# Ocenjevanje homografije

## Zbirke slik

#### Predpriprava MS Coco 2017 dataset

from Generator import prepair\_dataset  
  
INPUT\_DIR = "datasets/val2017"  
PREPROCESSED\_DIR = "datasets/val2017\_preprocessed"  
  
TARGET\_SIZE = (320, 240)  
prepair\_dataset(INPUT\_DIR, PREPROCESSED\_DIR, TARGET\_SIZE)

#### Generirane testnih primerov

from Generator import visualize\_generate\_pair  
  
*# Run visualization*for \_ in range(3):  
 visualize\_generate\_pair(PREPROCESSED\_DIR)

A close-up of a dog's face

AI-generated content may be incorrect.

A close-up of a person's face

AI-generated content may be incorrect.

A close-up of a screen

AI-generated content may be incorrect.

A close-up of a person's face

AI-generated content may be incorrect.

A close-up of a horse

AI-generated content may be incorrect.

A close-up of a machine

AI-generated content may be incorrect.

A collage of a baseball player

AI-generated content may be incorrect.

## Nevronski mreži

##### ResNet Block

from torchinfo import summary  
from Models import ResNetBlock  
  
*# For ResNet block*block = ResNetBlock(in\_channels=64, out\_channels=128, stride=2, dropout\_rate=0.1)  
summary(block, input\_size=(1, 64, 32, 32))

==========================================================================================

Layer (type:depth-idx) Output Shape Param #

==========================================================================================

ResNetBlock [1, 128, 16, 16] --

├─Conv2d: 1-1 [1, 128, 16, 16] 8,320

├─Conv2d: 1-2 [1, 128, 16, 16] 73,856

├─Dropout2d: 1-3 [1, 128, 16, 16] --

├─ReLU: 1-4 [1, 128, 16, 16] --

├─Conv2d: 1-5 [1, 128, 16, 16] 147,584

├─Dropout2d: 1-6 [1, 128, 16, 16] --

├─ReLU: 1-7 [1, 128, 16, 16] --

==========================================================================================

Total params: 229,760

Trainable params: 229,760

Non-trainable params: 0

Total mult-adds (Units.MEGABYTES): 58.82

==========================================================================================

Input size (MB): 0.26

Forward/backward pass size (MB): 0.79

Params size (MB): 0.92

Estimated Total Size (MB): 1.97

==========================================================================================

##### ResNet Body

from torchinfo import summary  
from Models import ResNetBody  
  
*# For ResNet body*body = ResNetBody(in\_channels=2, dropout\_rate=0.1)  
summary(body, input\_size=(1, 2, 64, 64),depth=2)

==========================================================================================

Layer (type:depth-idx) Output Shape Param #

==========================================================================================

ResNetBody [1, 512] --

├─Sequential: 1-1 [1, 64, 32, 32] --

│ └─ResNetBlock: 2-1 [1, 64, 64, 64] 38,336

│ └─ResNetBlock: 2-2 [1, 64, 64, 64] 73,856

│ └─BatchNorm2d: 2-3 [1, 64, 64, 64] 128

│ └─ReLU: 2-4 [1, 64, 64, 64] --

│ └─MaxPool2d: 2-5 [1, 64, 32, 32] --

├─Sequential: 1-2 [1, 64, 16, 16] --

│ └─ResNetBlock: 2-6 [1, 64, 32, 32] 73,856

│ └─ResNetBlock: 2-7 [1, 64, 32, 32] 73,856

│ └─BatchNorm2d: 2-8 [1, 64, 32, 32] 128

│ └─ReLU: 2-9 [1, 64, 32, 32] --

│ └─MaxPool2d: 2-10 [1, 64, 16, 16] --

├─Sequential: 1-3 [1, 128, 8, 8] --

│ └─ResNetBlock: 2-11 [1, 128, 16, 16] 229,760

│ └─ResNetBlock: 2-12 [1, 128, 16, 16] 295,168

│ └─BatchNorm2d: 2-13 [1, 128, 16, 16] 256

│ └─ReLU: 2-14 [1, 128, 16, 16] --

│ └─MaxPool2d: 2-15 [1, 128, 8, 8] --

├─Sequential: 1-4 [1, 128, 8, 8] --

│ └─ResNetBlock: 2-16 [1, 128, 8, 8] 295,168

│ └─ResNetBlock: 2-17 [1, 128, 8, 8] 295,168

│ └─BatchNorm2d: 2-18 [1, 128, 8, 8] 256

│ └─ReLU: 2-19 [1, 128, 8, 8] --

├─Flatten: 1-5 [1, 8192] --

├─Linear: 1-6 [1, 512] 4,194,816

==========================================================================================

Total params: 5,570,752

Trainable params: 5,570,752

Non-trainable params: 0

Total mult-adds (Units.MEGABYTES): 787.15

==========================================================================================

Input size (MB): 0.03

Forward/backward pass size (MB): 17.11

Params size (MB): 22.28

Estimated Total Size (MB): 39.42

==========================================================================================

##### Homography Regressor

from torchinfo import summary  
  
*# For regression model*model = HomographyRegressor()  
summary(model, input\_size=(1, 2, 64, 64))

==========================================================================================

Layer (type:depth-idx) Output Shape Param #

==========================================================================================

HomographyRegressor [1, 8] --

├─ResNetBody: 1-1 [1, 512] --

│ └─Sequential: 2-1 [1, 64, 32, 32] 112,320

│ └─Sequential: 2-2 [1, 64, 16, 16] 147,840

│ └─Sequential: 2-3 [1, 128, 8, 8] 525,184

│ └─Sequential: 2-4 [1, 128, 8, 8] 590,592

│ └─Flatten: 2-5 [1, 8192] --

│ └─Linear: 2-6 [1, 512] 4,194,816

├─RegressionHead: 1-2 [1, 8] --

│ └─Linear: 2-7 [1, 8] 4,104

==========================================================================================

Total params: 5,574,856

Trainable params: 5,574,856

Non-trainable params: 0

Total mult-adds (Units.MEGABYTES): 787.16

==========================================================================================

Input size (MB): 0.03

Forward/backward pass size (MB): 17.11

Params size (MB): 22.30

Estimated Total Size (MB): 39.44

==========================================================================================

##### Homography Classifier

from torchinfo import summary  
  
*# For classification model*model = HomographyClassifier(num\_classes=21, class\_dim=8)  
summary(model, input\_size=(1, 2, 64, 64), depth=2)

==========================================================================================

Layer (type:depth-idx) Output Shape Param #

==========================================================================================

HomographyClassifier [1, 8, 21] --

├─ResNetBody: 1-1 [1, 512] --

│ └─Sequential: 2-1 [1, 64, 32, 32] 112,320

│ └─Sequential: 2-2 [1, 64, 16, 16] 147,840

│ └─Sequential: 2-3 [1, 128, 8, 8] 525,184

│ └─Sequential: 2-4 [1, 128, 8, 8] 590,592

│ └─Flatten: 2-5 [1, 8192] --

│ └─Linear: 2-6 [1, 512] 4,194,816

├─ClassificationHead: 1-2 [1, 8, 21] --

│ └─Linear: 2-7 [1, 168] 86,184

==========================================================================================

Total params: 5,656,936

Trainable params: 5,656,936

Non-trainable params: 0

Total mult-adds (Units.MEGABYTES): 787.24

==========================================================================================

Input size (MB): 0.03

Forward/backward pass size (MB): 17.11

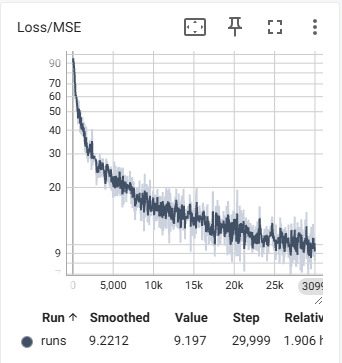
Params size (MB): 22.63

Estimated Total Size (MB): 39.77

==========================================================================================

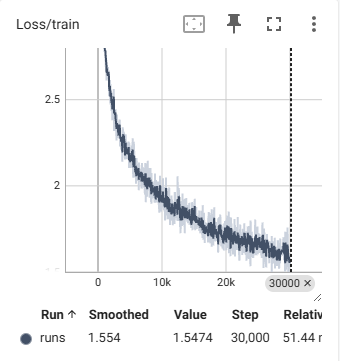
## Učenje

### Regression model (30k epochs, 64 samples)



(Logaritemska skala)

### Classification model (30k epochs, 64 samples)



## Rezultati

##### Regresija

*# visualization of regression results*device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")  
model = HomographyRegressor().to(device)  
state = torch.load("checkpoints\_homography\_regressor\_all/h\_regressor\_all.pth")  
model.load\_state\_dict(state)  
model.eval()  
  
img = get\_random\_images(1, image\_dir=PREPROCESSED\_DIR)[0]  
visualize\_regression\_result(model=model, image=img)

A screenshot of a black and white photo of motorcycles

AI-generated content may be incorrect.

A bus on the road

AI-generated content may be incorrect.

A cow walking on the street

AI-generated content may be incorrect.

##### Klasifikacija

*# visualization of classification results*device = torch.device("cuda" if torch.cuda.is\_available() else "cpu")  
model = HomographyClassifier(num\_classes=21, class\_dim=8).to(device)  
state = torch.load("checkpoints\_homography\_classify\_all/checkpoint\_epoch\_30000.pth")["model\_state\_dict"]  
model.load\_state\_dict(state)  
model.eval()  
  
img = get\_random\_images(1, image\_dir=PREPROCESSED\_DIR)[0]  
visualize\_classification\_result(model=model, image=img, soft\_decode=True)

A screenshot of a building

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a room

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

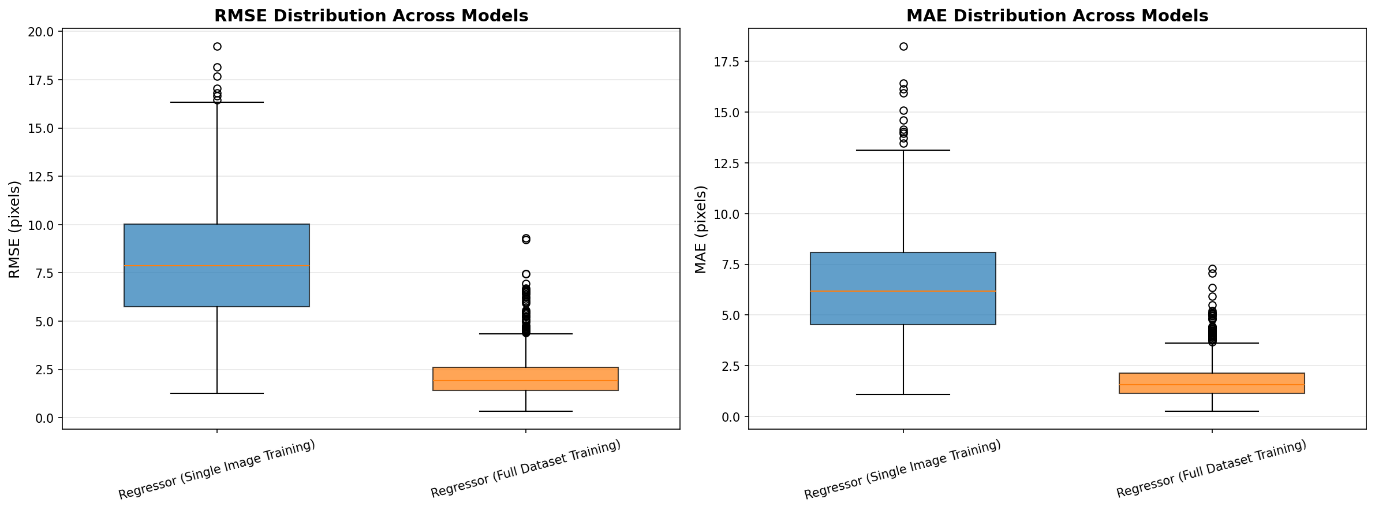
A screenshot of a computer

AI-generated content may be incorrect.

## Evaluacija

##### Regression models

images = get\_random\_images(100, "datasets/val2017\_preprocessed")  
test\_samples = generate\_test\_set(images, samples\_per\_image=10)  
  
*# Compare regression models*regressor\_single = HomographyRegressor()  
regressor\_single.load\_state\_dict(torch.load("checkpoints\_homography\_regressor\_single/h\_regressor\_single.pth"))  
  
regressor\_all = HomographyRegressor()  
regressor\_all.load\_state\_dict(torch.load("checkpoints\_homography\_regressor\_all/h\_regressor\_all.pth"))  
  
results = {  
 'Regressor (Single Image Training)': evaluate\_regressor(regressor\_single, test\_samples),  
 'Regressor (Full Dataset Training)': evaluate\_regressor(regressor\_all, test\_samples)  
}  
  
summarize\_and\_plot(results, save\_dir="eval\_results")



================================================================================

EVALUATION SUMMARY

================================================================================

ocena z Regressor (Single Image Training) mean: 10.29 std: 6.52

ocena z Regressor (Full Dataset Training) mean: 2.77 std: 2.11

================================================================================

##### Classification models

*# Compare classifiers (soft vs hard decoding)*classifier = HomographyClassifier()  
classifier.load\_state\_dict(torch.load("checkpoints\_homography\_classify\_all/h\_classify\_all.pth"))  
  
results = {  
 'Classifier (soft)': evaluate\_classifier(classifier, test\_samples, soft\_decode=True),  
 'Classifier (hard)': evaluate\_classifier(classifier, test\_samples, soft\_decode=False)  
}  
  
summarize\_and\_plot(results, save\_dir="eval\_results")



================================================================================

EVALUATION SUMMARY

================================================================================

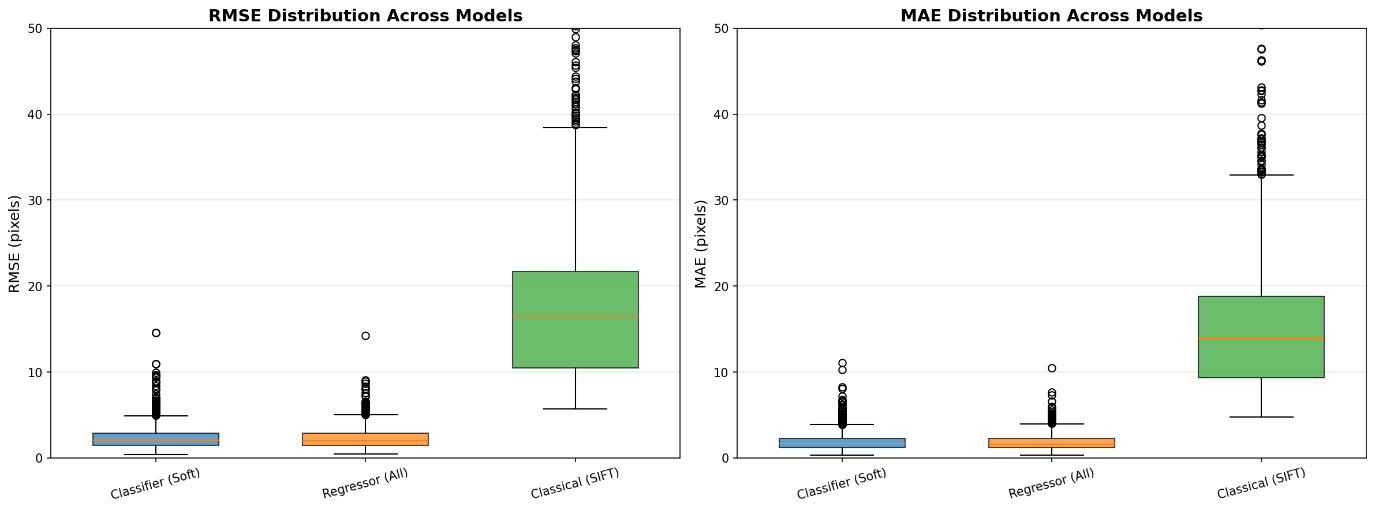
ocena z Classifier (soft) mean: 2.95 std: 2.56

ocena z Classifier (hard) mean: 3.40 std: 3.05

================================================================================

##### Comparing best models with classical

results = {  
 'Classifier (Soft)': evaluate\_classifier(classifier, test\_samples, soft\_decode=True),  
 'Regressor (All)': evaluate\_regressor(regressor\_all, test\_samples),  
 'Classical (SIFT)': evaluate\_classical(test\_samples)  
}  
  
summarize\_and\_plot(results, save\_dir="eval\_results", ylim=50)



================================================================================

EVALUATION SUMMARY

================================================================================

ocena z Classifier (Soft) mean: 3.05 std: 2.74

ocena z Regressor (All) mean: 2.96 std: 2.37

ocena z Classical (SIFT) mean: 24.30 std: 21.01

================================================================================