Future Vision

1950

FUTURE VISION

Computer Vision

2017 MWF 1PM 368
FaceApp

- Learning-based face transformations
FaceApp apologizes for building a racist AI

If only all algorithmic bias were as easy to spot as this: FaceApp, a photo-editing app that uses a neural network for editing selfies in a photorealistic way, has apologized for building a racist algorithm.

The app lets users upload a selfie or a photo of a face, and offers a series of filters that can then be applied to the image to subtly or radically alter its appearance — its appearance-shifting effects include aging and even changing gender.

The problem is the app also included a so-called “hotness” filter, and this filter was racist. As users pointed out, the filter was lightening skin tones to achieve its mooted “beautifying” effect. You can see the filter pictured above in a before and after shot of President Obama.

In an emailed statement apologizing for the racist algorithm, FaceApp’s founder and CEO Yaroslav Goncharov told us: “We are deeply sorry for this unquestionably serious issue. It is an unfortunate side-effect of the underlying neural network caused by the training set bias, not intended behaviour. To mitigate the issue, we have renamed the effect to exclude any positive connotation associated with it. We are also working on the complete fix that should arrive soon.”
Project 5 mark distribution
Project 5 avg. prec. distribution
Project 5 – well done!

“Finally, there will be extra credit and recognition for the students who achieve the highest average precision.”

Drumroll please...

*. 100% - Multi-scale squares at every position
1. 93.9% - Kyle Myerson
2. 93.7% - Tiffany Chen
3. 92.7% - Qikun (Tim) Guo
4. 92.4% - Lucas Lehnert
5. 92.1% - Katya Schwieggershausen
Results on our photos
Results on our photos

[Kyle Myerson]
Profile?
Profile?

image: "spring2017_profile.jpg" green=detection

[Kyle Myerson]
Are there any faces here?
No faces...
More TAs!

• 70 in class now (+ 70 on waitlist)
• 107 pre-registered for fall 2017
Reading architecture diagrams

Layers
- Kernel sizes
- Strides
- # channels
- # kernels
- Max pooling
Convolutions: More detail

32x32x3 image

- Height: 32
- Width: 32
- Depth: 3
Convolutions: More detail

32x32x3 image

5x5x3 filter
Convolutions: More detail

Convolution Layer

- 32x32x3 image
- 5x5x3 filter
- convolve (slide) over all spatial locations

activation map

Andrej Karpathy
Convolutions: More detail

For example, if we had 6 5x5 filters, we’ll get 6 separate activation maps:

We stack these up to get a “new image” of size 28x28x6!
Convolutions: More detail

- **CONV, ReLU**
  - e.g. 6
  - 5x5x3 filters

- **CONV, ReLU**
  - e.g. 10
  - 5x5x6 filters

- **CONV, ReLU**
Convolutions: More detail

Output size:
\[(N - F) / \text{stride} + 1\]
Our connectomics diagram

Auto-generated from network declaration by *nolearn* (for Lasagne / Theano)

Input
75x75x4

Conv 1
3x3x4
64 filters
Max pooling
2x2 per filter

Conv 2
3x3x64
48 filters
Max pooling
2x2 per filter

Conv 3
3x3x48
48 filters
Max pooling
2x2 per filter

Conv 4
3x3x48
48 filters
Max pooling
2x2 per filter
AlexNet diagram (simplified)

Input size
227 x 227 x 3

Conv 1
11 x 11 x 3
Stride 4
96 filters

Conv 2
5 x 5 x 48
Stride 1
256 filters

Conv 3
3 x 3 x 256
Stride 1
384 filters

Conv 4
3 x 3 x 192
Stride 1
384 filters

Conv 4
3 x 3 x 192
Stride 1
256 filters

Krizhevsky et al. 2012
Beyond AlexNet

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan & Andrew Zisserman 2015

These are the pre-trained “VGG” networks that you use in Project 6
### ConvNet Configuration

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>11 weight layers</td>
<td>11 weight layers</td>
<td>13 weight layers</td>
<td>16 weight layers</td>
<td>16 weight layers</td>
<td>19 weight layers</td>
</tr>
<tr>
<td>input (224 × 224 RGB image)</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
<td>conv3-64</td>
</tr>
<tr>
<td></td>
<td>LRN</td>
<td>conv3-64</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
<tr>
<td>maxpool</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
<td>conv3-128</td>
</tr>
<tr>
<td>maxpool</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td>maxpool</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
<td>conv3-256</td>
</tr>
<tr>
<td>maxpool</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td>maxpool</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td>maxpool</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
<td>conv3-512</td>
</tr>
<tr>
<td>maxpool</td>
<td>FC-4096</td>
<td>FC-4096</td>
<td>FC-1000</td>
<td>soft-max</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 2: Number of parameters (in millions).

<table>
<thead>
<tr>
<th>Network</th>
<th>A, A-LRN</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parameters</td>
<td>133</td>
<td>133</td>
<td>134</td>
<td>138</td>
<td>144</td>
</tr>
</tbody>
</table>
Table 4: ConvNet performance at multiple test scales.

<table>
<thead>
<tr>
<th>ConvNet config. (Table 1)</th>
<th>smallest image side</th>
<th>top-1 val. error (%)</th>
<th>top-5 val. error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>train ($S$)</td>
<td>test ($Q$)</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>256</td>
<td>224,256,288</td>
<td>28.2</td>
</tr>
<tr>
<td>C</td>
<td>256</td>
<td>224,256,288</td>
<td>27.7</td>
</tr>
<tr>
<td></td>
<td>384</td>
<td>352,384,416</td>
<td>27.8</td>
</tr>
<tr>
<td>[256; 512]</td>
<td>256,384,512</td>
<td>26.3</td>
<td>8.2</td>
</tr>
<tr>
<td>D</td>
<td>256</td>
<td>224,256,288</td>
<td>26.6</td>
</tr>
<tr>
<td></td>
<td>384</td>
<td>352,384,416</td>
<td>26.5</td>
</tr>
<tr>
<td>[256; 512]</td>
<td>256,384,512</td>
<td><strong>24.8</strong></td>
<td><strong>7.5</strong></td>
</tr>
<tr>
<td>E</td>
<td>256</td>
<td>224,256,288</td>
<td>26.9</td>
</tr>
<tr>
<td></td>
<td>384</td>
<td>352,384,416</td>
<td>26.7</td>
</tr>
<tr>
<td>[256; 512]</td>
<td>256,384,512</td>
<td><strong>24.8</strong></td>
<td><strong>7.5</strong></td>
</tr>
</tbody>
</table>
Google LeNet (2014)

- Conv (op#0)
- Relu (op#1)
- MaxPool (op#2)
- LRN (op#3)
- Conv (op#4)
- Conv (op#5)
- Conv (op#6)

22 layers
6.67% error
ImageNet top 5
Inception!

Another view of GoogLeNet’s architecture.
Parallel layers

Full Inception module
ResNet (He et al., 2015)

ResNet won ILSVRC 2015 with a top-5 error rate of 3.6%

Depending on their skill and expertise, humans generally hover around a 5-10% error.

But the task is arguably not well defined.
Revolution of Depth

ImageNet Classification top-5 error (%)

Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)

VGG, 19 layers (ILSVRC 2014)

ResNet, 152 layers (ILSVRC 2015)

CIFAR-10

• 60,000 32x32 color images, 10 classes

Here are the classes in the dataset, as well as 10 random images from each:

airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
Simply stacking layers?

- **Plain** nets: stacking 3x3 conv layers...
- 56-layer net has **higher training error** and test error than 20-layer net

Simply stacking layers?

- “Overly deep” plain nets have higher training error
- A general phenomenon, observed in many datasets

a shallower model (18 layers)

a deeper counterpart (34 layers)

- Richer solution space
- A deeper model should not have higher training error

Regular net

\[ H(x) \] is any desired mapping, hope the 2 weight layers fit \( H(x) \)
Residual Unit

A residual block
Residual Unit

The inputs of a lower layer is made available to a node in a higher layer.
Network “Design”

CIFAR-10 experiments

- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error