porocilo

October 25, 2022

[14]:
```
%matplotlib inline

import cv2
import matplotlib.pyplot as plt
import numpy as np

plt.ion()
```

[14]:
```
<matplotlib.pyplot._IonContext at 0x7f39ecd3d990>
```

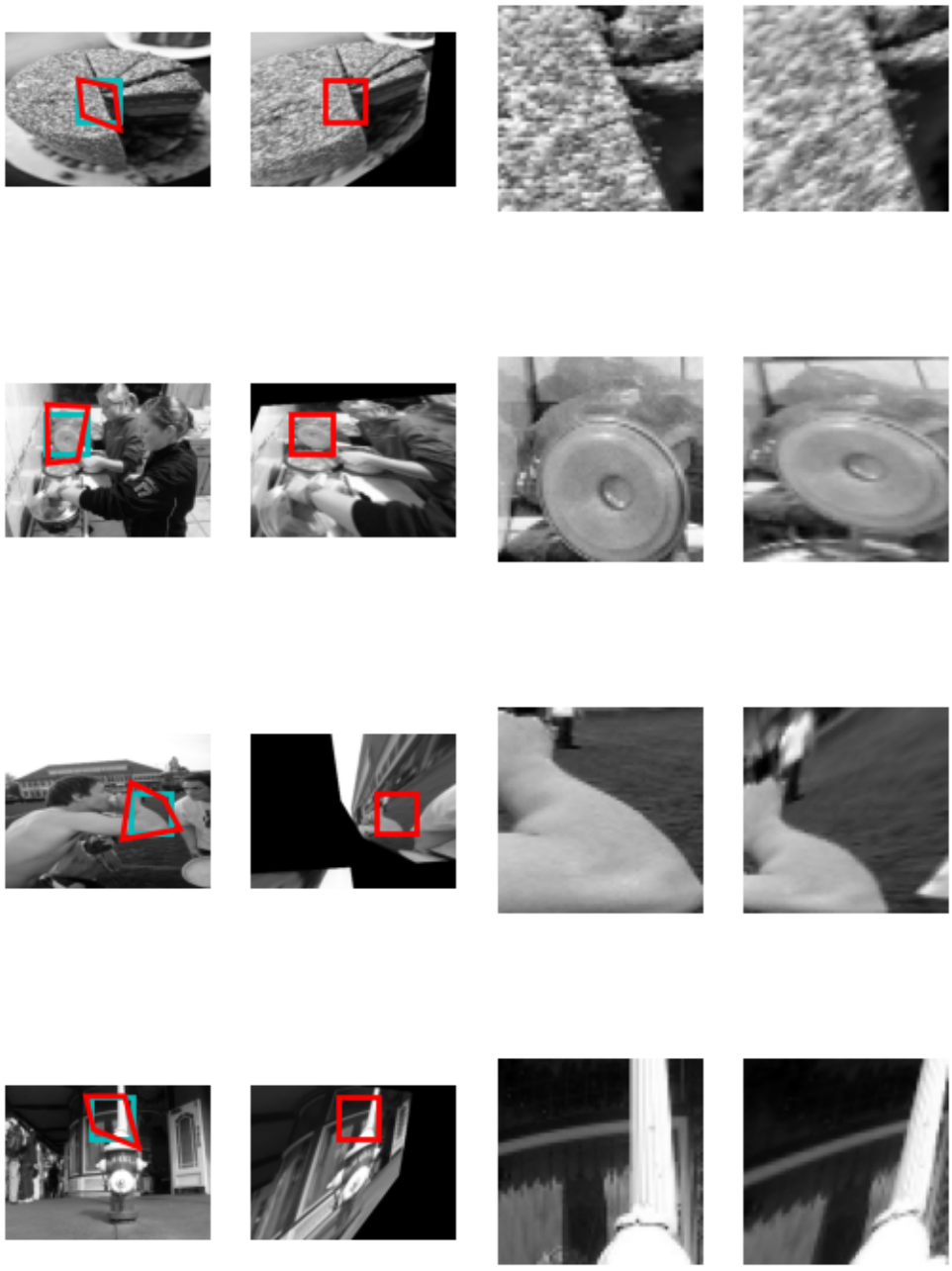
[15]:
```
from generator_podatkov import *
from oceni_homografijo import *
```

1 Zbirke slik

1.1 MS Coco 2014 dataset

[16]:
```
data_generator = generator_homography_pair(
    generator_repeater(
        generator_scale_image(
            generator_mscoco('../train2014/'),
            size=(240, 320),
        ),
        reps=1
    ),
    sample_size=(64, 64),
)


for n in range(6):
    pts, pts_aug, H_tr, slika, slika_warp, vzorec, vzorec_aug =␣
 ↪next(data_generator)

    fig, ax = plt.subplots(1, 4)
    ax[0].imshow(slika, cmap='gray')
    ax[0].add_patch(plt.Polygon(pts[:, ::-1], fill=False, ec='c', lw=2.0))
    ax[0].add_patch(plt.Polygon(pts_aug[:, ::-1], fill=False, ec='r', lw=2.0))
```

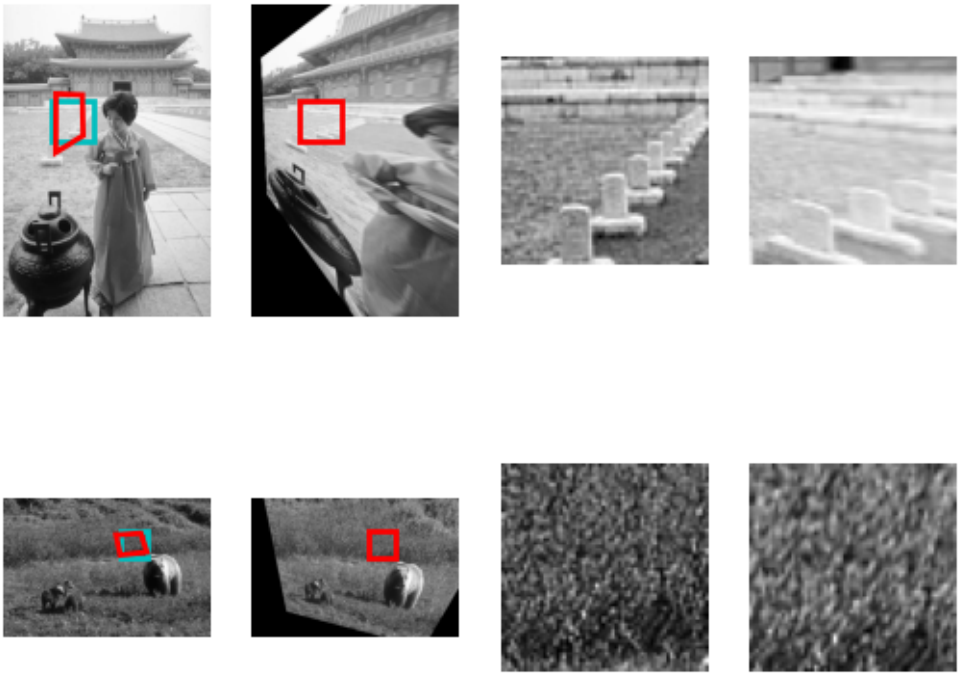
```
    ax[0].set_axis_off()

    ax[1].imshow(slika_warp, cmap='gray')
    ax[1].add_patch(plt.Polygon(pts[:, ::-1], fill=False, ec='r', lw=2.))
    ax[1].set_axis_off()

    ax[2].imshow(vzorec, cmap='gray')
    ax[2].set_axis_off()
    ax[3].imshow(vzorec_aug, cmap='gray')
    ax[3].set_axis_off()
```



1.2 Berkeley segmentation dataset

[17]:
```
data_generator = generator_homography_pair(
        generator_bsr('../BSR/'),
    sample_size=(64, 64),
)


for n in range(6):
    pts, pts_aug, H_tr, slika, slika_warp, vzorec, vzorec_aug =␣
 ↪next(data_generator)

    fig, ax = plt.subplots(1, 4)
    ax[0].imshow(slika, cmap='gray')
    ax[0].add_patch(plt.Polygon(pts[:, ::-1], fill=False, ec='c', lw=2.0))
    ax[0].add_patch(plt.Polygon(pts_aug[:, ::-1], fill=False, ec='r', lw=2.0))
    ax[0].set_axis_off()

    ax[1].imshow(slika_warp, cmap='gray')
    ax[1].add_patch(plt.Polygon(pts[:, ::-1], fill=False, ec='r', lw=2.))
    ax[1].set_axis_off()

    ax[2].imshow(vzorec, cmap='gray')
    ax[2].set_axis_off()
    ax[3].imshow(vzorec_aug, cmap='gray')
```

```
    ax[3].set_axis_off()
```



[18]:
```
import torch
import torch.nn

torch.no_grad().__enter__()

from mreza_klasifikacija import mreza_klasifikacija
from mreza_regresija import mreza_regressija
```

2 Nevronski mreži (preurejen izpis slojev iz torch)

2.1 Regresija

- TeloMreže
- Linear(in_features=512, out_features=8, …)

2.2 Klasifikacija

- TeloMreže()
- Linear(in_features=512, out_features=168, …)
- Softmax(dim=2)

2.3 TeloMreže

- ResnetBlock()
- ResnetBlock()

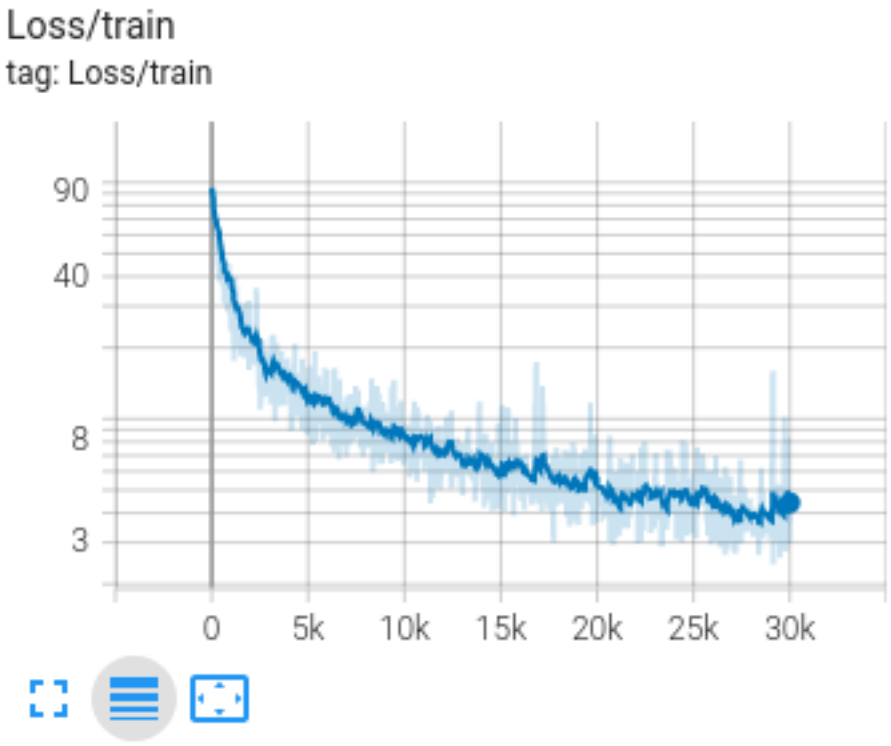
- MaxPool2d(kernel_size=2, stride=2, …)
- ResnetBlock()
- ResnetBlock()
- MaxPool2d(kernel_size=2, stride=2, …)
- ResnetBlock()
- ResnetBlock()
- MaxPool2d(kernel_size=2, stride=2, …)
- ResnetBlock()
- ResnetBlock()
- MaxPool2d(kernel_size=2, stride=2, …)
- Linear(in_features=8192, out_features=512, bias=True)
- Dropout(p=0.5, inplace=False)

2.4 ResnetBlock

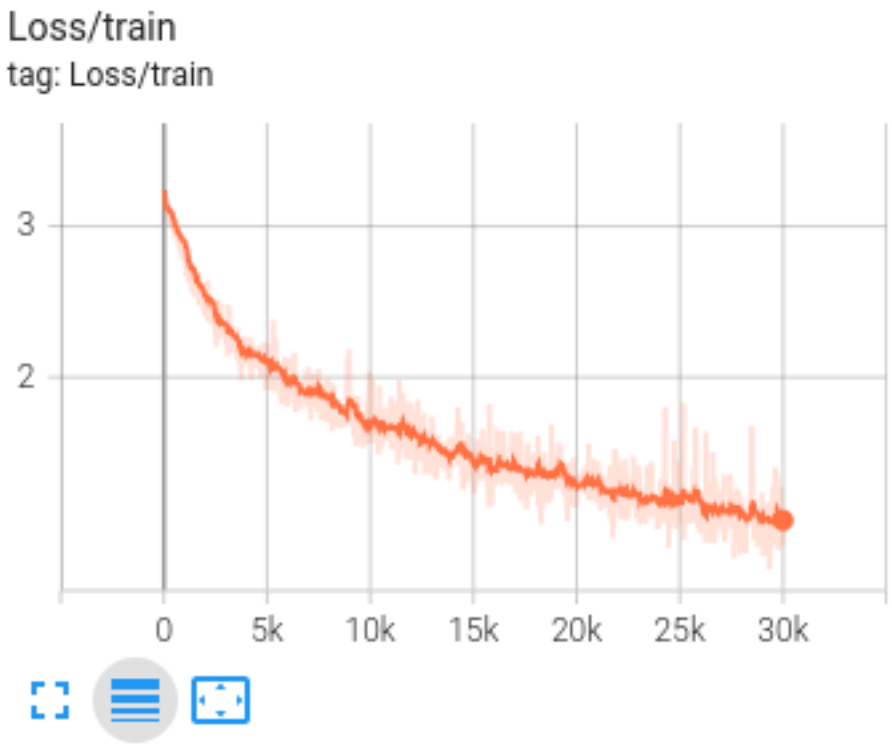
- Conv2d(…)
- BatchNorm2d(…)
- Dropout2d(…)
- ReLU(…)
- Conv2d(…)
- BatchNorm2d(…)
- Dropout2d(…)
- ReLU(…)
- Conv2d(…)

3 Učenje (30.000 iteracij po 64 vzorcev)

3.1 Izguba pri učenju regresije



3.2 Izguba pri učenju klasifikacije



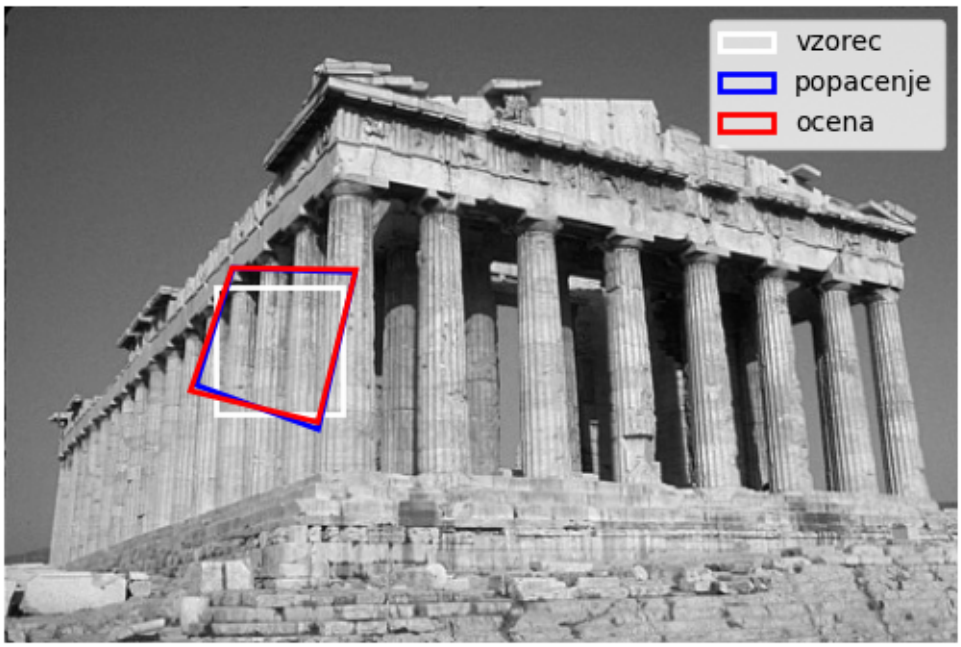
[19]:
```
mreza_klas = torch.load('./mreza_klasifikacija.pickle')
mreza_reg = torch.load('./mreza_regresija.pickle')

mreza_klas.eval()
mreza_reg.eval();
```

4 Rezultati regresije

[183]:
```
def f():
    pts, pts_aug, H_tr, slika, slika_warp, vzorec, vzorec_aug =␣
 ↪next(data_generator)
    data_in = np.array([vzorec, vzorec_aug])[np.newaxis, :, ::, ::]

    reg_out = mreza_reg(torch.from_numpy(data_in)).detach().numpy()
    pts_aug_est = pts+reg_out[0]

    fig, ax = plt.subplots(1, 1)
    ax.imshow(slika, cmap='gray')
    ax.add_patch(plt.Polygon(pts[:, ::-1], fill=False, ec='w', lw=2.0,␣
 ↪label='vzorec'))
```

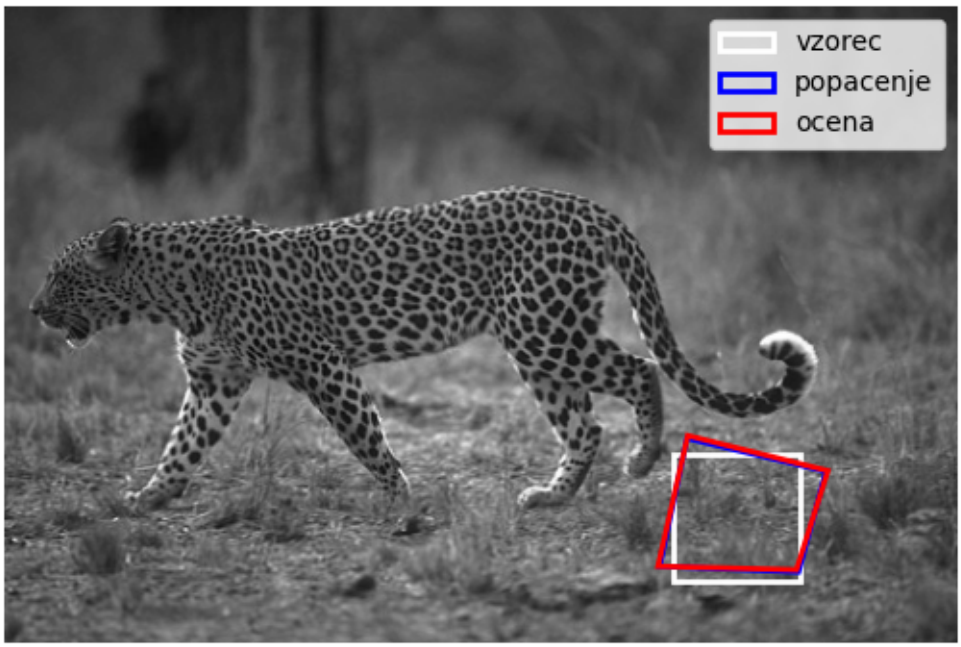
```
    ax.add_patch(plt.Polygon(pts_aug[:, ::-1], fill=False, ec='b', lw=2.0,␣
↪label='popacenje'))
    ax.add_patch(plt.Polygon(pts_aug_est[:, ::-1], fill=False, ec='r', lw=2.0,␣
↪label='ocena'))
    ax.set_axis_off()
    plt.legend()

    offset = reg_out[0]
    pts_vzorec = np.array([
        [0, 0],
        [0, 1.],
        [1., 1.],
        [1., 0],
    ])*vzorec.shape[0]

    H = offset_v_homografijo(pts_vzorec, -offset)
    vzorec_aug_est = cv2.warpPerspective(vzorec_aug, H, vzorec.shape[::-1])

    fig, ax = plt.subplots(1, 3, sharex=True, sharey=True)
    ax[0].imshow(vzorec, cmap='gray', vmin=0, vmax=1)
    ax[0].set_axis_off()
    ax[0].set_title('original')
    ax[1].imshow(vzorec_aug, cmap='gray', vmin=0, vmax=1)
    ax[1].set_axis_off()
    ax[1].set_title('popacen')
    ax[2].imshow(vzorec_aug_est, cmap='gray', vmin=0, vmax=1)
    ax[2].set_axis_off()
    ax[2].set_title('poravnan')
```

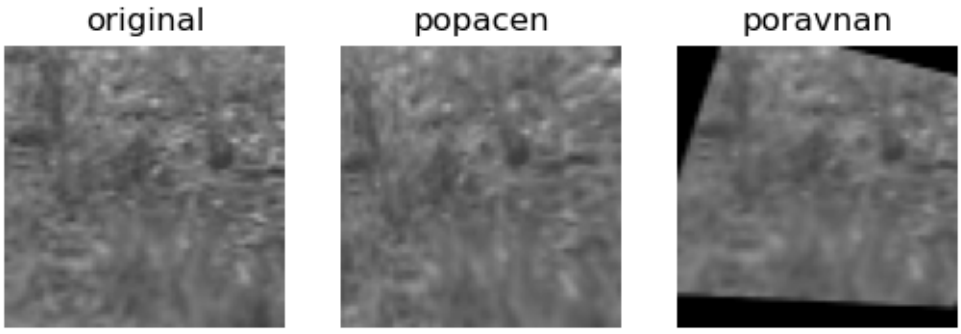
[21]:
```
f()
```



Glede na napovedano območje (rdeč pravokotnik) je poravnava precej dobra. Iz vzorcev spodaj pa še vedno ravidno, da poravnan vzorec nekoliko odstopa od originala.

[22]:
```
f()
```





Tudi scena/vzorec z nejasnimi robovi se poravna precej dobro.

[43]:
```
f()
```

Ta primer je kljub jasnim ostrimo robom v sliki poravnan slabše.

[169]:
```
f()
```

Tudi vzorec brez ostrih elementov je lahko zelo dobro poravnan. Najverjetneje na podlagi nizhkofrekvenčnega gradienta.

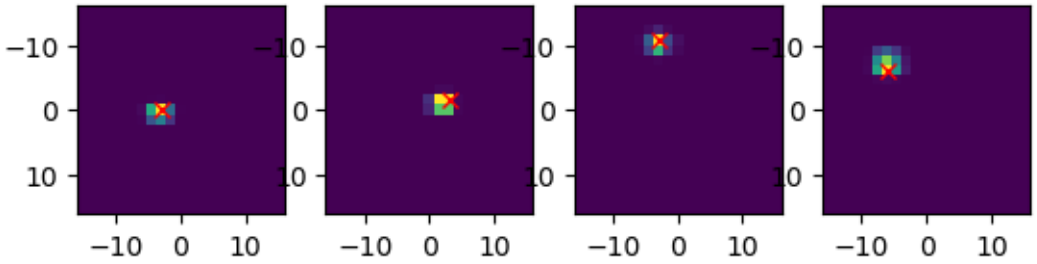
5 Rezultati klasifikacije

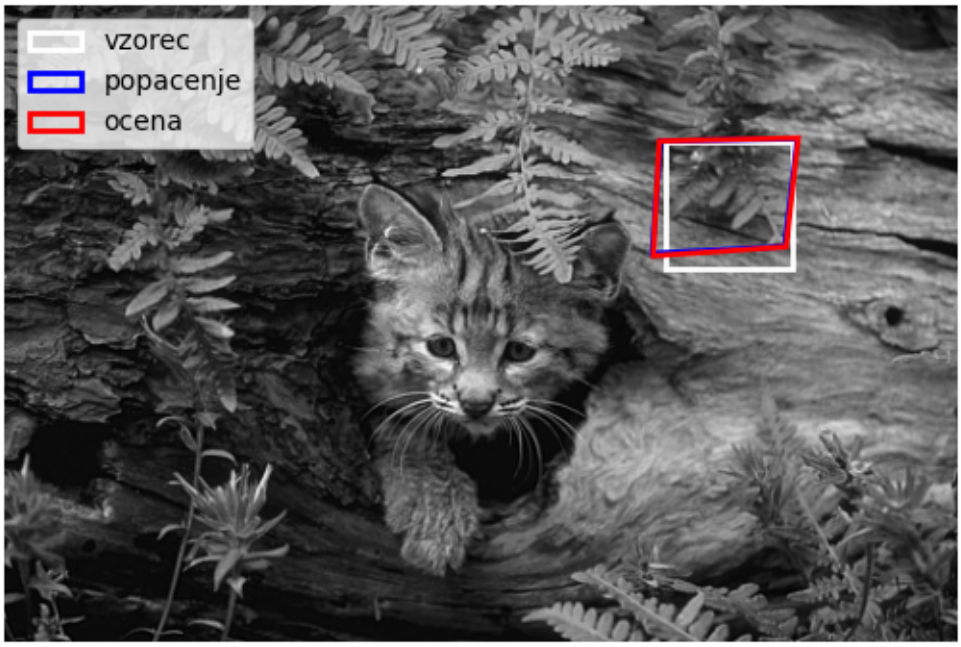
[188]:
```
def f():
    pts, pts_aug, H_tr, slika, slika_warp, vzorec, vzorec_aug =␣
 ↪next(data_generator)
    data_in = np.array([vzorec, vzorec_aug])[np.newaxis, :, ::, ::]
```

```
  klas_out = mreza_klas(torch.from_numpy(data_in))
  klas_out = torch.nn.Softmax(1)(klas_out)

  offset = classification_to_value(klas_out, -16, 16)[0]
  pts_aug_est = pts+offset

  pts_vzorec = np.array([
      [0, 0],
      [0, 1.],
      [1., 1.],
      [1., 0],
  ])*vzorec.shape[0]

  H = offset_v_homografijo(pts_vzorec, -offset)
  vzorec_aug_est = cv2.warpPerspective(vzorec_aug, H, vzorec.shape[::-1])


  pred_maps = (klas_out[:, :, ::2].reshape(-1, 21, 1, 4) * klas_out[:, :, 1::
↪2].reshape(-1, 1, 21, 4)).detach().numpy()
  n=0
  fig, ax = plt.subplots(1, 4)
  ax[0].imshow(pred_maps[n, :, :, 0], extent=(-16, 16, 16, -16))
  ax[0].plot(offset[0, 1], offset[0, 0], 'rx')
  ax[1].imshow(pred_maps[n, :, :, 1], extent=(-16, 16, 16, -16))
  ax[1].plot(offset[1, 1], offset[1, 0], 'rx')
  ax[2].imshow(pred_maps[n, :, :, 2], extent=(-16, 16, 16, -16))
  ax[2].plot(offset[2, 1], offset[2, 0], 'rx')
  ax[3].imshow(pred_maps[n, :, :, 3], extent=(-16, 16, 16, -16))
  ax[3].plot(offset[3, 1], offset[3, 0], 'rx')

  fig, ax = plt.subplots(1, 1)
  ax.imshow(slika, cmap='gray')
  ax.add_patch(plt.Polygon(pts[:, ::-1], fill=False, ec='w', lw=2.0,␣
↪label='vzorec'))
  ax.add_patch(plt.Polygon(pts_aug[:, ::-1], fill=False, ec='b', lw=2.0,␣
↪label='popacenje'))
  ax.add_patch(plt.Polygon(pts_aug_est[:, ::-1], fill=False, ec='r', lw=2.0,␣
↪label='ocena'))
  ax.set_axis_off()
  plt.legend()

  fig, ax = plt.subplots(1, 3, sharex=True, sharey=True)
  ax[0].imshow(vzorec, cmap='gray', vmin=0, vmax=1)
  ax[0].set_axis_off()
  ax[0].set_title('original')
  ax[1].imshow(vzorec_aug, cmap='gray', vmin=0, vmax=1)
  ax[1].set_axis_off()
```

```
    ax[1].set_title('popacen')
    ax[2].imshow(vzorec_aug_est, cmap='gray', vmin=0, vmax=1)
    ax[2].set_axis_off()
    ax[2].set_title('poravnan')
```

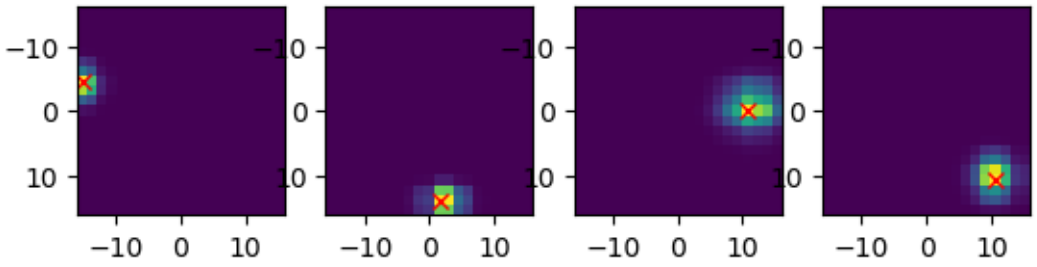
[171]:
```
f()
```

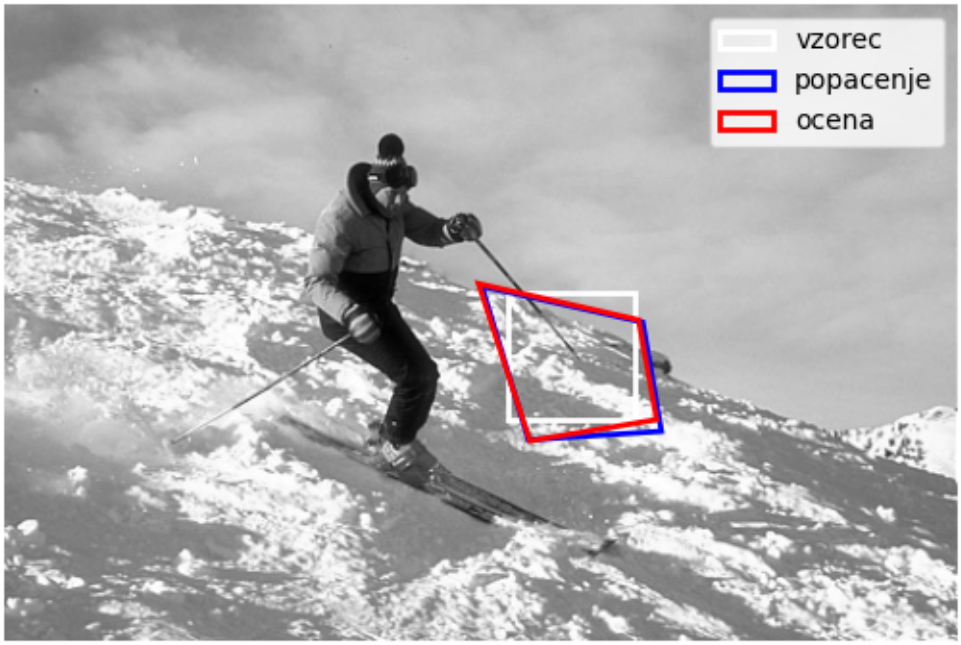




Če za klasifikacijo izrišemo napovedi, lahko vidimo kako prepričana je mreža v ocenjen rezultat. V gornjem primeru je zelo gotova v svojo napoved, napoved je tudi dokaj pravilna.

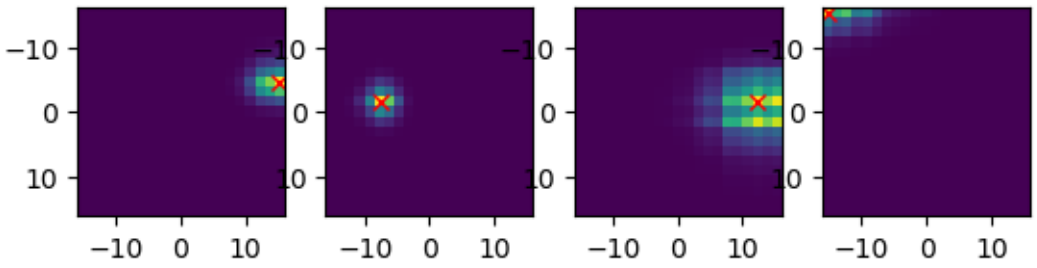
[198]:
```
f()
```

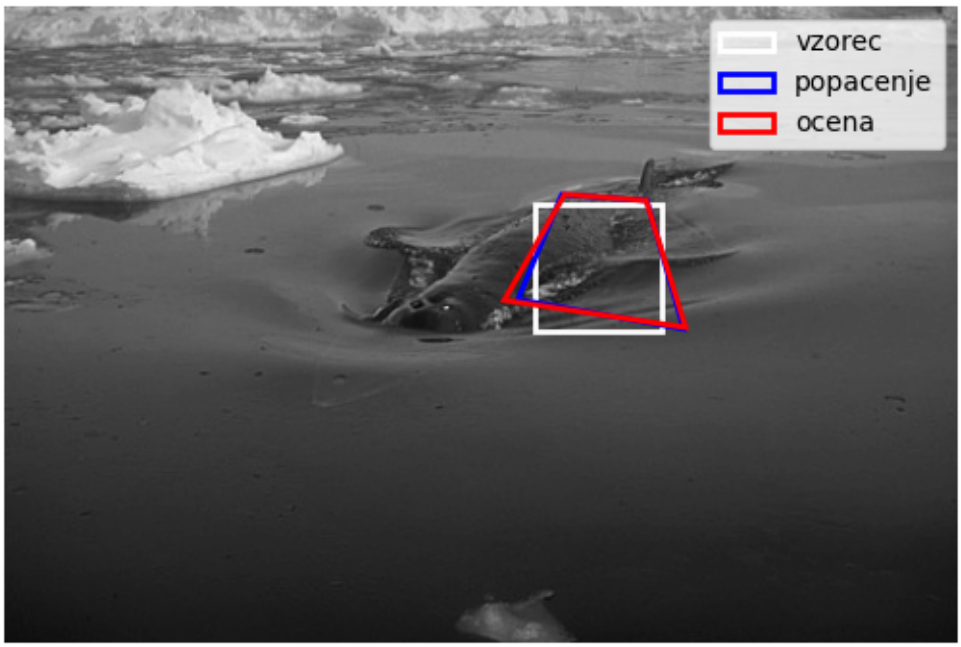


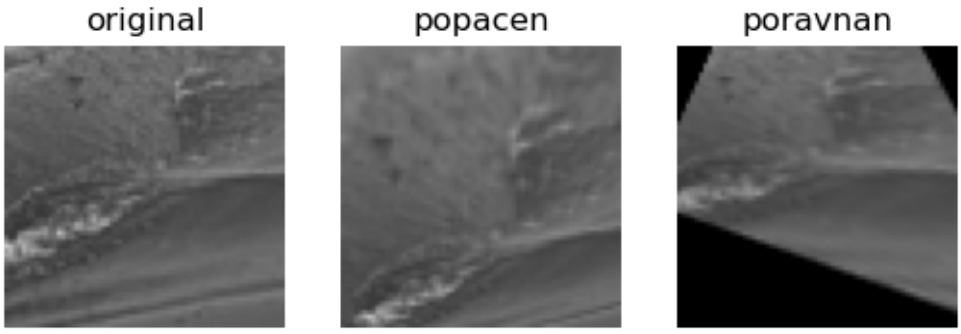


V tem primeru je mreža nekoliko negotova v napoved tretje točke, ta je tudi nekoliko nepravilna.

[200]:
```
f()
```







V tem primeru je mreža precej negotova v napoved tretje točke, vendar je ta dokaj pravilna.

6 Primerjava metod

[176]:
```
%run -i testiraj_mrezo.py
```

```
steps proccesed: 1054
steps proccesed: 2107
steps proccesed: 3010
steps proccesed: 4001
steps proccesed: 5082
steps proccesed: 6136
```

```
steps proccesed: 7216
```

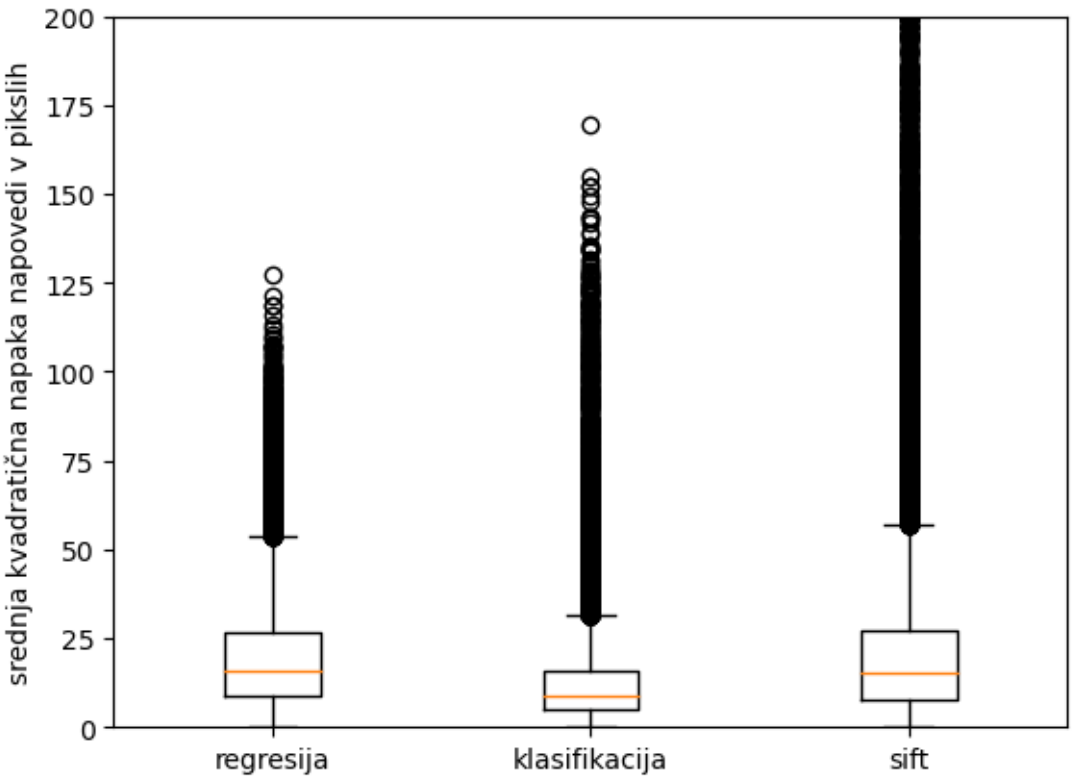
```
<Figure size 640x480 with 0 Axes>
```

[177]:
```
print('ocena z regresijsko glavo')
print(f'mean: {np.mean(reg_err_list):.2f} std: {np.std(reg_err_list):.2f}')
print('ocena z klasifikacijsko glavo')
print(f'mean: {np.mean(class_err_list):.2f} std: {np.std(class_err_list):.2f}')
print('ocena z sift detektorjem')
print(f'mean: {np.mean(sift_err_list):.2f} std: {np.std(sift_err_list):.2f}')
```

```
ocena z regresijsko glavo
mean: 20.15 std: 16.05
ocena z klasifikacijsko glavo
mean: 12.92 std: 13.91
ocena z sift detektorjem
mean: 56.39 std: 2612.40
```

[202]:
```
plt.boxplot([np.ravel(reg_err_list), np.ravel(class_err_list), np.
  ↪ravel(sift_err_list)], labels=['regresija', 'klasifikacija', 'sift'])
plt.ylabel('srednja kvadratična napaka napovedi v pikslih')
plt.ylim(0, 200)
```

[202]:
```
(0.0, 200.0)
```



[ ]: