

SIGFLAG

Writeup: FaustCTF 2018: Jodlgang

Student: Markus Vogl

Team: [SIGFLAG.at](https://sigflag.at)

Uni: [JKU Linz](https://www.jku.at)

Institute: [Institute for Network and Security](#)

Sponsor: [VACE](#)

SIGFLAG

Overview

- 01 Application Overview
- 02 Problem analysis
- 03 Machine Learning: VGG16
- 04 The stupid solution
- 05 The smart solution

1. Application Overview

Name	Size	Type	Date Modified	Permissions
data	5 items	Folder	2018-06-02 01:38:21	drwxrwxrwx
jodlenv	6 items	Folder	2018-06-01 16:20:59	drwxrwxrwx
bin	19 items	Folder	2018-06-01 16:20:59	drwxrwxrwx
include	0 items	Folder	2018-06-01 02:29:40	drwxrwxrwx
lib	1 item	Folder	2018-06-01 02:29:40	drwxrwxrwx
lib64	1 item	Link to Folder	2018-06-01 02:29:40	lrwxrwxrwx
pyvenv.cfg	69 bytes	Text	2018-06-01 02:29:40	-rw-rw-r--
share	2 items	Folder	2018-06-01 02:30:01	drwxrwxrwx
jodlgang	9 items	Folder	2018-06-02 01:38:21	drwxrwxrwx
cnn_weights.h5	507.2 MB	Document	2018-06-01 00:43:38	-rw-r--r--
db.sqlite3	172.0 kB	Unknown	2018-06-01 02:30:06	-rw-rw-r--
jodlgang	6 items	Folder	2018-06-01 02:30:06	drwxrwxrwx
__init__.py	0 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
__pycache__	5 items	Folder	2018-06-01 17:38:26	drwxrwxrwx
settings.py	4.8 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
static	3 items	Folder	2018-06-01 02:30:06	drwxrwxrwx
urls.py	799 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
wsgi.py	393 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
jodlplatform	15 items	Folder	2018-06-01 17:33:10	drwxrwxrwx
admin.py	2.9 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
apps.py	99 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
backends.py	2.0 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
context_processors.py	382 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
forms.py	4.2 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
__init__.py	28 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
managers.py	304 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
migrations	5 items	Folder	2018-06-01 02:30:06	drwxrwxrwx
models.py	1.5 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
__pycache__	9 items	Folder	2018-06-01 17:34:27	drwxrwxrwx
test.py	70 bytes	Text	2018-06-01 17:31:55	-rw-r--r--
tests.py	60 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
urls.py	712 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
utils.py	687 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
views.py	1.4 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
manage.py	540 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
runme.py	1.1 kB	Text	2018-06-01 21:33:03	-rwxr-xr-x
static	2 items	Folder	2018-06-01 00:43:38	drwxrwxrwx
templates	9 items	Folder	2018-06-01 00:43:38	drwxrwxrwx
css	5 items	Folder	2018-06-01 00:43:38	drwxrwxrwx
bootstrap.css	172.8 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
bootstrap.css.map	425.2 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
bootstrap.min.css	140.4 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
bootstrap.min.css.map	557.9 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
login.css	593 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
js	4 items	Folder	2018-06-01 00:43:38	drwxrwxrwx
bootstrap.bundle.js	210.6 kB	Program	2018-06-01 00:43:38	-rw-rw-r--
bootstrap.bundle.js.map	359.5 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
bootstrap.bundle.min.js	70.8 kB	Program	2018-06-01 00:43:38	-rw-rw-r--
bootstrap.bundle.min.js.map	293.7 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
templates	9 items	Folder	2018-06-01 00:43:38	drwxrwxrwx
about.html	1.4 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
added_note.html	525 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
add_note.html	1.4 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
base.html	1.2 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
home.html	1.7 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
index.html	563 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
logged_out.html	139 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
login.html	1.7 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
raw_base.html	603 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
tensorflow	7 items	Folder	2018-06-01 17:34:27	drwxrwxrwx
functions.py	1.1 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
im2col.py	2.2 kB	Text	2018-06-01 17:51:02	-rw-rw-r--
__init__.py	0 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
initializer.py	773 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
layers.py	6.3 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
model.py	6.0 kB	Text	2018-06-01 00:43:38	-rw-rw-r--
__pycache__	6 items	Folder	2018-06-01 17:52:56	drwxr-xr-x
jodlgang.nginx.conf	641 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--
uwsgi_params	664 bytes	Text	2018-06-01 00:43:38	-rw-rw-r--

1. Application Overview

- UWSGI: Web Server Gateway Interface
- Django Webapp on port 8000
 - Bootstrap CSS + JS + html templates
- Virtual python env + separate user
 - No easy cross service attacks
- SQLite DB
- Tensorflow
 - Numpy implementation of Tensorflow
 - 500MB .h5 pretrained model

1. Application Overview: Database

db.sqlite3 - Sqliteman

File Context Database System Help

Schema Pragmas

Database

- main
 - Tables (9)
 - auth_group
 - auth_group_permissions
 - auth_permission
 - django_admin_log
 - django_content_type
 - django_migrations
 - django_session
 - jodlplatform_note
 - jodlplatform_user**
 - Columns (6)
 - Indexes (0)
 - System Indexes (1)
 - Triggers (0)
 - Views (0)
 - System Catalogue (2)

Col: 1 Row: 1/1

Full View Item View Script Output

	password	last_login	id	email	name	is_staff
1	32ceb14da5f7c28f8f37ffc9a7364a6b	{null}	0	wenke.schubert@jodlgang.com	Wenke Schubert	0
2	b5715d99a4281ef920c775565c72bc72	{null}	1	mila.dietrich@jodlgang.com	Mila Dietrich	0
3	b671ba066f3eb00723e5726ff411e4e0	{null}	2	leni.schmitt@jodlgang.com	Leni Schmitt	0
4	b683cbe128273d66bc15eddd50a34a0e	{null}	3	maila.richter@jodlgang.com	Maila Richter	0
5	13463923d5a18079ed73d7249d27a1df	{null}	4	clara.gross@jodlgang.com	Clara Groß	0
6	33189d99c0f7172c85c0f50cbebf72fb	{null}	5	hannes.krueger@jodlgang.com	Hannes Krüger	0
7	8486e43f7987c827da0dd4685205aa6b	{null}	6	pia.hahn@jodlgang.com	Pia Hahn	0
-	11ba46bd1feb440a9a73ca383a6bb80	{null}	7	leila.fischer@jodlgang.com	Leila Fischer	0

Query OK
Row(s) returned: 256 (More rows can be fetched. Scroll the resultset for more rows and/or read the documentation.)

Sqlite: 3.23.1

2. Problem analysis: backends.py

```
13 class FaceAuthenticationBackend(object):
14     def authenticate(self, request, **kwargs):
15         if 'face_img' not in request.FILES:
16             raise PermissionDenied
17
18         try:
19             user = User.objects.get(email=kwargs["username"])
20         except User.DoesNotExist:
21             raise PermissionDenied
22
23         logger.debug("Retrieving face recognition CNN")
24         cnn = get_face_recognition_cnn()
25
26         try:
27             logger.debug("Converting image to numpy array")
28             face_img = np.array(Image.open(request.FILES['face_img'])).astype(np.float)
29         except Exception as e:
30             logger.error("Exception in face recognition: {} ({}).format(str(e), type(e))")
31             raise PermissionDenied
32
33         if len(face_img.shape) != 3 or face_img.shape[0] != cnn.input_height or face_img.sh
34             logger.info("Dimensions mismatch")
35             raise PermissionDenied
36
37         try:
38             before = time.time()
39             class_probabilities = cnn.inference(face_img[None, :])[0]
40             after = time.time()
41             logger.debug("Inference took {} seconds ...".format(after - before))
42             most_likely_class = np.argmax(class_probabilities)
43             if class_probabilities[most_likely_class] <= 0.5 or user.id != most_likely_class
44                 raise PermissionDenied
45             return user
46         except Exception as e:
47             logger.error("Exception in face recognition: {} ({}).format(str(e), type(e))")
48
49     def get_user(self, user_id):
50         try:
51             user = User.objects.get(id=user_id)
52             return user
53         except User.DoesNotExist:
54             return None
```

2. Problem analysis

- Login with one of of 530 known emails
- “Password” is an image
 - Must be $(\text{cnn.input_width} * \text{cnn.input_width} * 3)$
 - So: $224 * 224 * 3 = \text{square RGB}$
- `Cnn.inference(face)`
 - Argmax
 - must be >0.5

2. Problem Analysis: Model

```
admin.py  backends.py  utils.py
1  from tensorflow.model import FaceRecognitionCNN
2  from jodlplatform import face_recognition_cnn
3  from django.conf import settings
4  import numpy as np
5  import os
6
7
8  def get_face_recognition_cnn():
9      global face_recognition_cnn
10     if face_recognition_cnn is None:
11         np.seterr(all="raise")
12         # Set up face recognition CNN
13         weights_file = getattr(settings, "CNN_WEIGHTS", None)
14         if weights_file is None or not os.path.exists(weights_file):
15             raise ValueError("Weights for face recognition CNN could not be found")
16
17         face_recognition_cnn = FaceRecognitionCNN()
18         face_recognition_cnn.restore_weights(weights_file)
19
20     return face_recognition_cnn
21
```

2. Problem Analysis: Model #2

```
admin.py  backends.py  utils.py  model.py
32 conv3_2 = ConvLayer(3, 256, 256, RectifiedLinearUnit(), TruncatedNormalInitializer(m
33 conv3_3 = ConvLayer(3, 256, 256, RectifiedLinearUnit(), TruncatedNormalInitializer(m
34 pool3 = MaxPoolLayer(2, padding=0, stride=2)
35
36 # Block 4
37 # (28, 28, 256) -> (14, 14, 512)
38 conv4_1 = ConvLayer(3, 256, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
39 conv4_2 = ConvLayer(3, 512, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
40 conv4_3 = ConvLayer(3, 512, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
41 pool4 = MaxPoolLayer(2, padding=0, stride=2)
42
43 # Block 5
44 # (14, 14, 512) -> (7, 7, 512)
45 conv5_1 = ConvLayer(3, 512, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
46 conv5_2 = ConvLayer(3, 512, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
47 conv5_3 = ConvLayer(3, 512, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
48 pool5 = MaxPoolLayer(2, padding=0, stride=2)
49
50 fc6 = FullyConnectedLayer(7 * 7 * 512, 4096, RectifiedLinearUnit(), TruncatedNormalI
51 fc7 = FullyConnectedLayer(4096, 4096, RectifiedLinearUnit(), TruncatedNormalInitiali
52 fc8 = FullyConnectedLayer(4096, 530, Softmax(), TruncatedNormalInitializer(mean=0, s
53
54 self._layers = OrderedDict([
55     ("conv1_1", conv1_1),
56     ("conv1_2", conv1_2),
57     ("pool1", pool1),
58     ("conv2_1", conv2_1),
59     ("conv2_2", conv2_2),
60     ("pool2", pool2),
61     ("conv3_1", conv3_1),
62     ("conv3_2", conv3_2),
63     ("conv3_3", conv3_3),
64     ("pool3", pool3),
65     ("conv4_1", conv4_1),
66     ("conv4_2", conv4_2),
67     ("conv4_3", conv4_3),
68     ("pool4", pool4),
69     ("conv5_1", conv5_1),
70     ("conv5_2", conv5_2),
71     ("conv5_3", conv5_3),
72     ("pool5", pool5),
73     ("fc6", fc6),
74     ("fc7", fc7),
75     ("fc8", fc8),
76 ])
```



Wait a second....

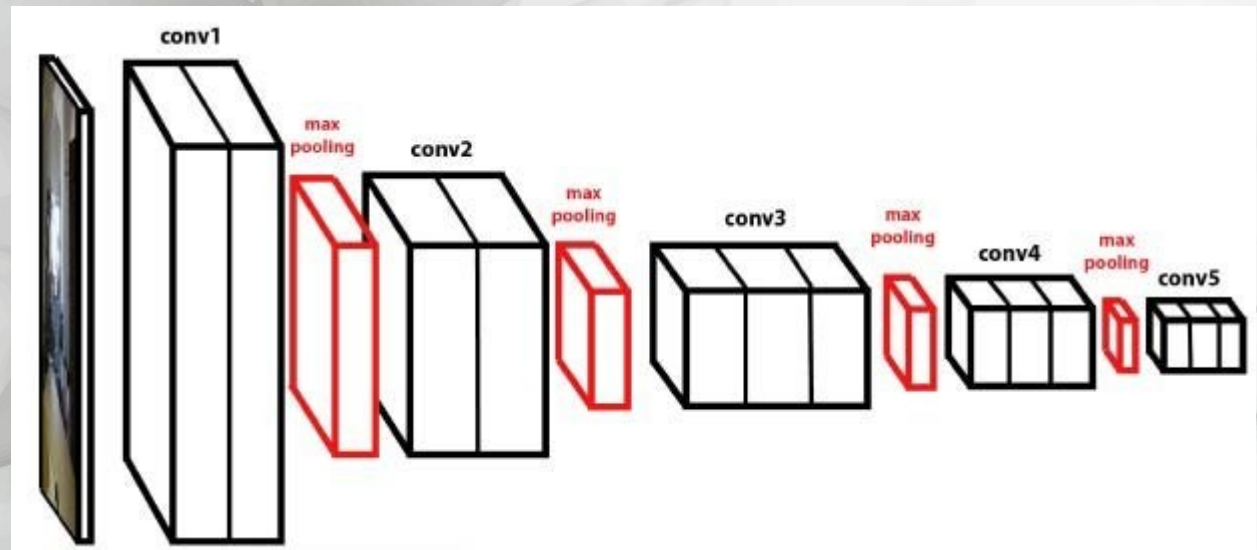
```
FullyConnectedLayer(4096, 4096, RectifiedLinearU
FullyConnectedLayer(4096, 530, Softmax(), Trunca
layers = OrderedDict([
```

Full View Item View Script Output					
	password	last_login	id	email	name
524	eec17e0aa74b036ceac599a61f60d74c	{null}	523	moritz.schuster@jodlgang.com	Moritz Schuster
525	b60a95571e70159ee37e8dd1afddfe73	{null}	524	olivia.schwarz@jodlgang.com	Olivia Schwarz
526	6f03979b4e408c1f3ce1deeb14fe410e	{null}	525	sofia.frank@jodlgang.com	Sofia Frank
527	b1148c5a6b91b9285e4e111594d60a7d	{null}	526	malte.busch@jodlgang.com	Malte Busch
528	59d7436e93e95a9178cd5f1ea9c5651d	{null}	527	miriam.schwarz@jodlgang.com	Miriam Schwarz
529	c4ca14159a509da603b4eae4ec80b1ae	{null}	528	fiona.gross@jodlgang.com	Fiona Groß
530	7746780551ecd72d23f11558fed95041	{null}	529	laura.bauer@jodlgang.com	Laura Bauer

3. Machine Learning: VGG

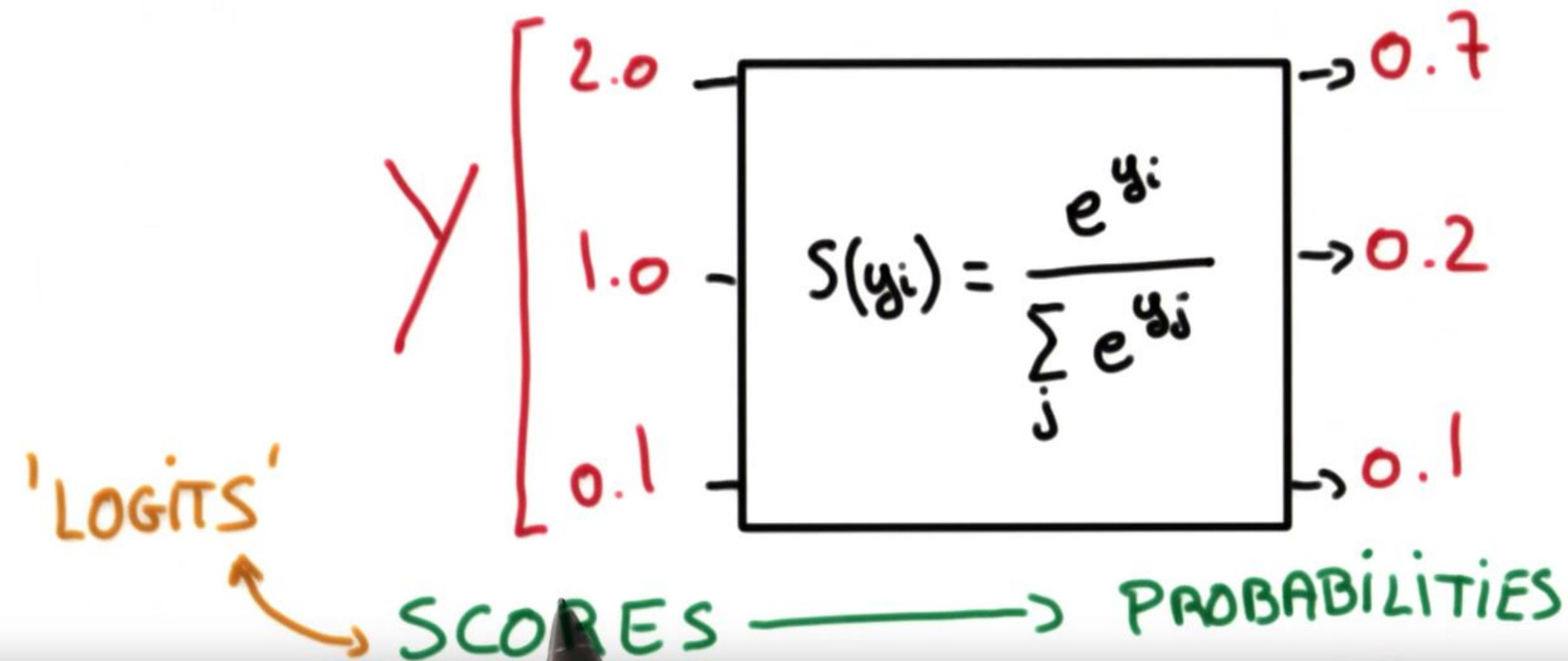
```
self._layers = OrderedDict([
    ("conv1_1", conv1_1),
    ("conv1_2", conv1_2),
    ("pool1", pool1),
    ("conv2_1", conv2_1),
    ("conv2_2", conv2_2),
    ("pool2", pool2),
    ("conv3_1", conv3_1),
    ("conv3_2", conv3_2),
    ("conv3_3", conv3_3),
    ("pool3", pool3),
    ("conv4_1", conv4_1),
    ("conv4_2", conv4_2),
    ("conv4_3", conv4_3),
    ("pool4", pool4),
    ("conv5_1", conv5_1),
    ("conv5_2", conv5_2),
    ("conv5_3", conv5_3),
    ("pool5", pool5),
    ("fc6", fc6),
    ("fc7", fc7),
    ("fc8", fc8),
```

1)



3. Machine Learning: Softmax

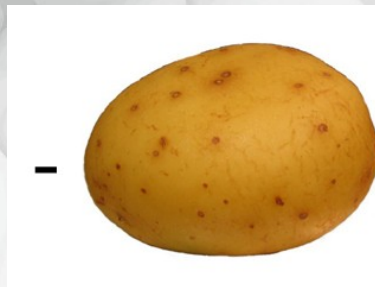
SOFTMAX



3. ML – The softmax issue

- Does this mean....

File: potato.jpg	User: Helena Döring	Prob: 0.63
File: bitconnect_dude.jpg	User: Marlene Schuster	Prob: 0.51
File: trump.jpg	User: Paula Hahn	Prob: 0.99
File: bitcoin.jpg	User: Pia Ingerfurth	Prob: 0.35



3. Machine Learning: CelebA Dataset

- Right: User: Theo-Fuchs.jpg from network traffic (user 60)
- Bottom: CelebA
 - 200k images
 - 10k identities
 - 178×218 px



4. The stupid solution

1. Take random CelebA image
2. Feed it to pretrained model
3. Get classifications → Same for every team
4. If $\max(\text{classifications}) > 50\%$ → JACKPOT
5. Save as $\text{id} = \text{argmax}(\text{classifications})$
6. Repeat on many computers, merge images
7. Try every image-ID on every team, because you forget Theo is #60 and you are team #60
Now they dump your traffic and reuse your images

4. The stupid solution: Code

```
1  #import matplotlib.pyplot as plt
2  from tensorflow.model import FaceRecognitionCNN
3  import json
4  import numpy as np
5  from PIL import Image
6  import os
7  import shutil
8
9  IMG_DIR = "img_align_celeba/"
10 IMG_OUT = "img_out/"
11
12 cnn = FaceRecognitionCNN()
13 cnn.restore_weights("cnn_weights.h5")
14
15 #class label mappings for debug output
16 with open('jodlplatform/migrations/class_label_mapping_names.json') as f:
17     name = json.load(f)
18
19 rand_files = np.random.choice(os.listdir(IMG_DIR), 1000) #1000 random images
20 for f in rand_files:
21     face_img = Image.open(IMG_DIR+f)
22     face_img = face_img.resize((224,224))
23     #plt.imshow(face_img); plt.show()
24     face_img = np.array(face_img).astype(np.float)[None, :] #extend by one dim
25     probabilities = cnn.inference(face_img)
26     userid = np.argmax(probabilities)
27     chance = probabilities[0,userid]
28     if chance > 0.5:
29         print("Name",name[userid],"Probability",probabilities[0,userid], "Filename", f)
30         shutil.copyfile(IMG_DIR+f, IMG_OUT+str(userid)+".jpg")
```

5. The smart solution

- Import weights into Keras/Tensorflow like a sane person → get x100 speedup on GPU
- Use a pretrained optimizer to maximize the class of a given output: “Activation maximization”
- Get trippy images like this.
- Read “How convolutional neural nets see the world” to understand why

