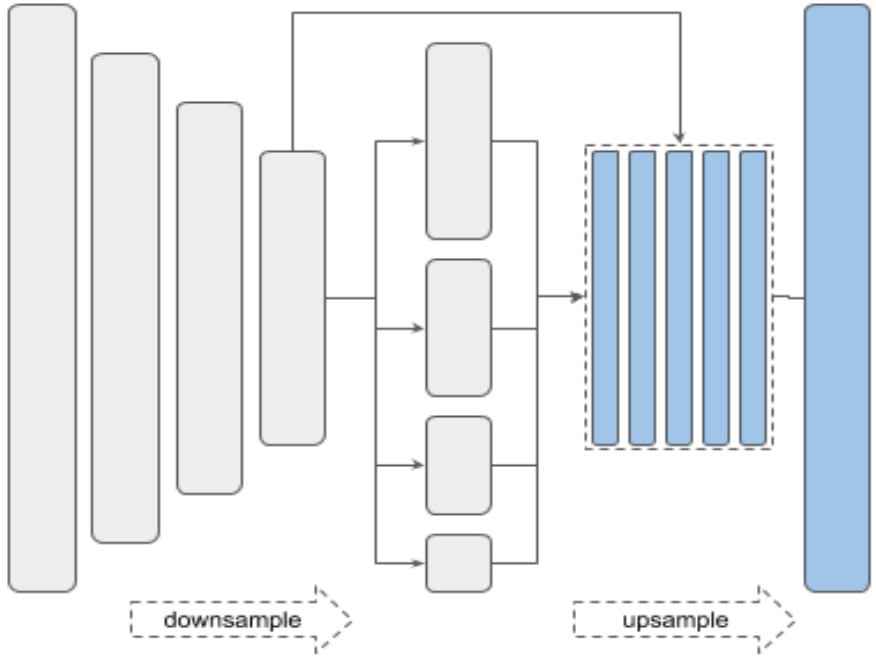
https://arxiv.org/pdf/1612.01105.pdf

5.2) PSPNet

PSPNet (Pyramid Scene Parsing Network) utilizza una rete neurale convoluzionale profonda (CNN) per effettuare classificazione, con alcune caratteristiche che la rendono particolarmente efficace:

- utilizza un Pyramid Pooling Module, che concatena:
  - Global pooling (conv 1x1)
  - Convoluzione 2x2
  - Convoluzione 3x3
  - Convoluzione 6x6
- La rete è sempre composta da:
  - il **modulo di estrazione** delle caratteristiche (encoder)
  - il **modulo di decodifica** per la segmentazione semantica (decoder)
- La classificazione finale utilizza **softmax** per assegnare una classe a ciascun pixel dell'immagine.





Risultati rete PSPNet

Abbiamo eseguito la rete PSPNet prima su immagini di dimensione 144x144x3 e poi su immagini 192x192x3 utilizzando come metrica di precisione **IoU Score e F1Score,** come funzione di loss **IoU Score,** come ottimizzatore: **adam** e abbiamo aggiunto Batch Normalization

PSPNet 144x144

Epoche di training: 38 (media di 160s)
Accuracy sul test: 0.9772
Loss sul test: 0.2397
F1 Score: 0.8784
Mean IoU: 0.7969

Classe	IoU
Background	0.9849
Car	0.8872
Wheels	0.8965
Lights	0.4719
Windows	0.7814

PSPNet 192x192

Epoche di training: 37 (media di 271s)
Accuracy sul test: 0.9810
Loss sul test: 0.2027
F1 Score: 0.9009
Mean IoU: 0.8284

Classe	IoU
Background	0.9867
Car	0.9068
Wheels	0.9089
Lights	0.5500
Windows	0.8219

https://arxiv.org/pdf/1707.03718.pdf

5.3) LinkNet

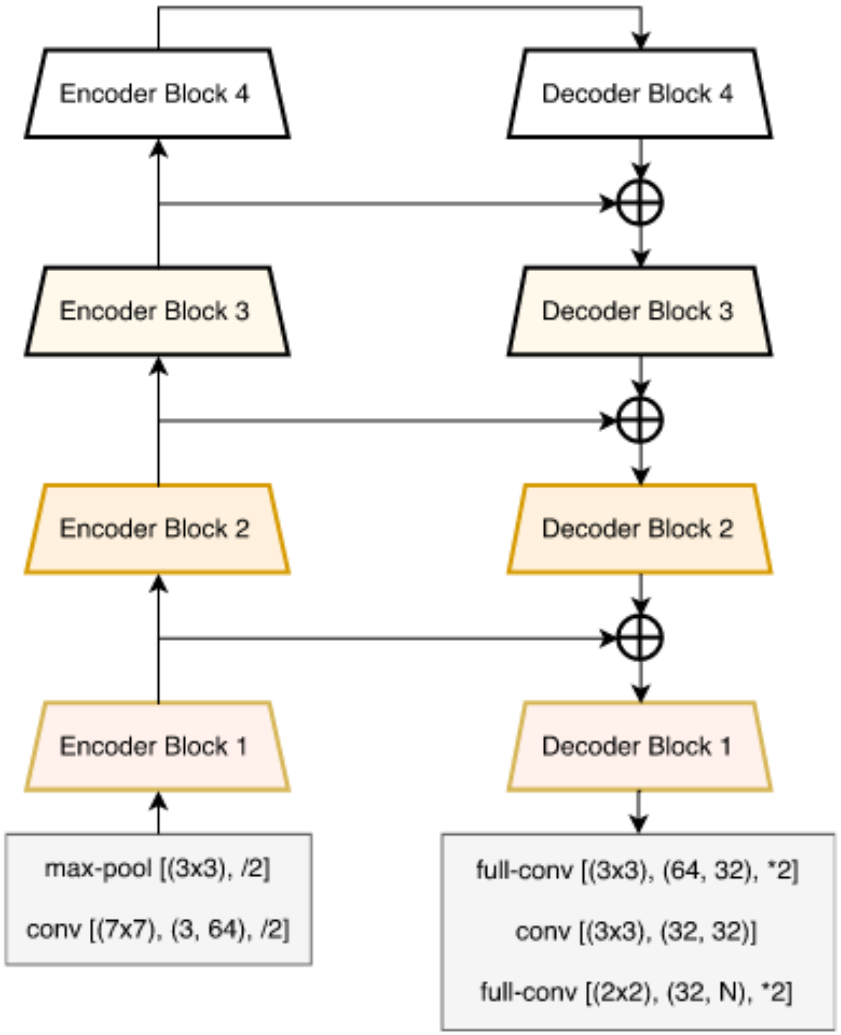
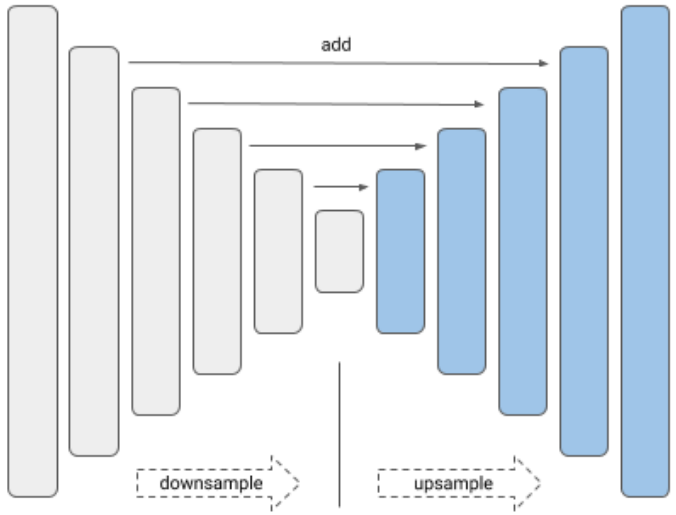


Fig. 1: LinkNet Architecture



LinkNet utilizza un'architettura a blocchi di encoder-decoder simile alla U-Net, ma con alcune modifiche:

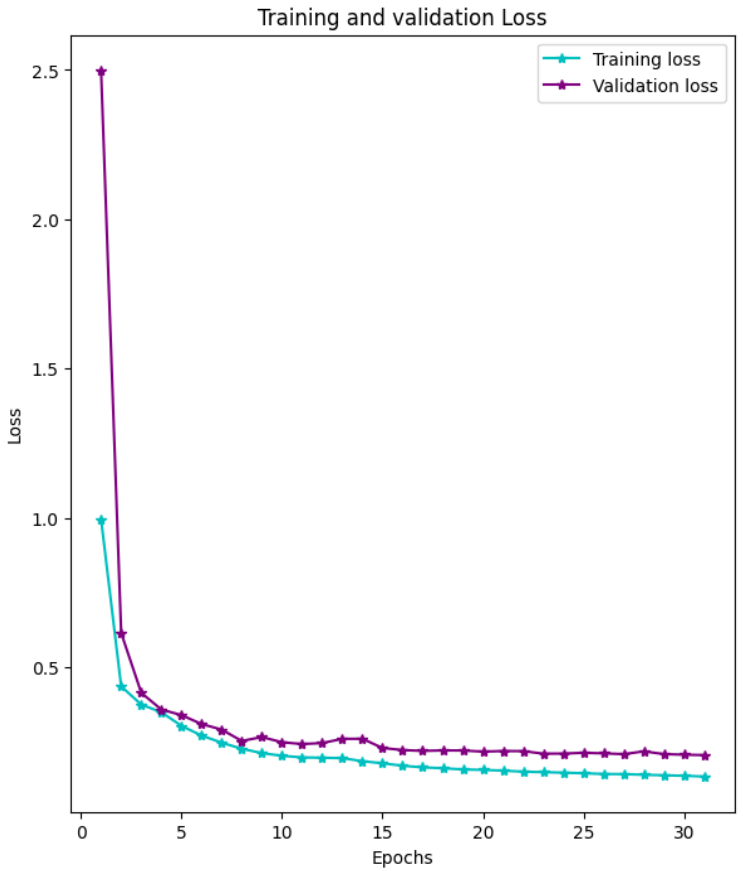
- kernel di dimensioni 7×7 (invece di un 3x3) con stride a 2 e max pooling
- Utilizza come encoder ResNet18 (quindi ha dei blocchi residuali)
- La novità principale della LinkNet è il modo in cui collega encoder e decoder: le informazioni dell'input vengono anche sempre passate all'output (evita la perdita di troppe informazioni durante downsampling)

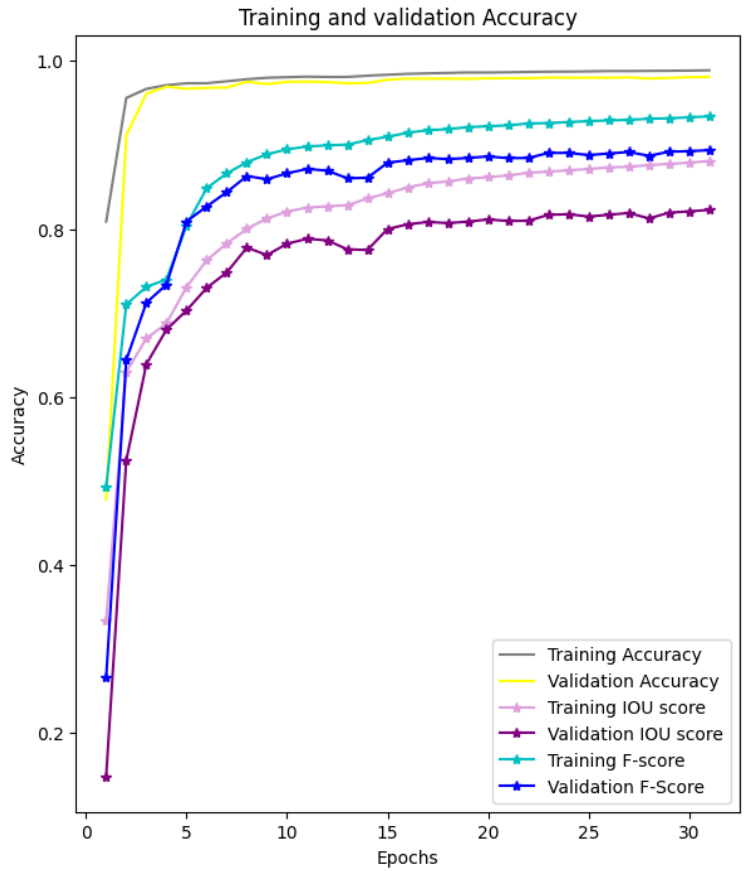
Risultati rete LinkNet

Abbiamo eseguito la rete LinkNet utilizzando come backbone 'resnet34' e preaddestramento sul dataset ImageNet su immagini di dimensione 128x128x3 utilizzando come metrica di precisione **IoU Score e F1Score,** come funzione di loss **IoU Score,** come ottimizzatore: **adam** e abbiamo aggiunto Batch Normalization

Epoche di training: 36 (media di 220s)
Accuracy sul test: 0.9801
Loss sul test: 0.2087
F1-Score: 0.8921
Mean IoU: 0.8202

Classe	IoU
Background	0.9867
Car	0.9015
Wheels	0.9003
Lights	0.5154
Windows	0.8176



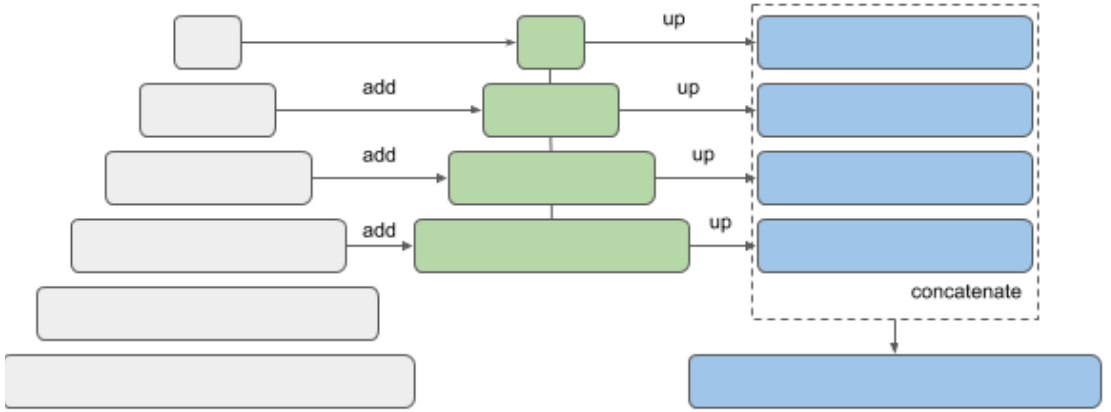


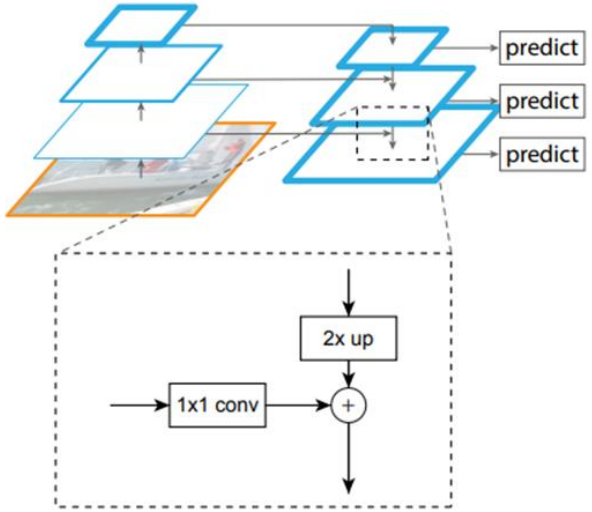
https://arxiv.org/pdf/1612.03144.pdf

5.4) FPNet (Feature Pyramide Network)

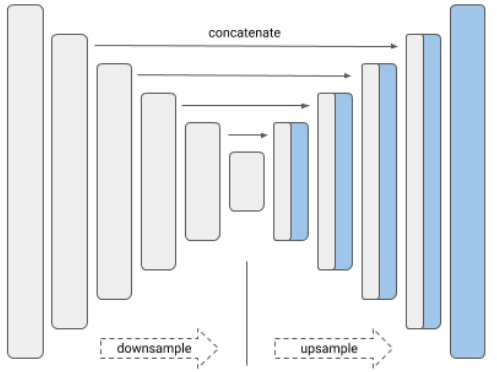
La rete FPN si basa sull'utilizzo della featurized image pyramid che permette di effettuare riconoscimento di feature su immagini di dimensioni differenti.

- FPN prende un'immagine di dimensione arbitraria e crea diverse mappe di dimensioni differenti. Il processo è indipendente dal backbone che si sceglie di usare come estrattore di feature
- La FPN è composta da un percorso bottom-up (encoder) e uno top-down(decoder). Il percorso bottom-up è la solita rete convoluzionale per l'estrazione delle caratteristiche;
- Nella sezione del decoder viene effettuato upsampling (con fattore a due e tecnica del nearest neightbor) delle diverse mappe. Il risultato è poi unito, tramite connessioni laterali, alla mappa di dimensione corrispondente dell'encoder
- L'output finale è una concatenazione delle mappe di diversa dimensione, si applica una convoluzione 3 × 3 a tutti i livelli uniti.





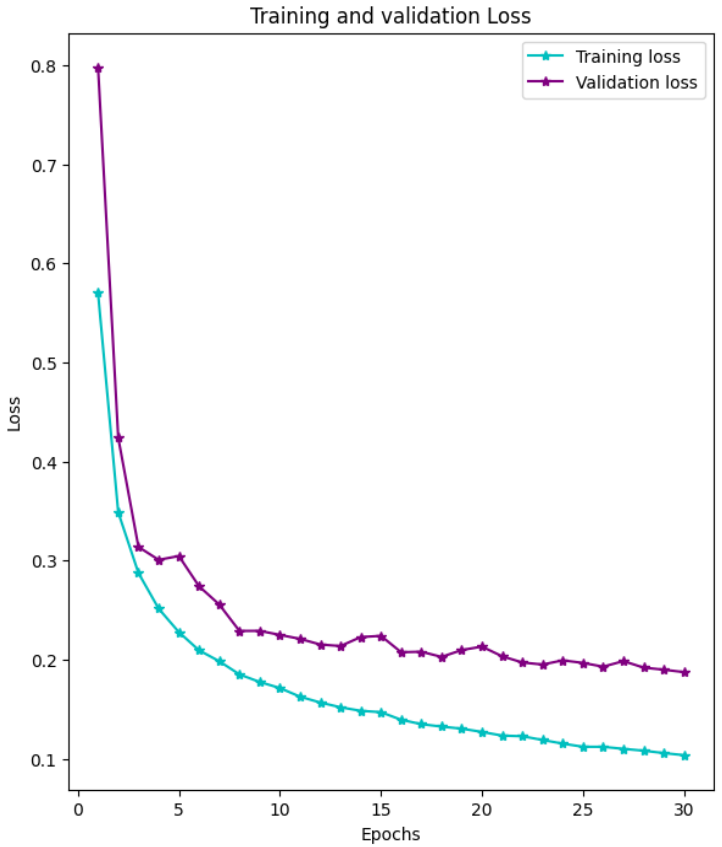
Risultati FPNet

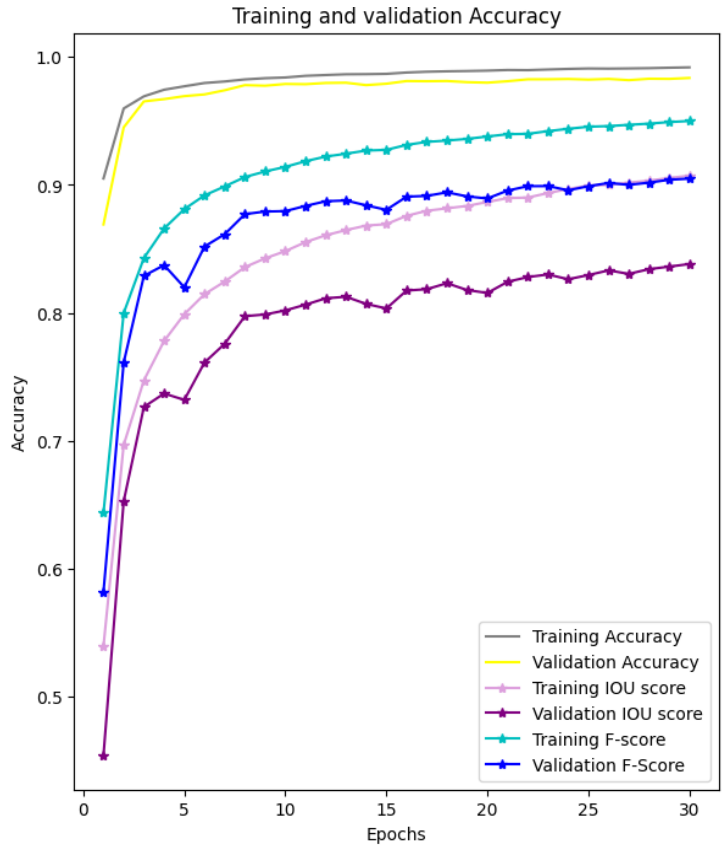


Abbiamo eseguito la rete FPNet utilizzando come backbone 'resnet34' e preaddestramento sul dataset ImageNet su immagini di dimensione 128x128x3 utilizzando come metrica di precisione **IoU Score e F1Score,** come funzione di loss **IoU Score,** come ottimizzatore: **adam** e abbiamo aggiunto Batch Normalization

Epoche di training: 35 (media di 400s)
Accuracy sul test: 0.9846
Loss sul test: 0.1745
F1 Score: 0.9121
Mean IoU: 0.8485

Classe	IoU
Background	0.9897
Car	0.9233
Wheels	0.9186
Lights	0.5798
Windows	0.8499

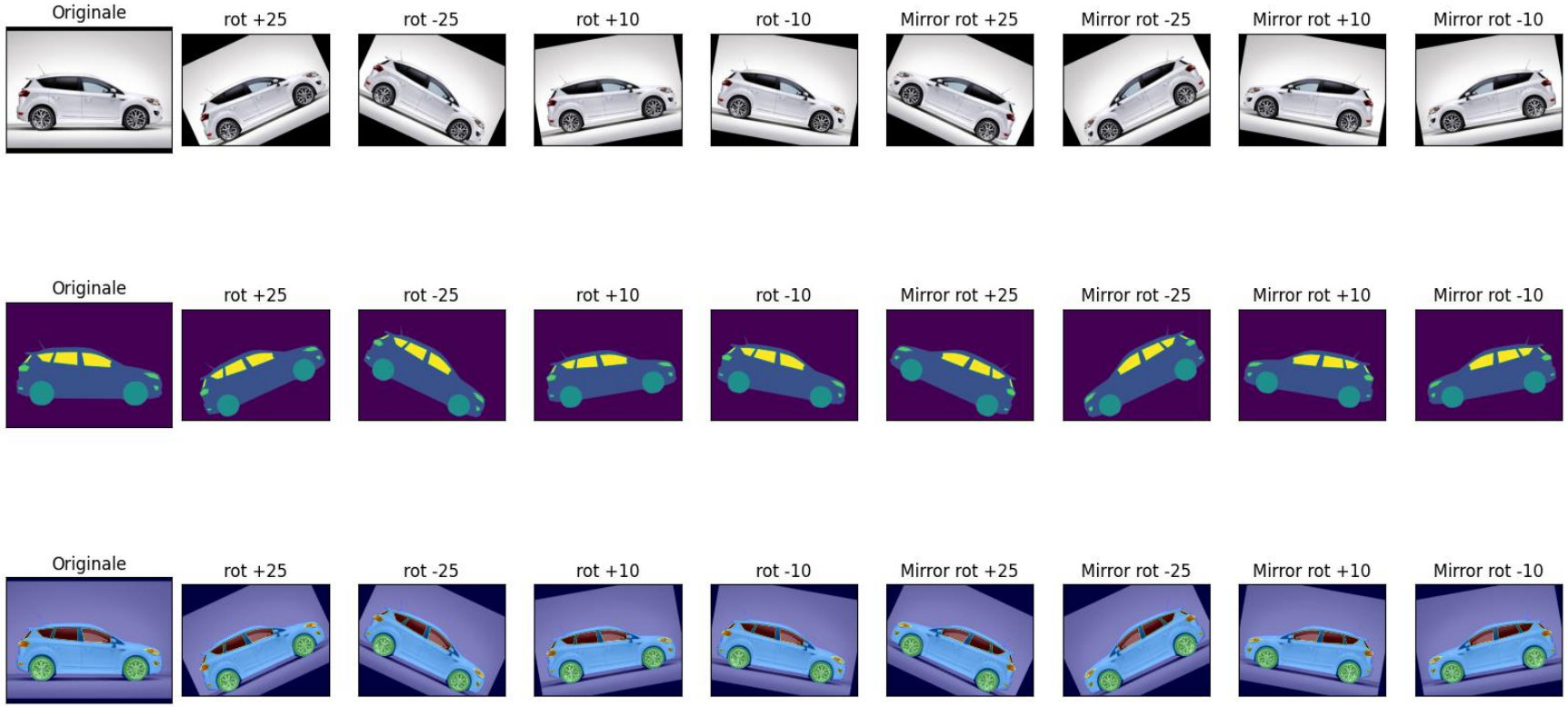




6.1) Tentativo di miglioramento: aumento ulteriore del dataset

Come ulteriore tentativo di miglioramento abbiamo provato ad incrementare ulteriormente il dataset di partenza, passando da 211 immagini e maschere a 3165 immagini e maschere
Delle trasformazioni iniziali è stata modificata la rotazione:

- Rotazione eseguita su ogni immagine con +10°, -10°, +25° e -25°
- Ogni immagine ruotata è poi stata anche specchiata

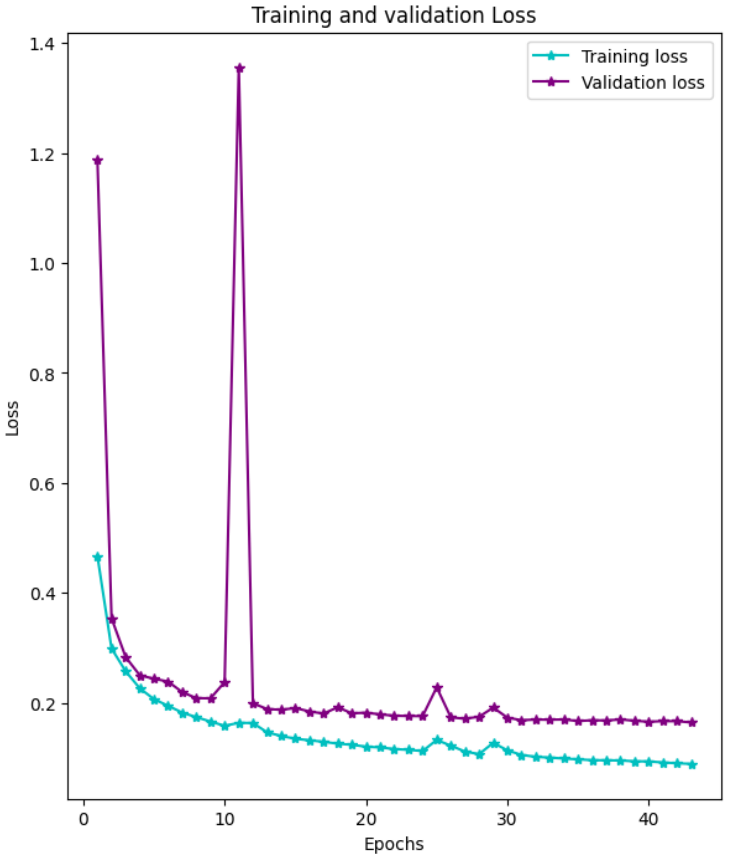


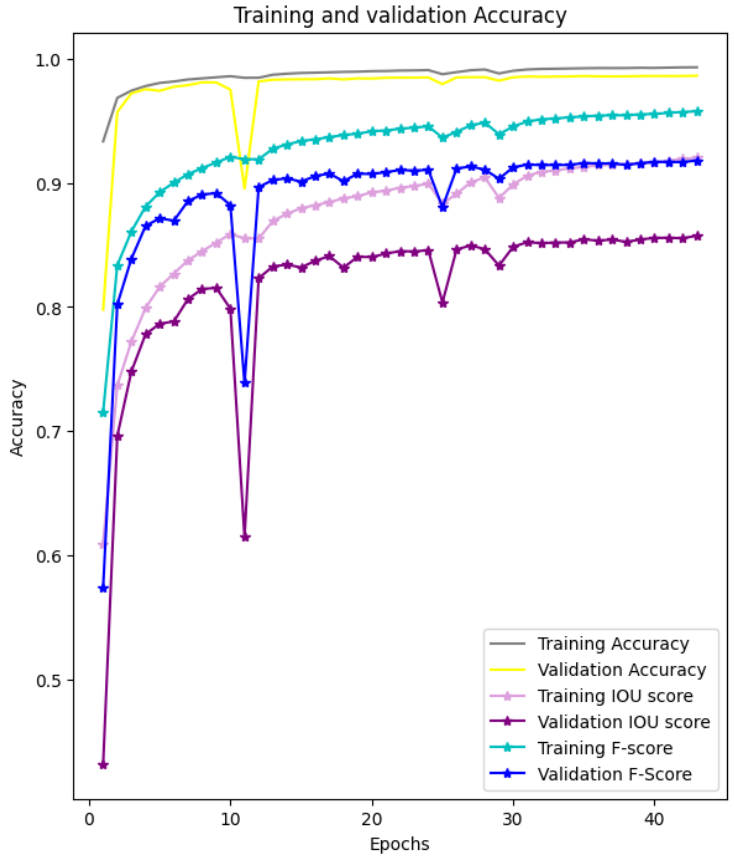
Risultati con FPNet sul dataset aumentato

Abbiamo eseguito la rete FPNet utilizzando come backbone 'resnet34' e preaddestramento sul dataset ImageNet su immagini di dimensione 128x128x3 utilizzando come metrica di precisione **IoU Score e F1Score,** come funzione di loss **IoU Score,** come ottimizzatore: **adam** e abbiamo aggiunto Batch Normalization

Epoche di training: 48 (media di 900s)
Accuracy sul test: 0.9867
Loss sul test: 0.1627
F1-Score: 0.9187
Mean IoU: 0.8586

Classe	IoU
Background	0.9915
Car	0.9292
Wheels	0.9188
Lights	0.6058
Windows	0.8610



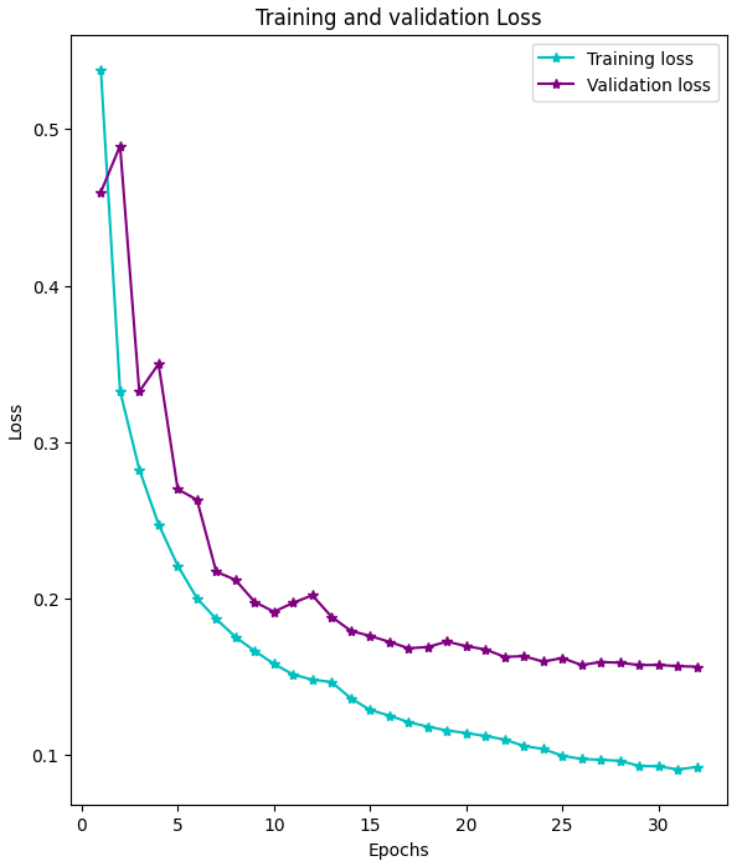


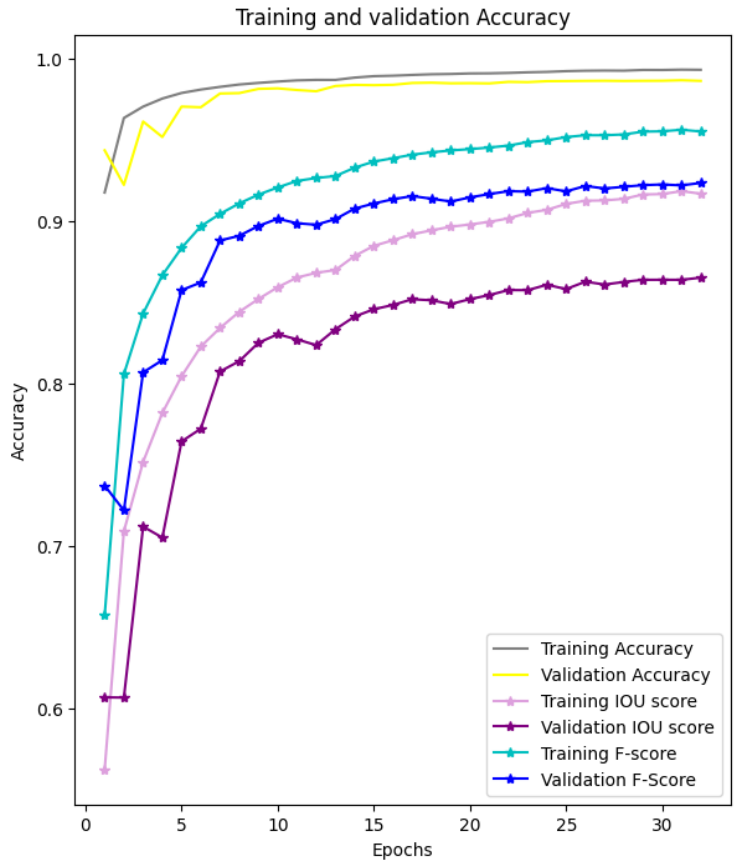
6.2) FPNet utilizzando vgg16 su 128x128

Abbiamo eseguito la rete FPNet utilizzando come backbone 'vgg16' e preaddestramento sul dataset ImageNet su immagini di dimensione 128x128x3 utilizzando come metrica di precisione **IoU Score e F1Score,** come funzione di loss **IoU Score,** come ottimizzatore: **adam** e abbiamo aggiunto Batch Normalization

Epoche di training: 37 (media di 1000s)
Accuracy sul test: 0.9858
Loss sul test: 0.1585
F1-Score: 0.9219
Mean IoU: 0.8631

Classe	IoU
Background	0.9902
Car	0.9295
Wheels	0.9253
Lights	0.6221
Windows	0.8690



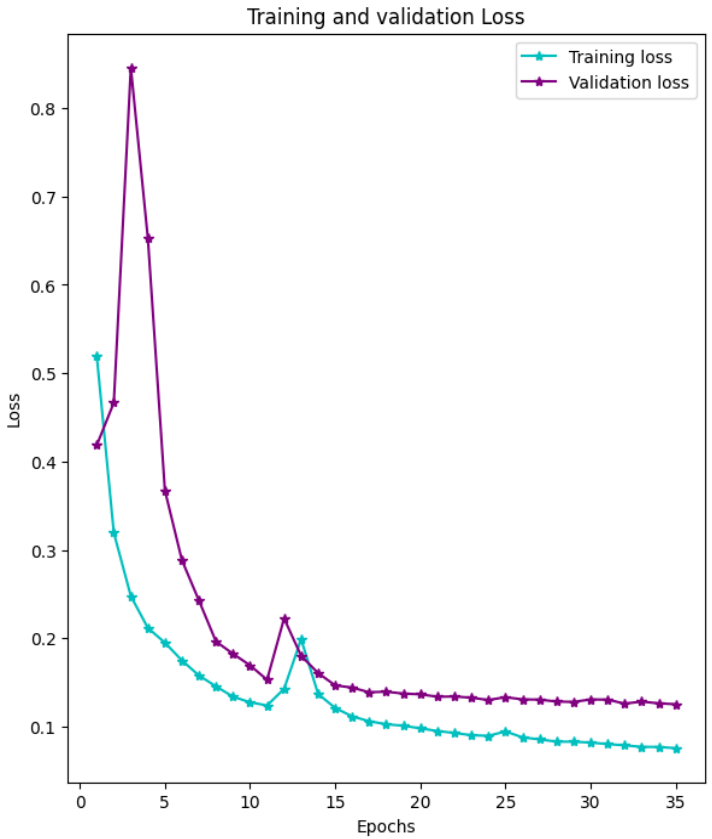


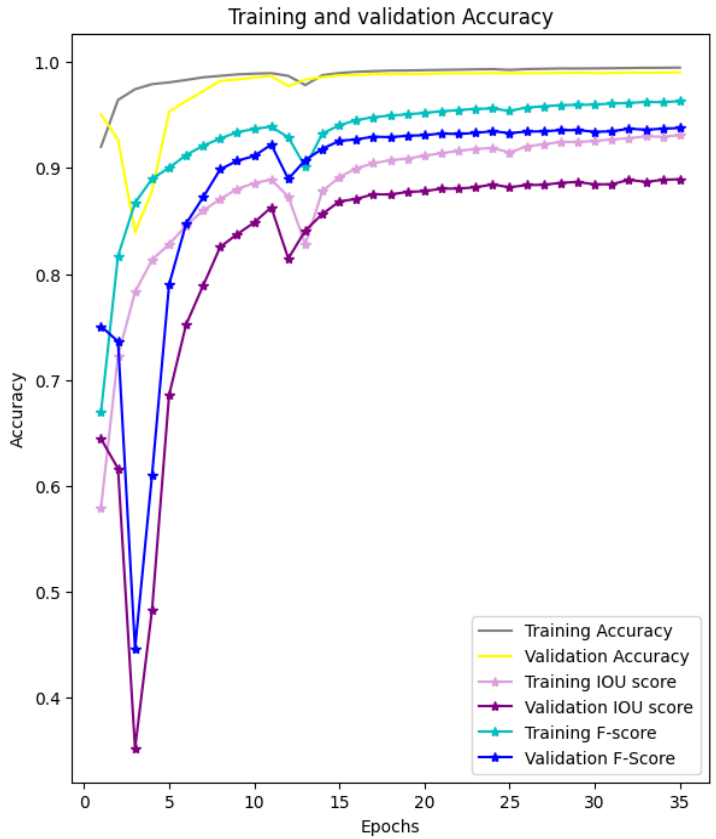
6.3) FPNet utilizzando vgg16 su 256x256

Abbiamo eseguito la rete FPNet utilizzando come backbone 'vgg16' e preaddestramento sul dataset ImageNet su immagini di dimensione 256x256x3 utilizzando come metrica di precisione **IoU Score e F1Score,** come funzione di loss **IoU Score,** come ottimizzatore: **adam** e abbiamo aggiunto Batch Normalization

Epoche di training: 40 (media di 2600s)
Accuracy sul test: 0.9904
Loss sul test: 0.1220
F1-Score: 0.9394
Mean IoU: 0.8922

Classe	IoU
Background	0.9934
Car	0.9518
Wheels	0.9439
Lights	0.6688
Windows	0.9158



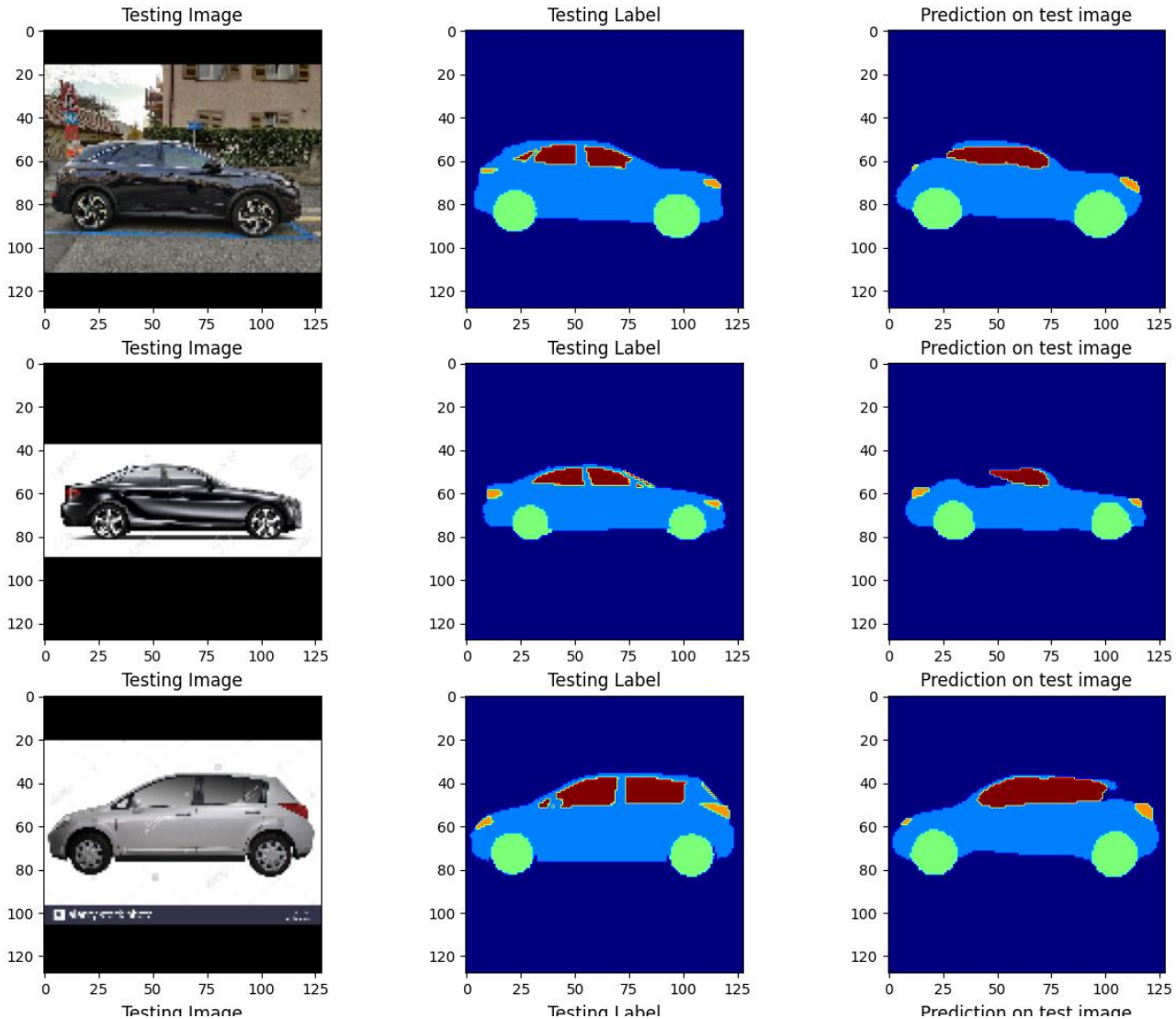


Risultati sul dataset originale (211 img)

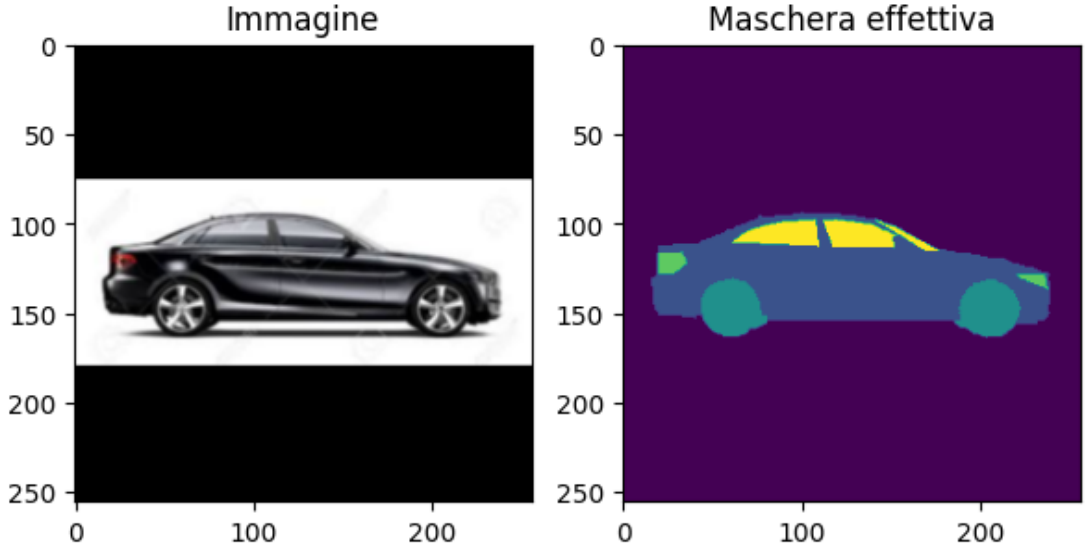
Per completezza abbiamo eseguito la rete FPNet con backbone 'vgg16' anche sul dataset iniziale composto solo da 211 immagini e maschere

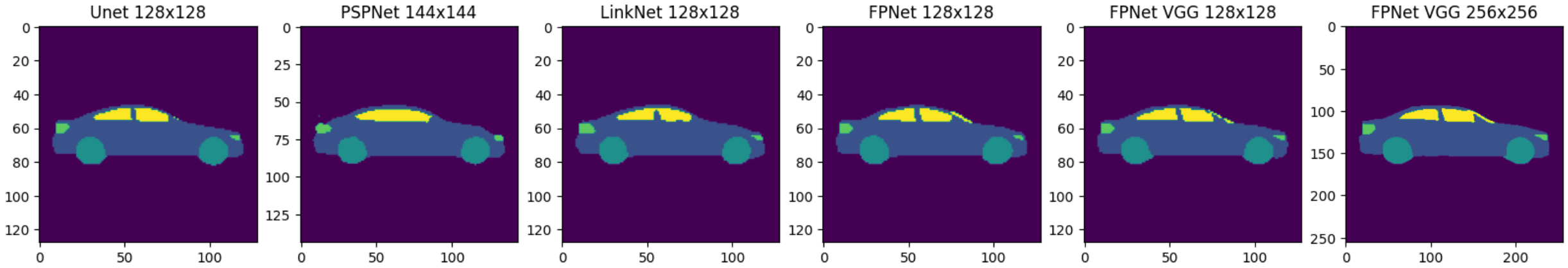
Epoche di training: 13 (media di 130s)
Accuracy sul test: 0.9565
Loss sul test: 0.4346
F1-Score: 0.7727
Mean IoU: 0.6200

Classe	IoU
Background	0.9623
Car	0.7383
Wheels	0.8186
Lights	0.2526
Windows	0.6316



- Confronto tra i risultati delle varie reti





4° 6° 5° 3° 2° 1°

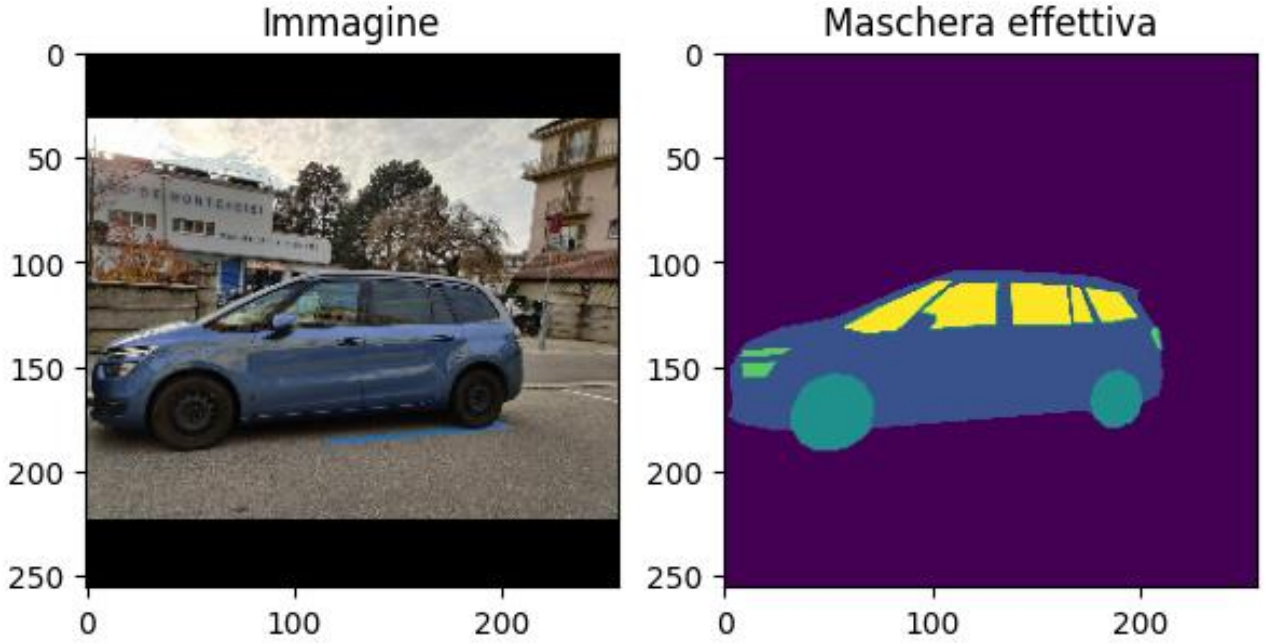
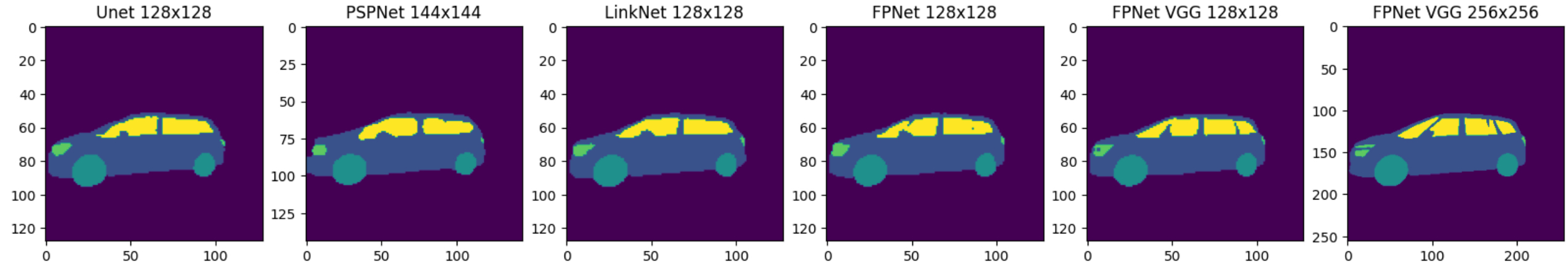
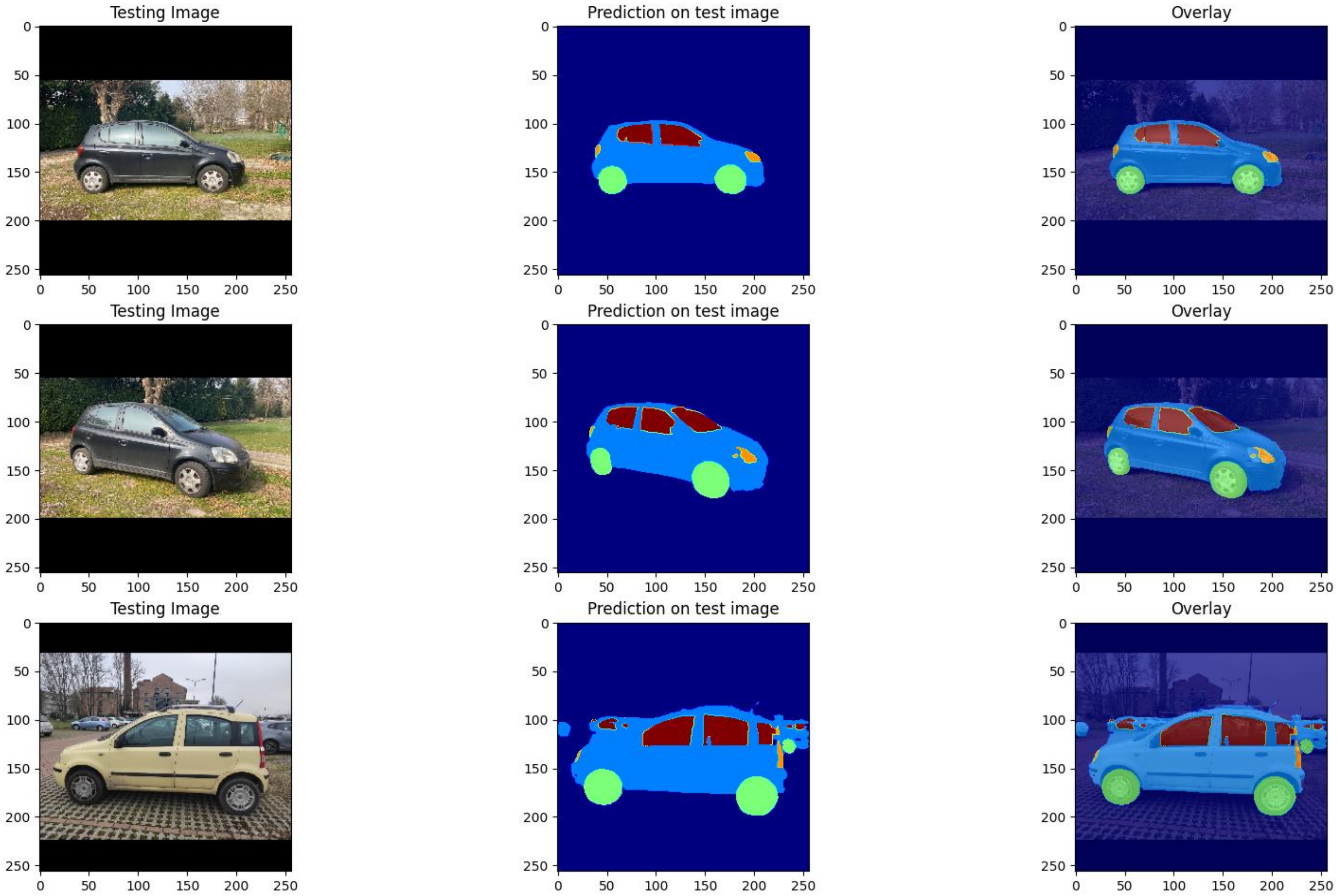


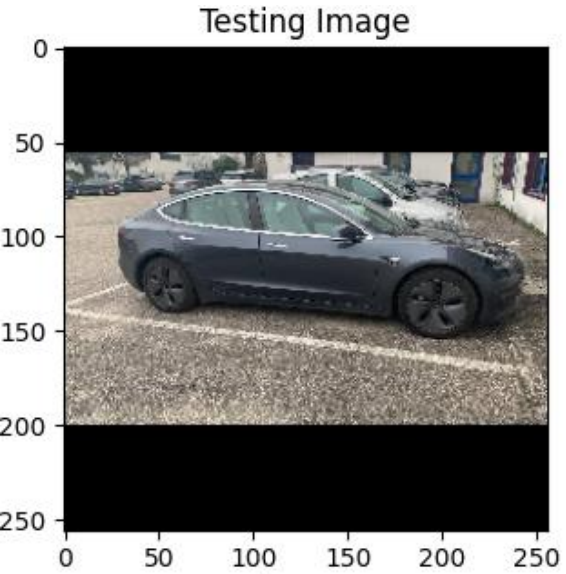
Immagine
Maschera effettiva

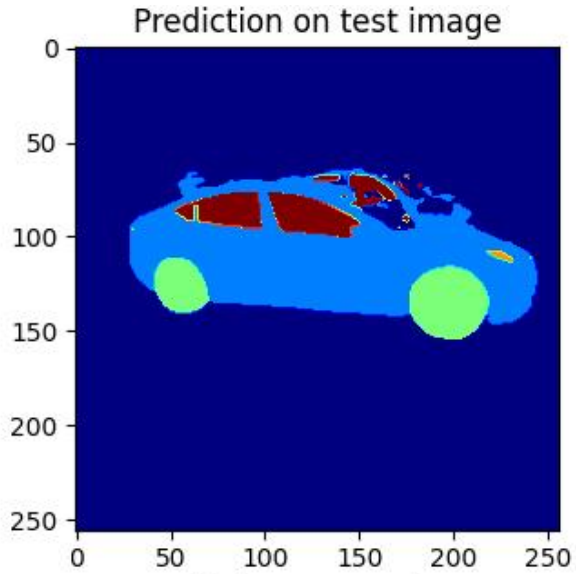

Unet 128x128
PSPNet 144x144
LinkNet 128x128
FPNet 128x128
FPNet VGG 128x128
FPNet VGG 256x256

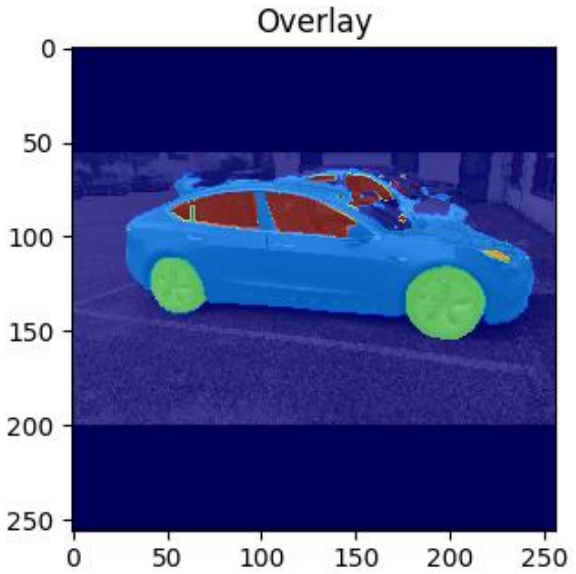
4° 6° 5° 3° 2° 1°

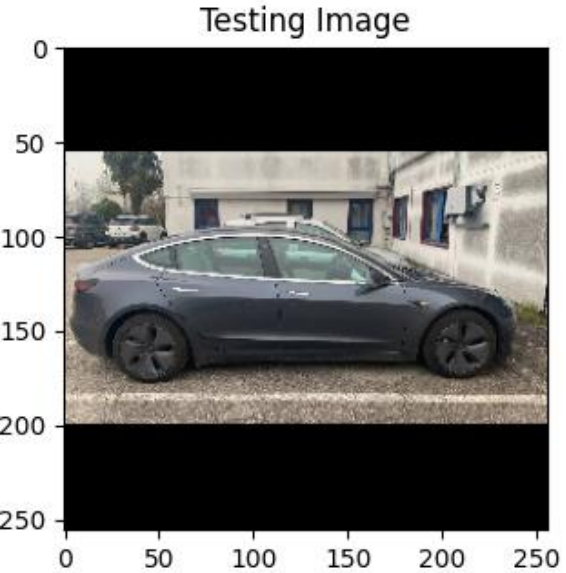
- Applicazione della rete sulle nostre auto

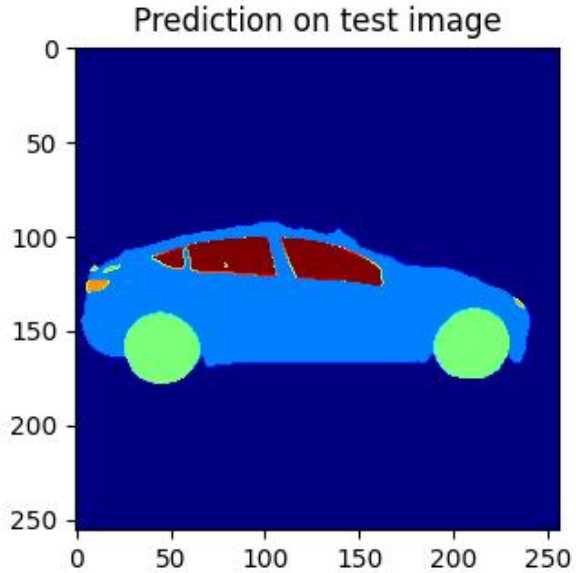


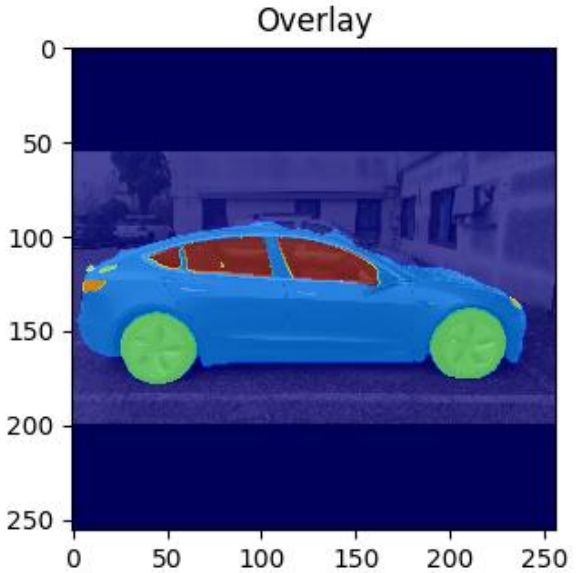

Testing Image
0
50
100
150
200
250
0
50
100
150
200
250


Prediction on test image
0
50
100
150
200
250
0
50
100
150
200
250


Overlay
0
50
100
150
200
250
0
50
100
150
200
250


Testing Image
0
50
100
150
200
250
0
50
100
150
200
250


Prediction on test image
0
50
100
150
200
250
0
50
100
150
200
250


Overlay
0
50
100
150
200
250
0
50
100
150
200
250

- **Possibili migliorie future:**
- **Utilizzo di immagini di dimensione maggiore di 256x256**
- **Utilizzo di un dataset di dimensioni maggiori**

FINE