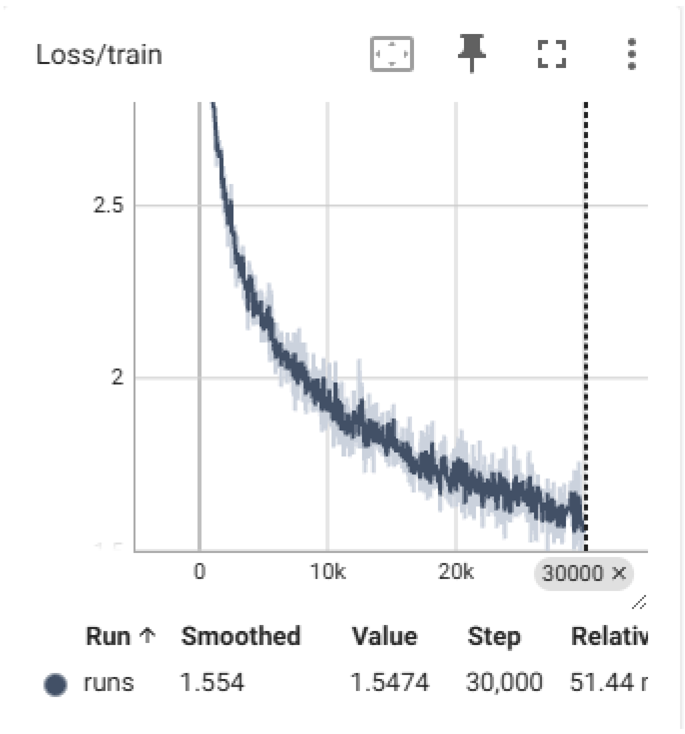


Run ↑	Smoothed	Value	Step	Relativ
● runs	1.554	1.5474	30,000	51.44 r

Ocenjevanje homografije

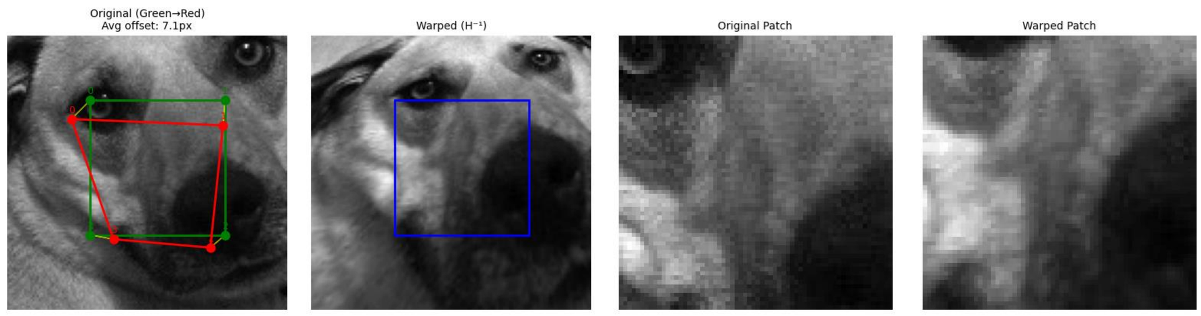
Zbirke slik

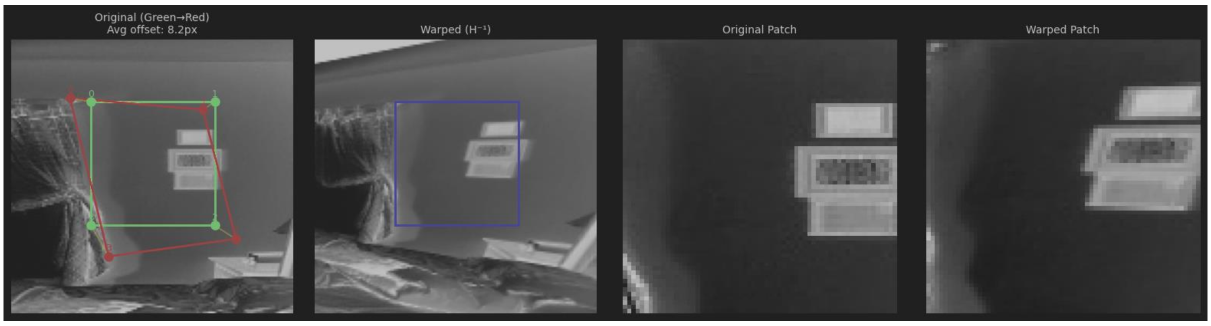
Predpriprava MS Coco 2017 dataset

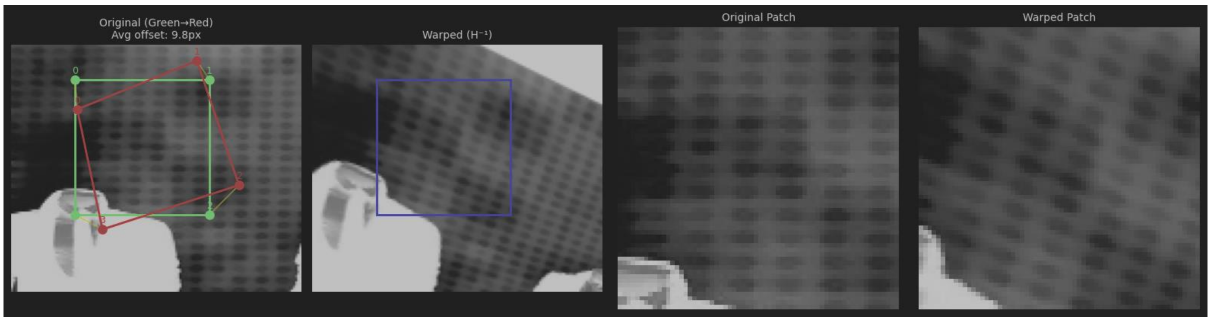
```
from Generator import prepair_dataset

INPUT_DIR = "datasets/val2017"
PREPROCESSED_DIR = "datasets/val2017_preprocessed"

TARGET_SIZE = (320, 240)
prepair_dataset(INPUT_DIR, PREPROCESSED_DIR, TARGET_SIZE)
```

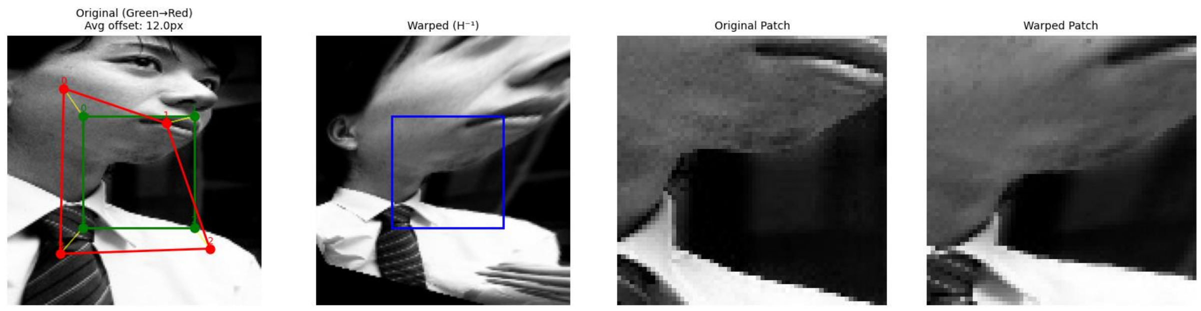
Generirane testnih primerov

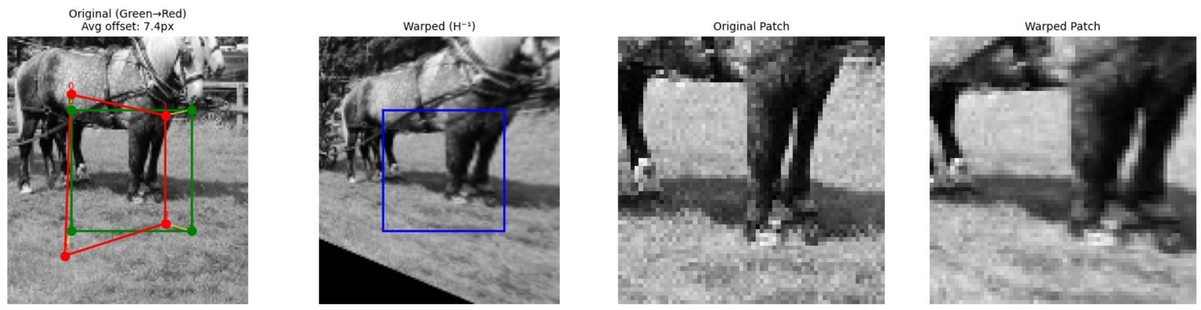
```
from Generator import visualize_generate_pair

# Run visualization
for _ in range(3):
    visualize_generate_pair(PREPROCESSED_DIR)
```

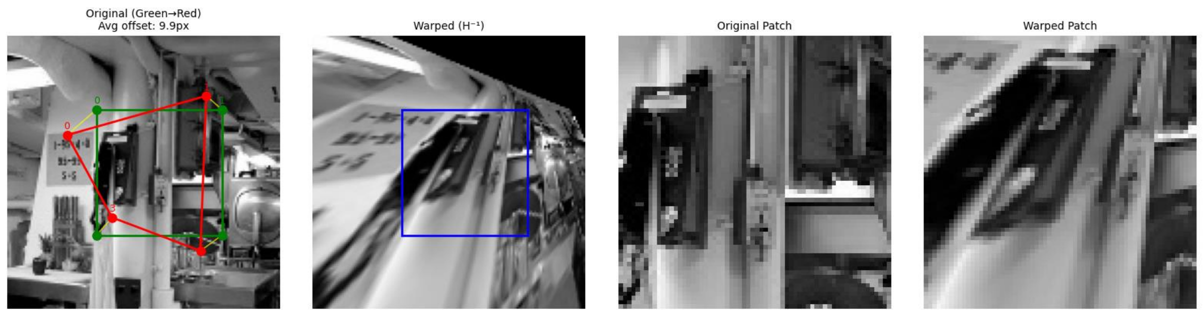

Original (Green→Red)
Avg offset: 7.1px
Warped (H⁻¹)
Original Patch
Warped Patch

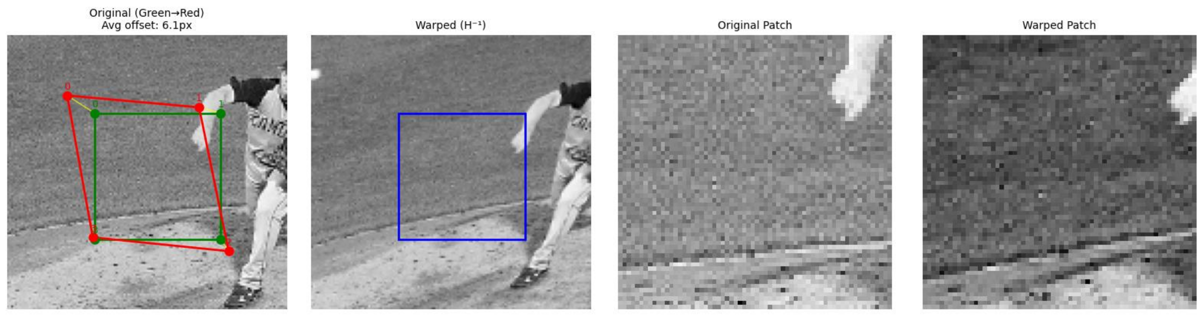

Original (Green→Red)
Avg offset: 8.2px
Warped (H⁻¹)
Original Patch
Warped Patch


Original (Green→Red)
Avg offset: 9.8px
Warped (H⁻¹)
Original Patch
Warped Patch


Original (Green→Red)
Avg offset: 12.0px
Warped (H⁻¹)
Original Patch
Warped Patch


Original (Green→Red)
Avg offset: 7.4px
Warped (H⁻¹)
Original Patch
Warped Patch


Original (Green→Red)
Avg offset: 9.9px
Warped (H⁻¹)
Original Patch
Warped Patch


Original (Green→Red)
Avg offset: 6.1px
Warped (H⁻¹)
Original Patch
Warped Patch

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

```python
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
==========================================================================================
```

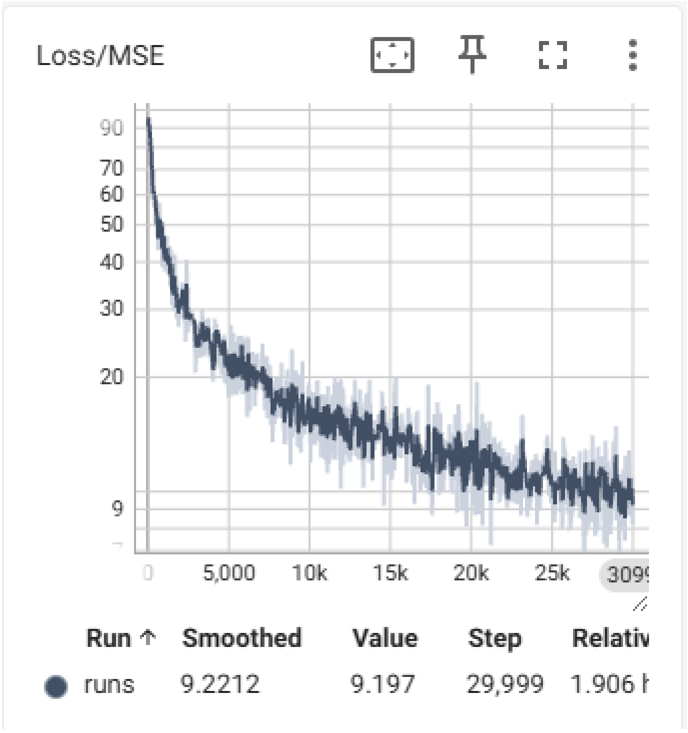
Homography Classifier

```
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

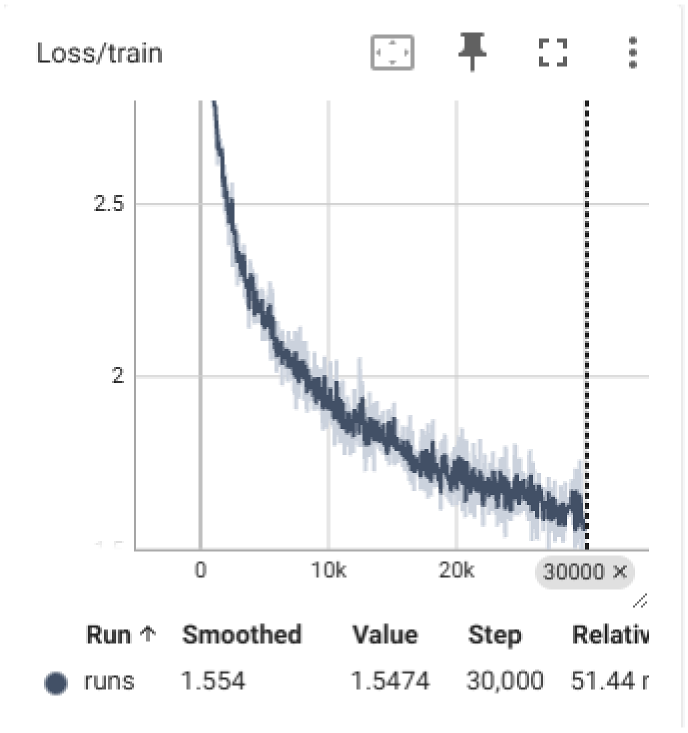
Regression model (30k epochs, 64 samples)



	Run ↑	Smoothed	Value	Step	Relativ
●	runs	9.2212	9.197	29,999	1.906 h

(LOGARITEMSKA SKALA)

Classification model (30k epochs, 64 samples)

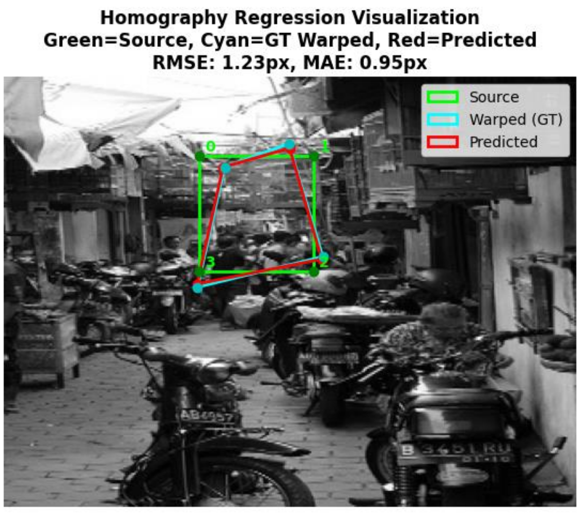


	Run ↑	Smoothed	Value	Step	Relativ
●	runs	1.554	1.5474	30,000	51.44 r

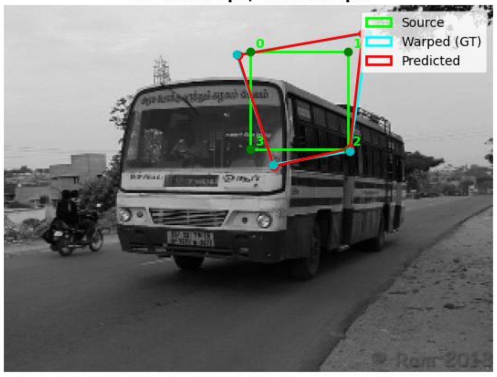
Rezultati

Regresija

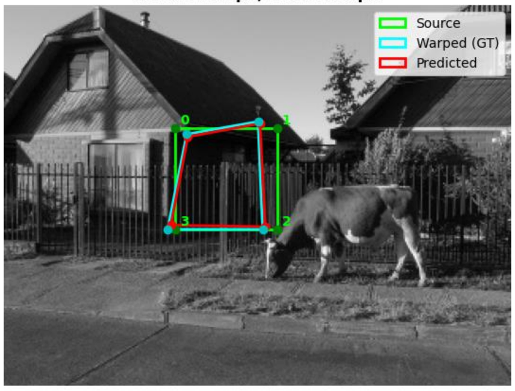
```
# visualization of regression results

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)
```


Homography Regression Visualization
Green=Source, Cyan=GT Warped, Red=Predicted
RMSE: 1.23px, MAE: 0.95px
Source
Warped (GT)
Predicted
0
1
2
3

Homography Regression Visualization
Green=Source, Cyan=GT Warped, Red=Predicted
RMSE: 1.43px, MAE: 0.95px

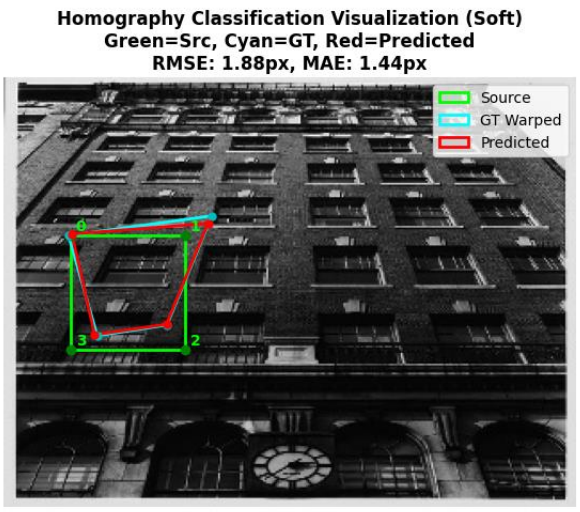


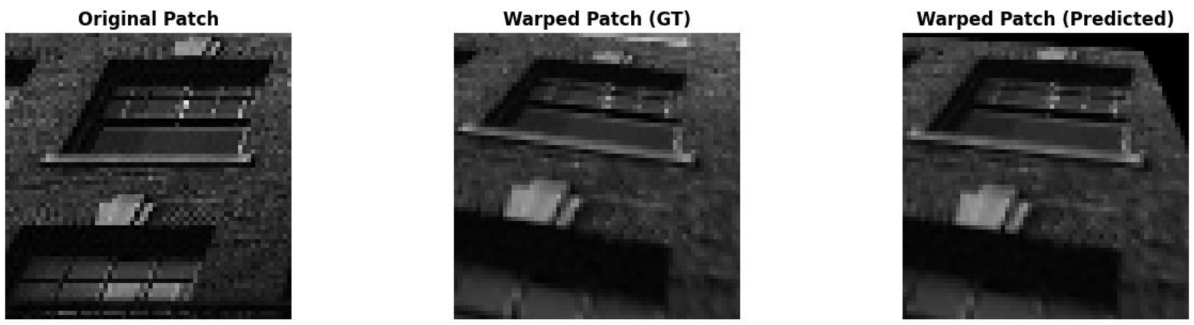
Homography Regression Visualization
Green=Source, Cyan=GT Warped, Red=Predicted
RMSE: 1.67px, MAE: 1.58px

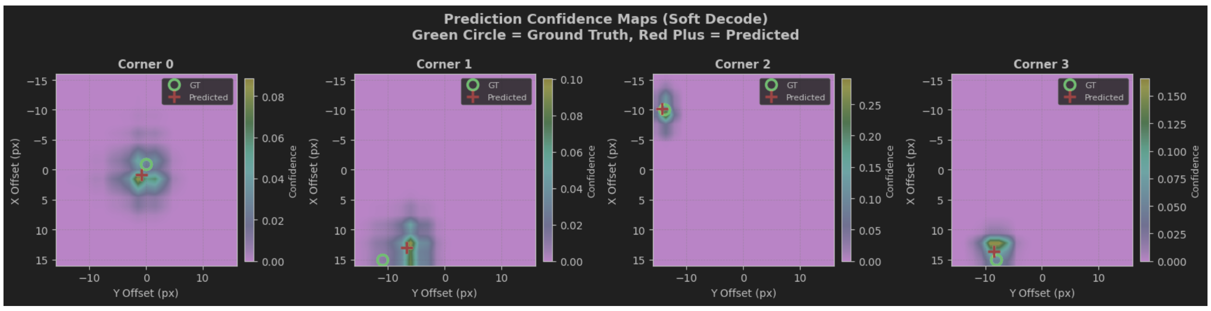


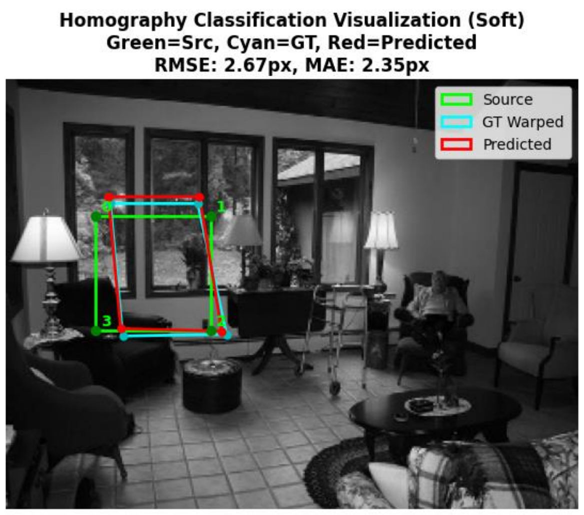
Klasifikacija

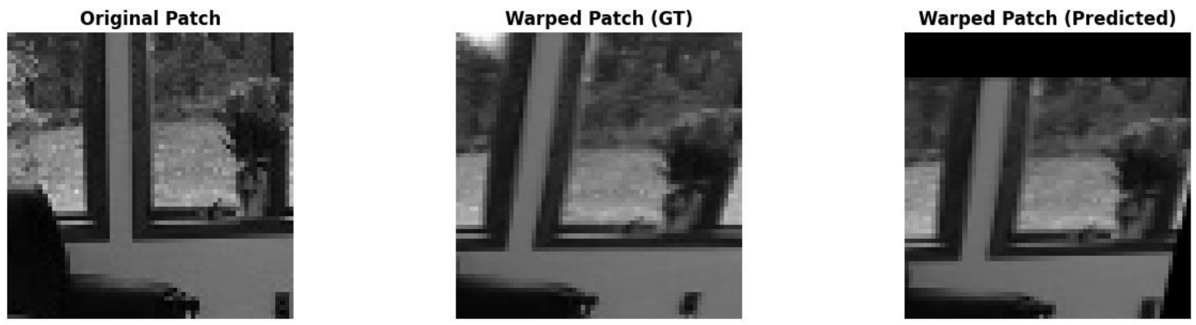
```
# visualization of classification results

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)
```

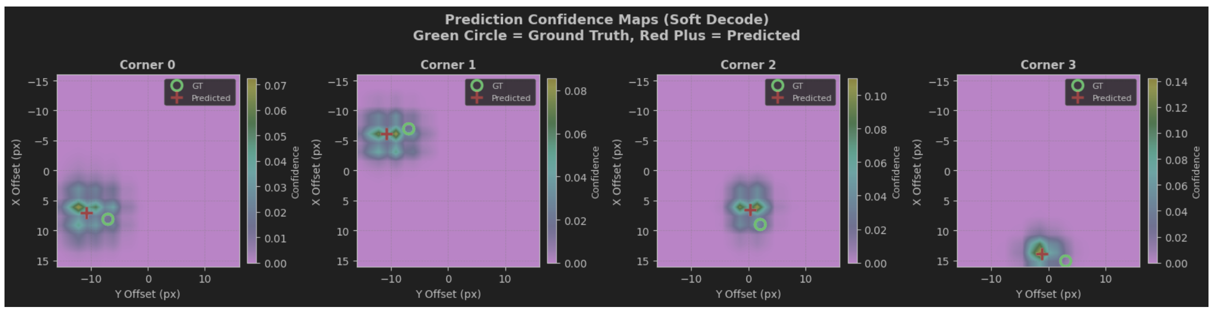

Homography Classification Visualization (Soft)
Green=Src, Cyan=GT, Red=Predicted
RMSE: 1.88px, MAE: 1.44px
Source
GT Warped
Predicted
0
1
2
3

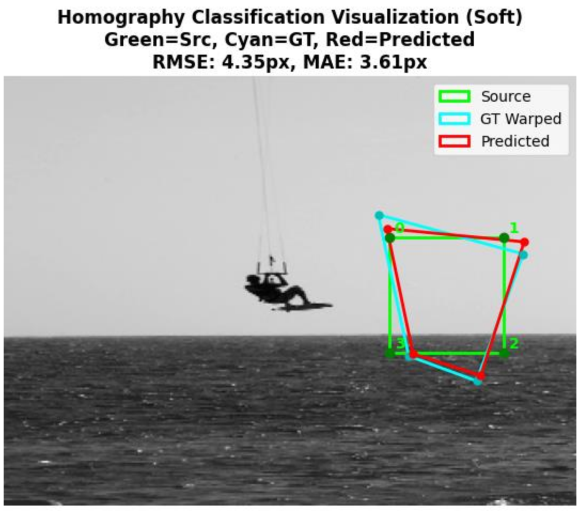




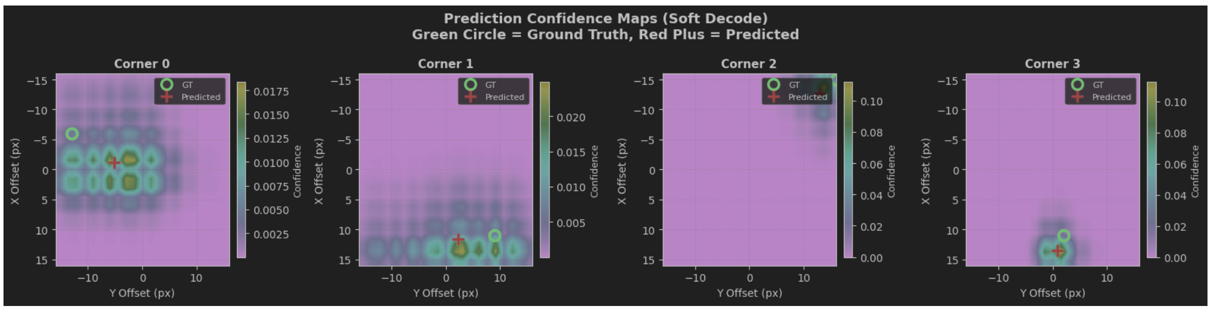

Homography Classification Visualization (Soft)
Green=Src, Cyan=GT, Red=Predicted
RMSE: 2.67px, MAE: 2.35px
Source
GT Warped
Predicted
0
1
2
3


Original Patch
Warped Patch (GT)
Warped Patch (Predicted)


Prediction Confidence Maps (Soft Decode)
Green Circle = Ground Truth, Red Plus = Predicted
Corner 0
Corner 1
Corner 2
Corner 3
GT
Predicted
X Offset (px)
Y Offset (px)
Confidence


Homography Classification Visualization (Soft)
Green=Src, Cyan=GT, Red=Predicted
RMSE: 4.35px, MAE: 3.61px
Source
GT Warped
Predicted
0
1
2
3

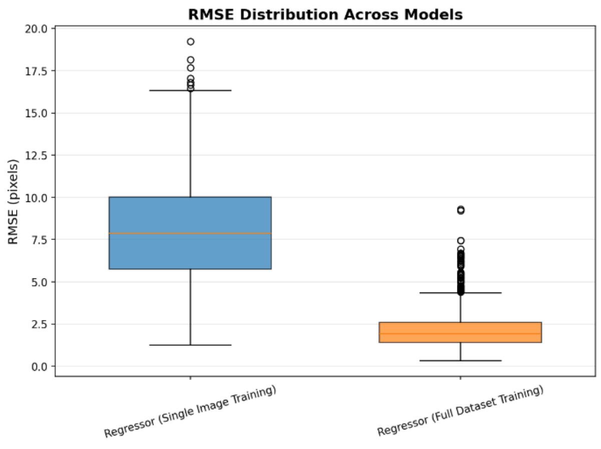
Original Patch
Warped Patch (GT)
Warped Patch (Predicted)

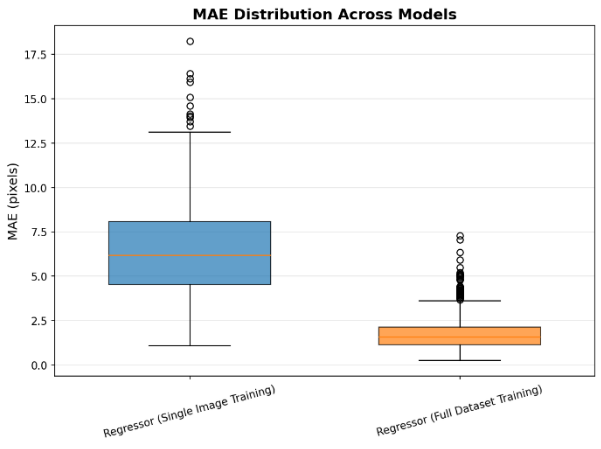

Prediction Confidence Maps (Soft Decode)
Green Circle = Ground Truth, Red Plus = Predicted
Corner 0
Corner 1
Corner 2
Corner 3
GT
Predicted
X Offset (px)
Y Offset (px)
Confidence

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")
```

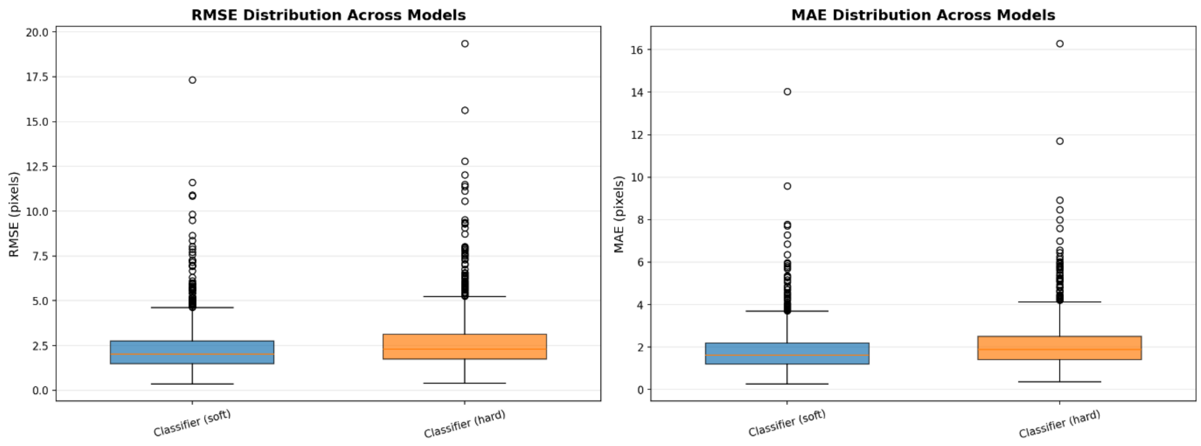




```
================================================================================
                               EVALUATION SUMMARY
================================================================================
          ocena z Regressor (Single Image Training) mean:  10.29  std:   6.52
          ocena z Regressor (Full Dataset Training) mean:   2.77  std:   2.11
================================================================================
```

Classification models

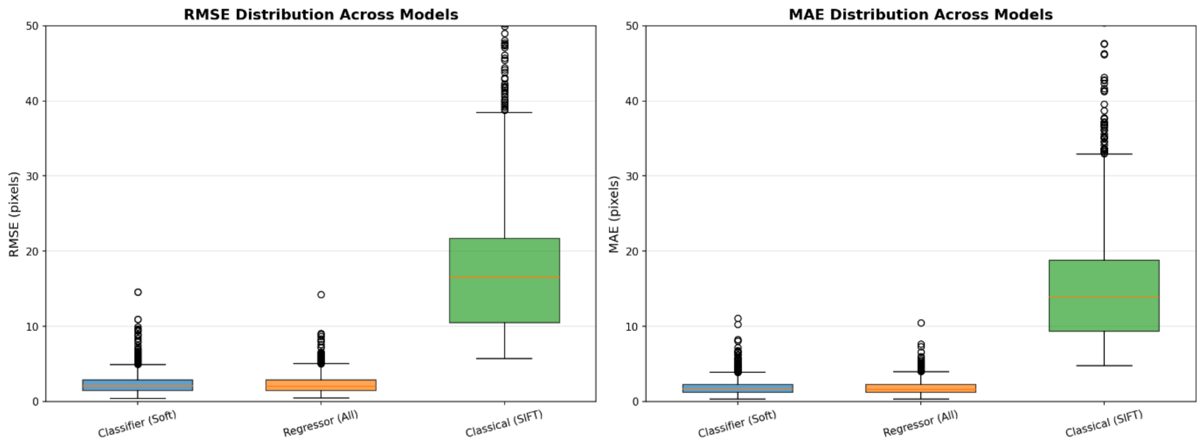
```python
# Compare classifiers (soft vs hard decoding)
classifier = HomographyClassifier()
classifier.load_state_dict(torch.load("checkpoints_homography_classify_all/h_classify_all.pth"))

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
================================================================================
```