

Deep Learning

Exercise 7: Transfer Learning

Instructor: Manuel Günther

Email: guenther@ifi.uzh.ch

Office: AND 2.54

Friday, April 23, 2020

Outline

- 1 Pre-trained Networks
- 2 Face Recognition

Outline

- 1 Pre-trained Networks
 - Obtaining Pre-Trained Networks
 - Turn Networks to Deep Feature Extractors

Obtaining Pre-Trained Networks

Pre-Defined Networks

- Networks implemented in `torchvision.models`
→ See <http://pytorch.org/vision/stable/models.html>
- Create pre-defined network, e.g.:
`network = torchvision.models.resnet18()`
- Download pre-trained (on ImageNet) network:
`network = torchvision.models.resnet18(pretrained=True)`

Input Normalization

- RGB color images in resolution $3 \times 244 \times 244$ with pixels $[0, 1]$
- Normalized by subtracting mean $(0.485, 0.456, 0.406)$ and dividing by variance $(0.229, 0.224, 0.225)$



Turn Networks to Deep Feature Extractors

Problem

- Pre-trained networks return logits
- We only get object of ResNet class
- Need to overwrite `forward`

Redefining `_forward_impl`

- Rewrite function:

```
def my_forward(self, x):
```

- Bind `my_forward` to object:

```
net = torchvision.models.resnet18(  
    pretrained=True)  
net._forward_impl = types.MethodType(  
    my_forward, net)
```

ResNet (Implementation)

```
def _forward_impl(self, x):  
    x = self.conv1(x)  
    x = self.bn1(x)  
    x = self.relu(x)  
    x = self.maxpool(x)  
  
    x = self.layer1(x)  
    x = self.layer2(x)  
    x = self.layer3(x)  
    x = self.layer4(x)  
  
    x = self.avgpool(x)  
    x = torch.flatten(x, 1)  
    ## x = self.fc(x)  
  
    return x  
  
def forward(self, x):  
    return self._forward_impl(x)
```

Outline

- 2 Face Recognition
 - Dataset
 - Face Identification

Dataset

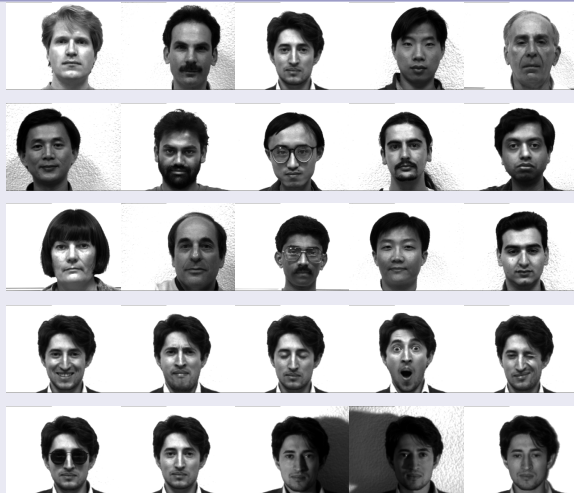
Face Recognition Dataset

- Images of 15 people
- Grayscale images as .gif
- File: `subjectXX.situation`
- 11 different variations:
happy, sad, sleepy, surprised, wink
glasses, noglasses, left, right, center
- Load: `PIL.Image.open(...)`

Download Face Data

<http://vision.ucsd.edu/content/yale-face-database>

Images

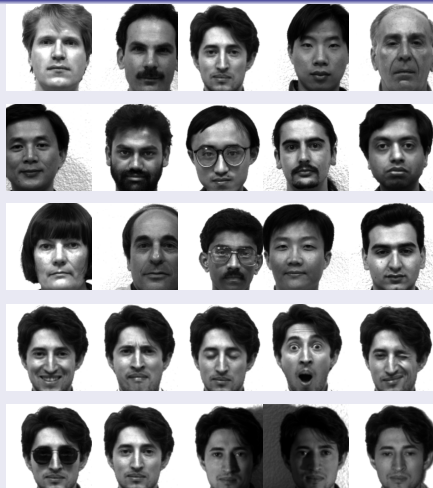


Dataset

Image Preprocessing

- 1 Scale image to 300 pixel height:
`torchvision.transforms.Resize(300)`
- 2 Take 224×224 center crop:
`torchvision.transforms.CenterCrop(224)`
- 3 Convert to Tensor:
`torchvision.transforms.ToTensor()`
- 4 Convert to 3-plane image:
`tensor.repeat(3,1,1)`
- 5 Normalize (see above)
- 6 Add batchsize dimension:
`tensor.unsqueeze(0)`

Images



Face Identification

Gallery

- Extract features for **normal** images
- Associate features with subject
 ⇒ 15 features in gallery

Exemplary Results

normal	happy	sad	sleepy
15/15	14/15	15/15	15/15
surprised	wink	glasses	noglasses
13/15	15/15	15/15	13/15
leftlight	rightlight	centerlight	total
12/15	12/15	11/14	150/164

Probing

- Go through all variations
- Go through all subjects
- Extract feature for image
- Compare to all gallery features
 → Compute gallery subject with lowest distance to probe
- Compare best gallery subject with probe subject
 → Same subject: correctly identified
- Count correct identifications

Face Identification

Task 1: Pre-Trained Network

- Download pre-trained network via `torchvision`
→ Try out versions (18, 32, ...)
- Turn network into deep feature extractor

Task 2: Feature Extraction

- Write a function that:
 - 1 Loads image via `PIL`
 - 2 Crops the face
 - 3 Transforms image to `tensor`
 - 4 Extracts the deep feature
 - 5 Returns `flattened` version as a `numpy` array

Face Identification

Task 3: Build Gallery

- 1 Store features for `normal` images

Task 4: Probing

- 1 Take all images of all subjects
 - Split by variation
- 2 Extract features for image
- 3 Compare to all gallery features
 - `scipy.spatial.distance.euclidean`
- 4 Count correctly classified faces
 - Split by variation
 - Compute total

Task 5: Variations (Optional)

- 1 Different network topologies
- 2 Other distance functions
- 3 Different preprocessing
 - Face detection?
- 4 Raw pixel values of cropped faces as features
- 5 Different variation in gallery
 - `happy`, `glasses`, `rightlight`