SIGFLAG

Writeup: FaustCTF 2018: Jodlgang

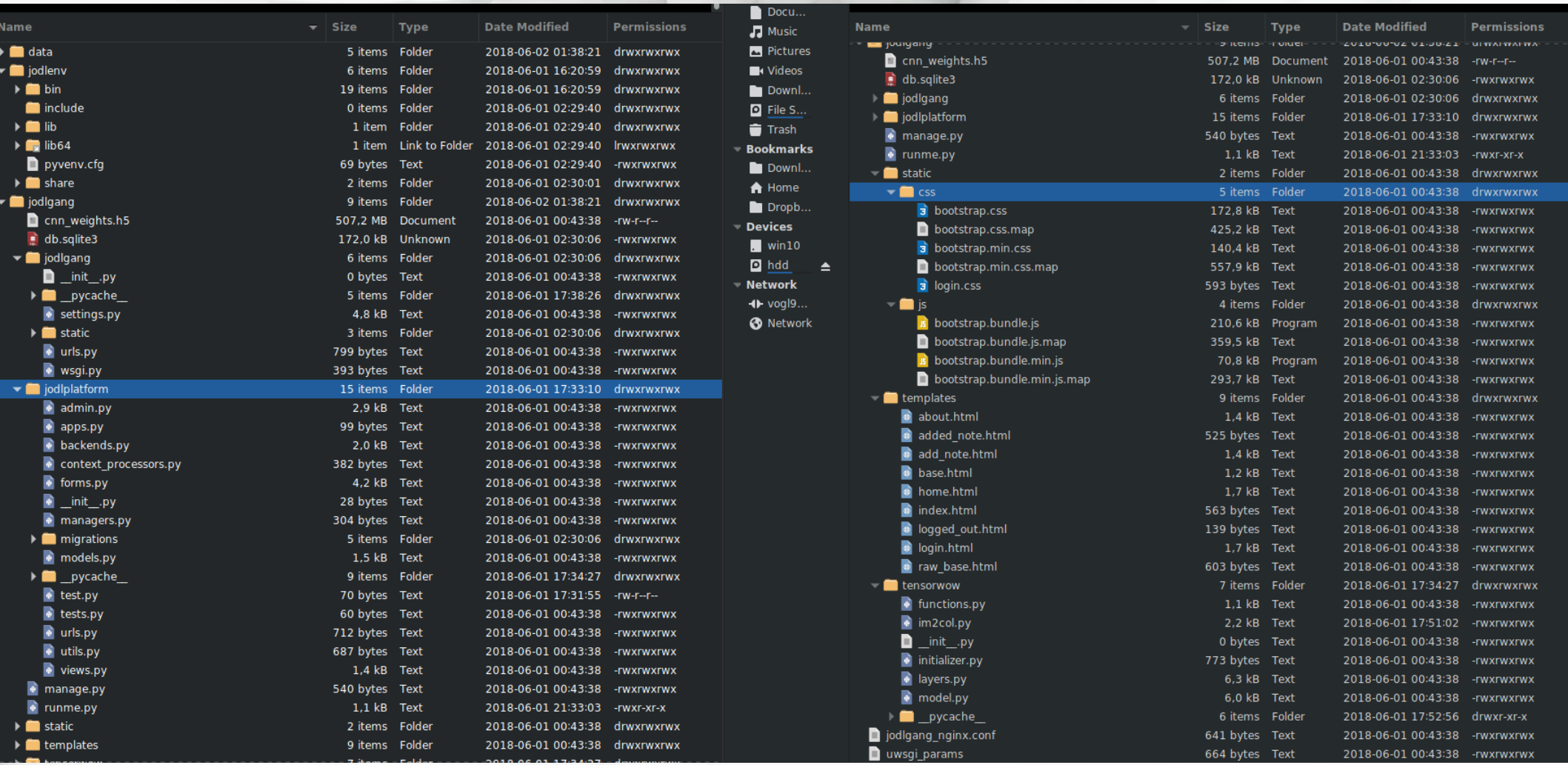
Student: Markus Vogl
Team: SIGFLAG.at
Uni: JKU Linz
Institute: Institute for Network and Security
Sponsor: VACE

Overview

SIGFLAG

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

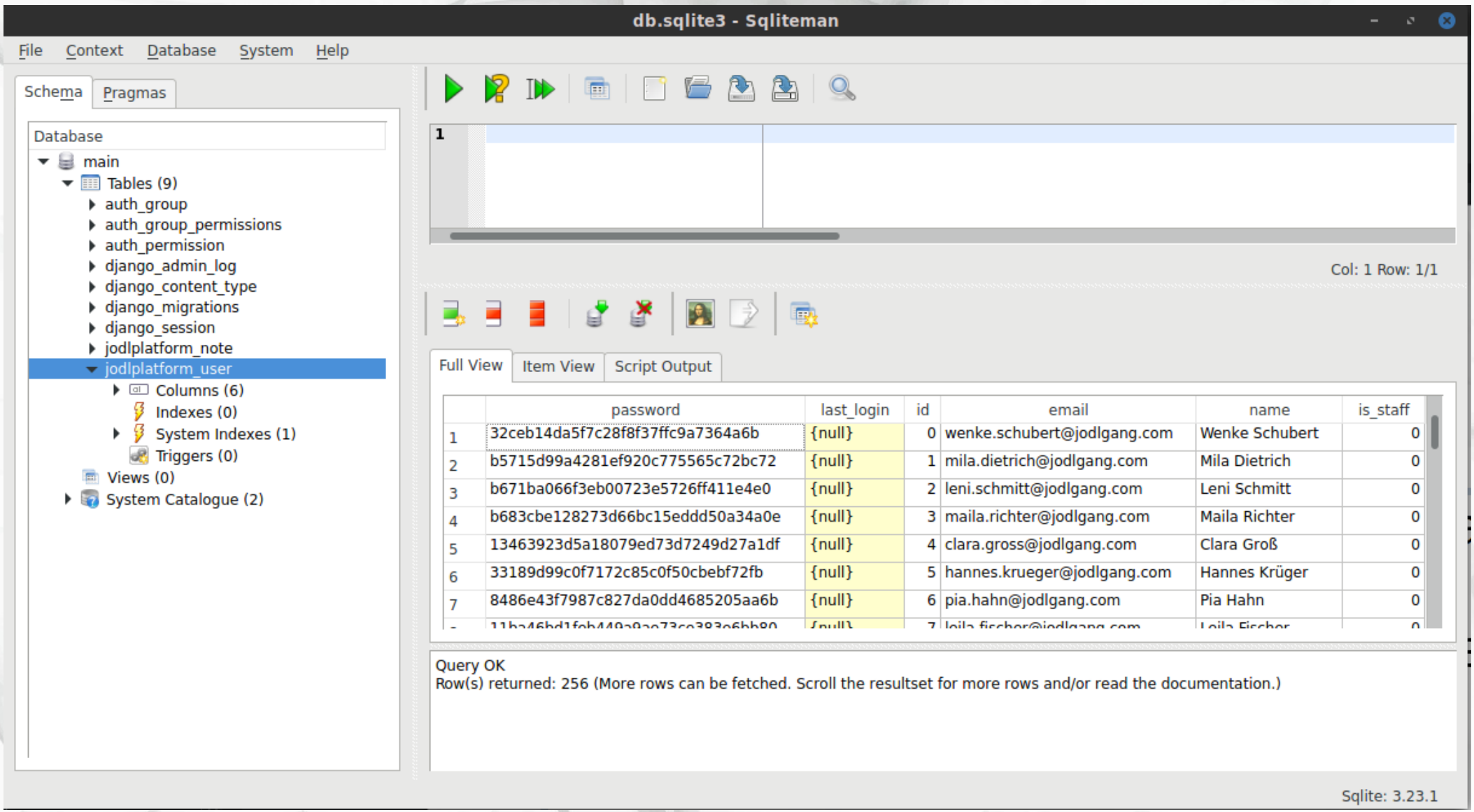
1. Application Overview



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
- Tensorwow
  - Numpy implementation of Tensorflow
  - 500MB .h5 pretrained model

1. Application Overview: Database



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.sha
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 (cnn.input_width * cnn.input_width * 3)
  - So: 224 x 224 x 3 = square RGB
- Cnn.inference(face)
  - Argmax
  - must be >0.5

2. Problem Analysis: Model

admin.py ⊘ backends.py ⊘ utils.py ⊘

```python
from tensorwow.model import FaceRecognitionCNN
from jodlplatform import face_recognition_cnn
from django.conf import settings
import numpy as np
import os


def get_face_recognition_cnn():
    global face_recognition_cnn
    if face_recognition_cnn is None:
        np.seterr(all="raise")
        # Set up face recognition CNN
        weights_file = getattr(settings, "CNN_WEIGHTS", None)
        if weights_file is None or not os.path.exists(weights_file):
            raise ValueError("Weights for face recognition CNN could not be found")

        face_recognition_cnn = FaceRecognitionCNN()
        face_recognition_cnn.restore_weights(weights_file)

    return face_recognition_cnn

```

2. Problem Analysis: Model #2

admin.py | backends.py | utils.py | model.py

```
        conv3_2 = ConvLayer(3, 256, 256, RectifiedLinearUnit(), TruncatedNormalInitializer(m
        conv3_3 = ConvLayer(3, 256, 256, RectifiedLinearUnit(), TruncatedNormalInitializer(m
        pool3 = MaxPoolLayer(2, padding=0, stride=2)

        # Block 4
        # (28, 28, 256) -> (14, 14, 512)
        conv4_1 = ConvLayer(3, 256, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
        conv4_2 = ConvLayer(3, 512, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
        conv4_3 = ConvLayer(3, 512, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
        pool4 = MaxPoolLayer(2, padding=0, stride=2)

        # Block 5
        # (14, 14, 512) -> (7, 7, 512)
        conv5_1 = ConvLayer(3, 512, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
        conv5_2 = ConvLayer(3, 512, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
        conv5_3 = ConvLayer(3, 512, 512, RectifiedLinearUnit(), TruncatedNormalInitializer(m
        pool5 = MaxPoolLayer(2, padding=0, stride=2)

        fc6 = FullyConnectedLayer(7 * 7 * 512, 4096, RectifiedLinearUnit(), TruncatedNormalI
        fc7 = FullyConnectedLayer(4096, 4096, RectifiedLinearUnit(), TruncatedNormalInitiali
        fc8 = FullyConnectedLayer(4096, 530, Softmax(), TruncatedNormalInitializer(mean=0, s

        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),
        ])
```

Wait a second….

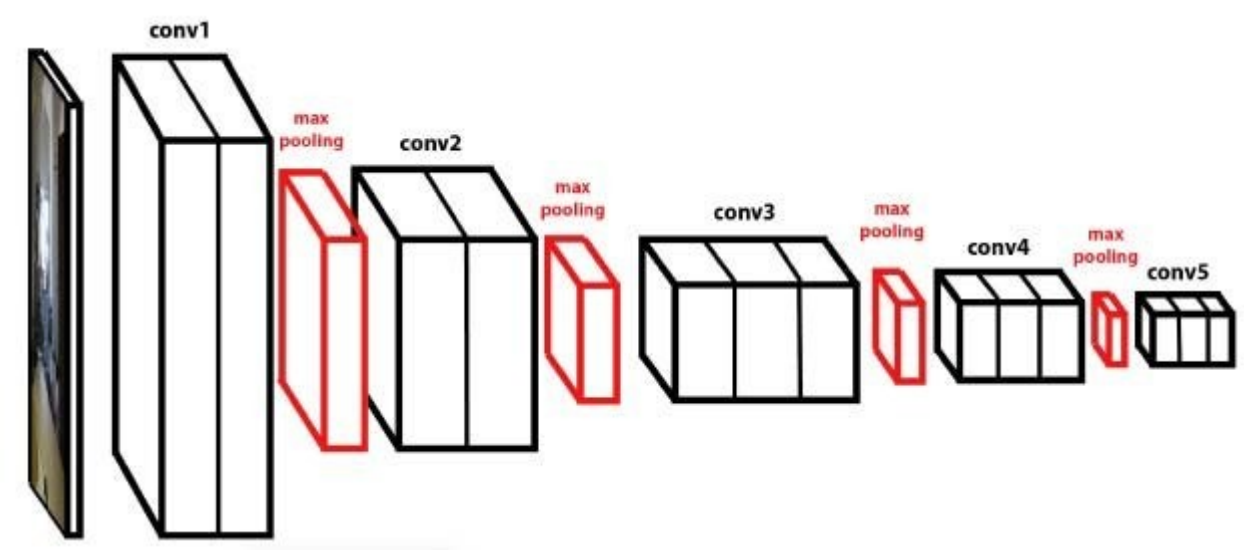
```
FullyConnectedLayer(4096, 4096, RectifiedLinearU
FullyConnectedLayer(4096, 530, Softmax(), Trunca

layers = OrderedDict([
```

Full View | Item View | Script Output

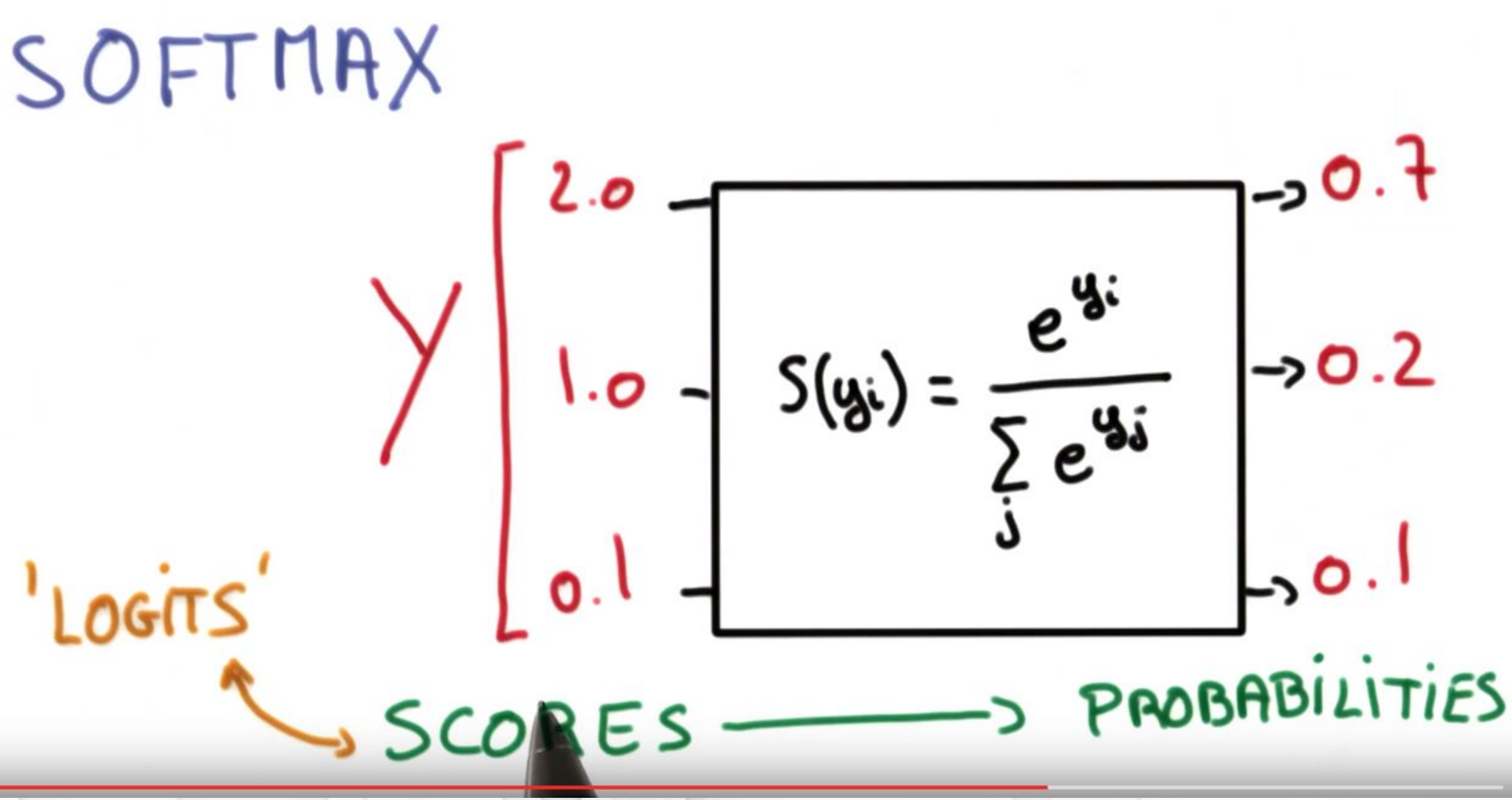
	password	last_login	id	email	name
524	eec17e0aa74b036ceac599a61f60d74c	{null}	523	moritz.schuster@jodlgang.com	Moritz Schuster
525	b60a95571e70159ee37e8dd1afddfe73	{null}	524	olivia.schwarz@jodlgang.com	Olivia Schwarz
526	6f03979b4e408c1f3ce1deeb14fe410e	{null}	525	sofia.frank@jodlgang.com	Sofia Frank
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),
])
```



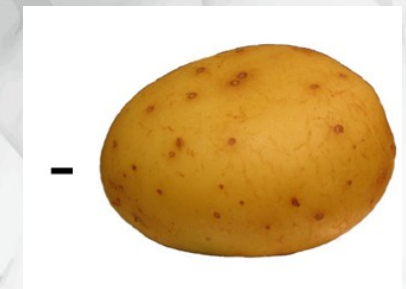
3. Machine Learning: Softmax



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(classifications) > 50% → JACKPOT
5. Save as id = argmax(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

```python
#import matplotlib.pyplot as plt
from tensorwow.model import FaceRecognitionCNN
import json
import numpy as np
from PIL import Image
import os
import shutil

IMG_DIR = "img_align_celeba/"
IMG_OUT = "img_out/"

cnn = FaceRecognitionCNN()
cnn.restore_weights("cnn_weights.h5")

#class label mappings for debug output
with open('jodlplatform/migrations/class_label_mapping_names.json') as f:
    name = json.load(f)

rand_files = np.random.choice(os.listdir(IMG_DIR), 1000) #1000 random images
for f in rand_files:
    face_img = Image.open(IMG_DIR+f)
    face_img = face_img.resize((224,224))
    #plt.imshow(face_img); plt.show()
    face_img = np.array(face_img).astype(np.float)[None, :] #extend by one dim
    probabilities = cnn.inference(face_img)
    userid = np.argmax(probabilities)
    chance = probabilities[0,userid]
    if chance > 0.5:
        print("Name",name[userid],"Probability",probabilities[0,userid], "Filename", f)
        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