Deep Learning - Homography

Homography

Any two images of the same planar surface can be related by a transformation called a **homography**. That is, a pixel coordinate in first image when multiplied by the homography is the same point on the plane projected into the second image.

For more information:

- https://en.wikipedia.org/wiki/Homography_(computer_vision)
- https://mattmaulion.medium.com/homography-transform-image-processing-eddbcb8e4ff7

We can encode a **homography** using 4 matching pairs of points.

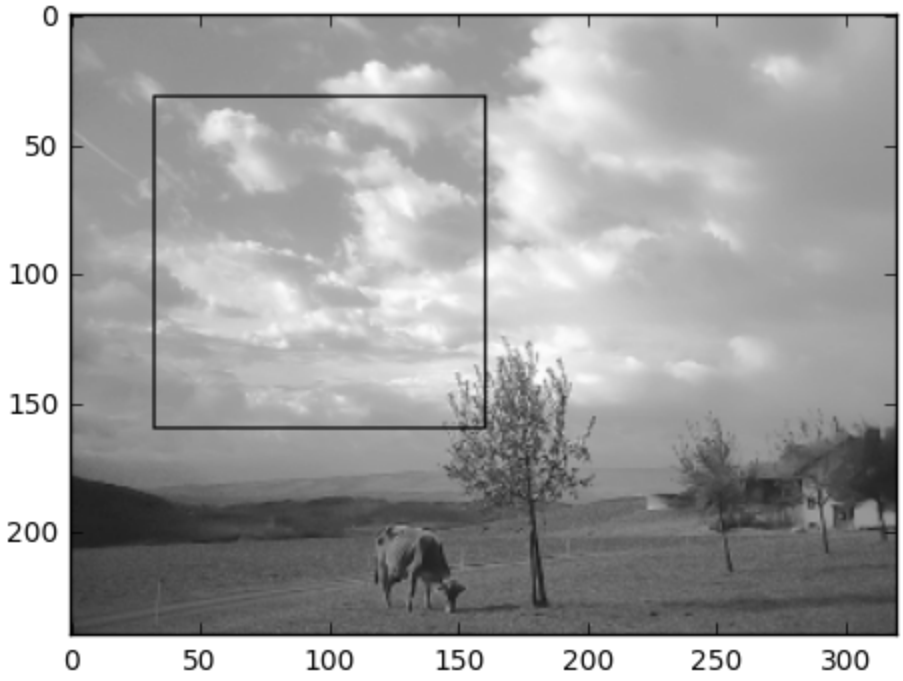
Task

Your task is to train a neural network to predict the **homography** between two crops of the same image.

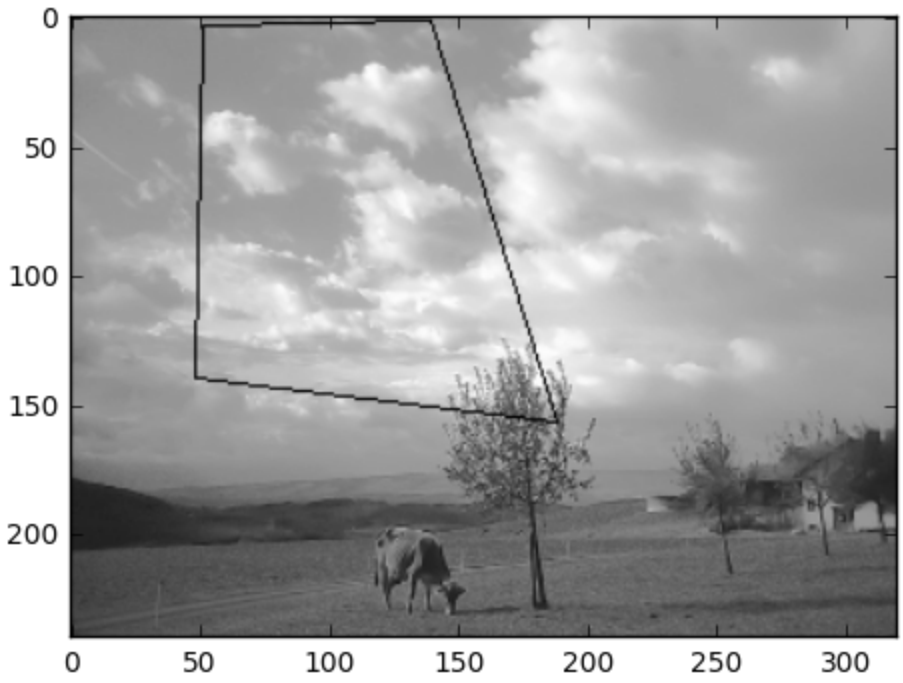
Dataset

The dataset was generated using the COCO dataset as follows:

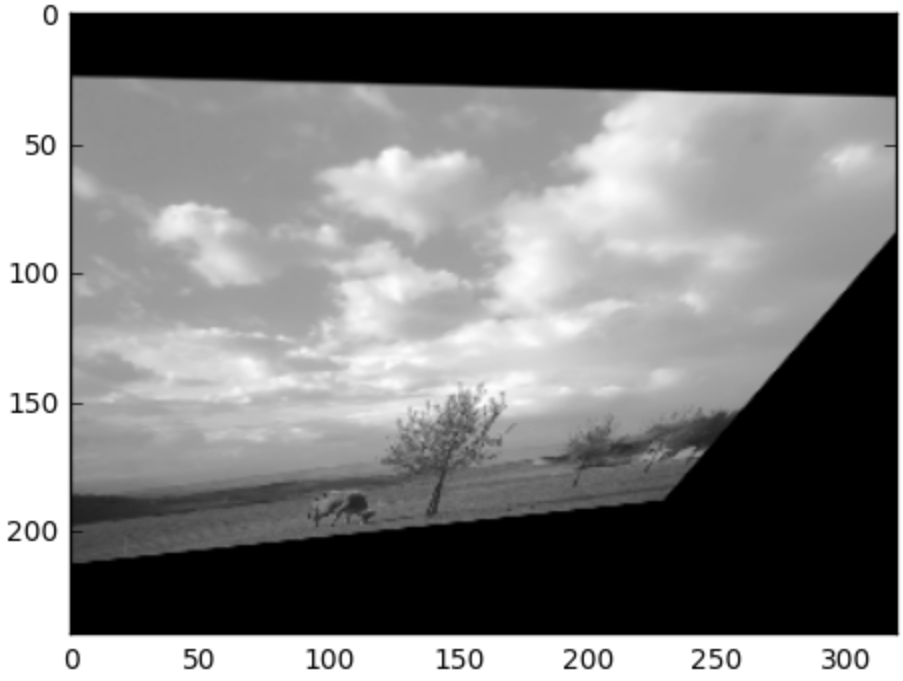
1. Read in image as grayscale and resize to (320, 240).
2. Define the four corners of a patch of size 128.



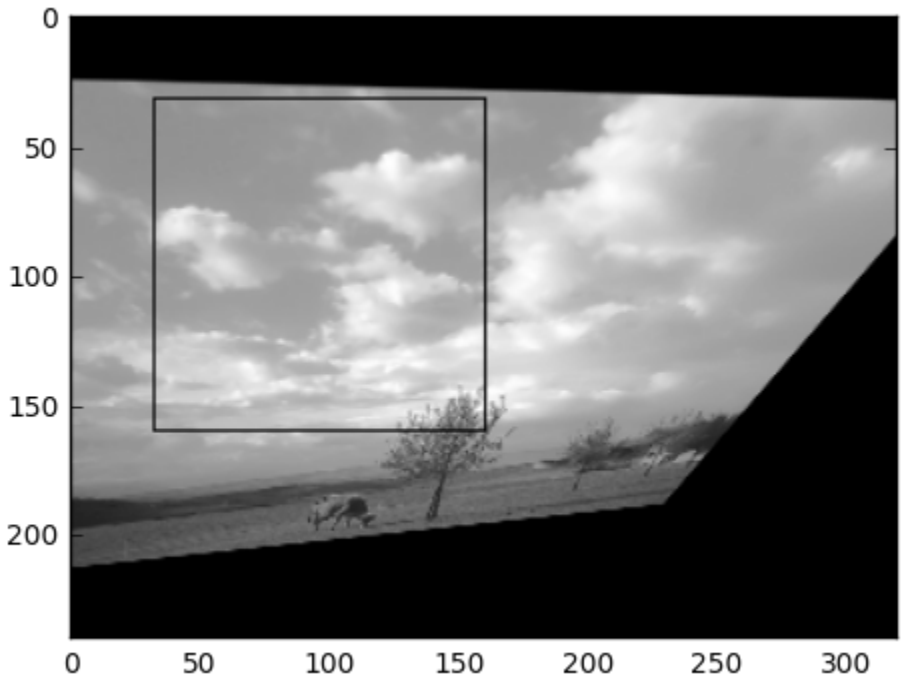
3. Randomly perturb the corners in the range (-32, 32).



4. Calculate the **homography** relating the original corners to the perturbed corners.
5. Use the inverse of this **homography** to warp the original image.



6. Take crops of the original and warped image using the non-perturbed corners.



7. Stack the crops together into an image with dimensions (2, 128, 128).
8. Calculate the corner offsets by subtracting the original corners from the perturbed corners.

The datasets can be accessed here:

Deep Learning Homography - Google Drive

https://drive.google.com/drive/folders/1ikm8MuB34-38xNS5v1dzZOBUJLXUV4ch?usp=sharing

The train and test datasets are provided in HDF5 files and can be opened using h5py (https://docs.h5py.org/en/stable/quick.html).

The images and labels are stored in the h5 file at the top level and are indexed from 0 using strings.

Here are some common functions:

```
import numpy as np
import h5py

# Open the file
h5_file = h5py.File('train.h5', 'r')

# Find the length of the dataset
dataset_length = len(h5_file['/'])

# Get the image and label at index i
data = h5_file[str(i)]
image = np.array(data) # image.shape = (2, 128, 128)
label = data.attrs.get('label').flatten() # label.shape = (8,)
```

There are 118287 samples in the training set.

There are 5000 samples in the testing set.

Network Structure

The choice of network structure is up to you. We provide a homography in the form of a vector of 8 values (corner offsets):progr

```
[tl.x, tl.y, tr.x, tr.y, br.x, br.y, bl.x, bl.y]

where:
tl = top left corner offset
tr = top right corner offset
br = bottom right corner offset
bl = bottom left corner offset
```

Therefore your network could have the following input → output dimensions:

```
(B, 2, 128, 128) → (B, 8)
```

```
where B is batch size.
```

Evaluation

The metric used to assess homography estimation networks is Mean Average Corner Error (MACE).

"To measure this metric, one first computes the L2 distance between the ground truth corner position and the estimated corner position. The error is averaged over the four corners of the image, and the mean is computed over the entire test set."

So we can compare your answers, a PyTorch implementation of MACE is provided:

```
y_pred # torch.Tensor (B, 8) output of network
y # torch.Tensor (B, 8) labels

diff = y.subtract(y_pred) # (B, 8) error per corner dimension

diff = diff.pow(2) # (B, 8) squared error per corner dimension

diff = diff.reshape((-1, 4, 2)) # (B, 4, 2) squared error by corner

diff = diff.sum(-1) # (B, 4) sum the squared error per corner

diff = diff.sqrt() # (B, 4) L2 distance per corner

diff = diff.mean(-1) # (B,) mean of L2 distance per sample

mace = diff.mean(-1) # (1,) mean over all of test dataset
```

where B is the number of elements in the test set. You can also split this up over batches and do the final mean on a concatenated vector of the diffs computed per batch.

Submission

Please return to us:

1. Your source code.

2. A short report with a brief overview of what you did and why, anything you learnt or found interesting and your final MACE value on the test set (if working implementation).

You are free to use whatever network architecture, loss function, optimiser, augmentation etc. as you wish! We would like to see how you approach a deep learning project from the ground up. You will get bonus points for:

- Clean code and structure.
- Custom implementations.
- An interesting report.

Please do not spend more time than you are comfortable with! We are more interested in your decision making and reasoning than a low MACE value. It should take you less than 10 hours to get a decent submission (including training).

We recommend using Colab to train the model if you do not have a powerful GPU at home: https://colab.research.google.com/

Please let us know if you run out of free GPU time.

This is a new assignment so things could very well be incomplete or not explained well! If you have any questions please email us at sams@bolt6.ai