Как я не выиграл конкурс Avito

Евгений Нижибицкий

ВМК МГУ

25 февраля 2015 г.

1. Исходные данные
2. Optical Character Recognition
3. Нейронные сети
   - Пост на fastml.com
   - AlexNet
   - Использование Caffe
4. Итоговый алгоритм
5. Что еще можно было попробовать

Исходные данные

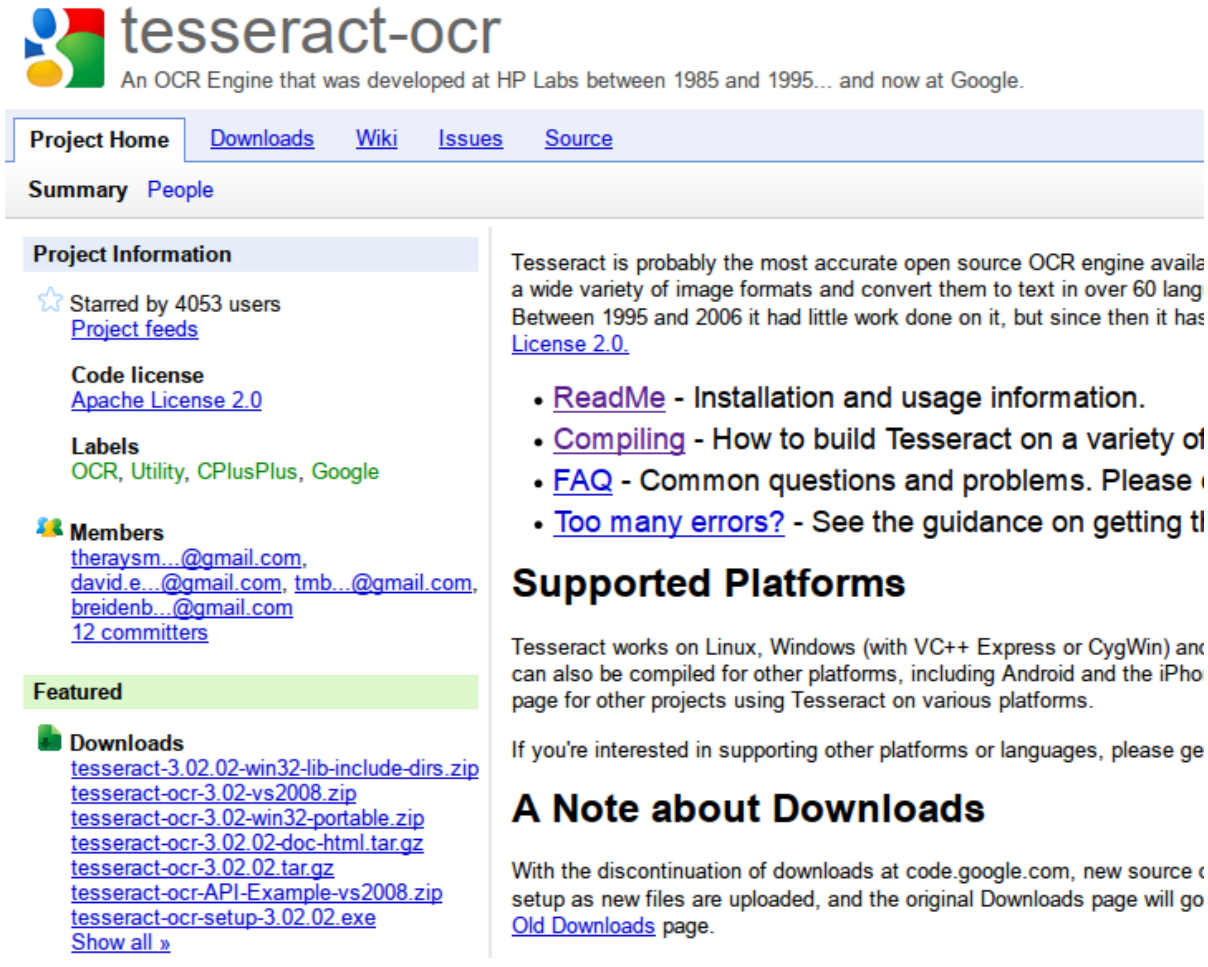
Исходные данные

Исходные данные

Исходные данные

Исходные данные

Очевидное решение — давайте распознавать текст!



tesseract-ocr

An OCR Engine that was developed at HP Labs between 1985 and 1995... and now at Google.

Project Home | Downloads | Wiki | Issues | Source

Summary | People

Project Information

Starred by 4053 users
Project feeds

Code license
Apache License 2.0

Labels
OCR, Utility, CPlusPlus, Google

Members
theraysm...@gmail.com,
david.e...@gmail.com, tmb...@gmail.com,
breidenb...@gmail.com
12 committers

Featured

Downloads
tesseract-3.02.02-win32-lib-include-dirs.zip
tesseract-ocr-3.02-vs2008.zip
tesseract-ocr-3.02-win32-portable.zip
tesseract-ocr-3.02.02-doc-html.tar.gz
tesseract-ocr-3.02.02.tar.gz
tesseract-ocr-API-Example-vs2008.zip
tesseract-ocr-setup-3.02.02.exe
Show all »

Tesseract is probably the most accurate open source OCR engine availa
a wide variety of image formats and convert them to text in over 60 lang
Between 1995 and 2006 it had little work done on it, but since then it has
License 2.0.

- ReadMe - Installation and usage information.
- Compiling - How to build Tesseract on a variety of
- FAQ - Common questions and problems. Please
- Too many errors? - See the guidance on getting t

Supported Platforms

Tesseract works on Linux, Windows (with VC++ Express or CygWin) and
can also be compiled for other platforms, including Android and the iPho
page for other projects using Tesseract on various platforms.

If you're interested in supporting other platforms or languages, please ge

A Note about Downloads

With the discontinuation of downloads at code.google.com, new source c
setup as new files are uploaded, and the original Downloads page will go
Old Downloads page.

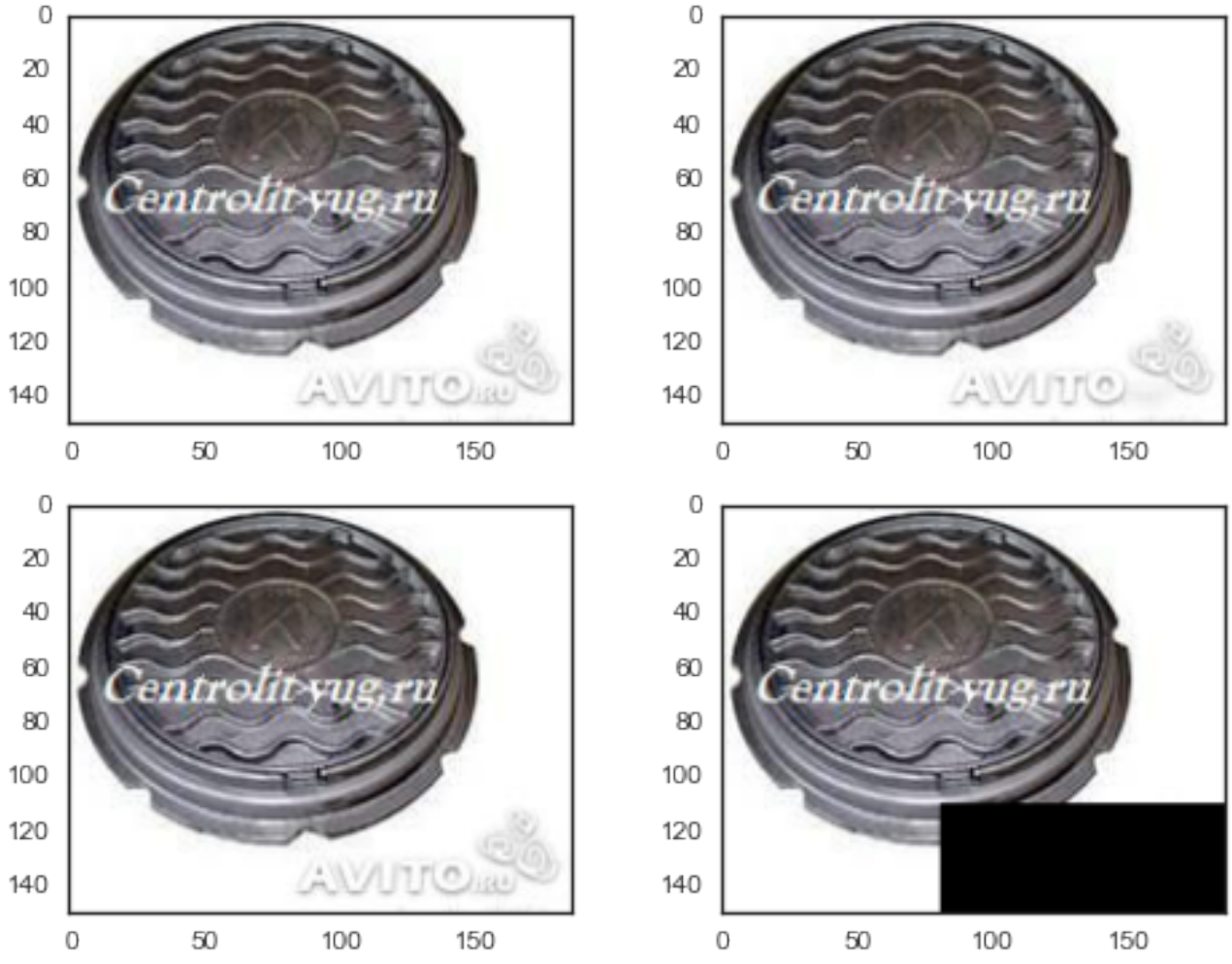
Что плохого, к примеру, в поиске доменов?

Давайте замажем!

```
plt.figure(figsize=(8,6))
im = Image.open('train/129641663_929522737.jpg')
plt.subplot(2,2,1)
plt.imshow(im)
imc = im.crop((im.size[0] - 37, im.size[1] - 15, im.size[0] - 19, im.size[1] - 3))
imc = imc.filter(ImageFilter.MedianFilter(9))
im.paste(imc, (im.size[0] - 37, im.size[1] - 15))
plt.subplot(2,2,2)
plt.imshow(im)

im = Image.open('train/129641663_929522737.jpg')
plt.subplot(2,2,3)
plt.imshow(im)
imc = Image.new('RGB', (105, 40))
im.paste(imc, (im.size[0] - 105, im.size[1] - 40))
plt.subplot(2,2,4)
plt.imshow(im)
plt.show()
```

Optical Character Recognition



Результат OCR

In [73]:
```
img = Image.open('blur/130060451_663770998.jpg')

print "tesseract:"
tool = pyocr.get_available_tools()[0]
langs = tool.get_available_languages()
rus = langs[2]
eng = langs[3]
print tool.image_to_string(img, lang=eng, builder=pyocr.builders.TextBuilder())

print "\ncuneiform:"
tool = pyocr.get_available_tools()[1]
langs = tool.get_available_languages()
rus = langs[3]
eng = langs[0]
print tool.image_to_string(img, lang=eng, builder=pyocr.builders.TextBuilder())
```

```
tesseract:
CoaaaeM Kpacmable CeMefiHble
m noKyMeHTanthxe dpMnbeL
Bbl a maan pom.
Wedding wdeo
Farmly vrdeo
vmeo sdmng servmes

www.5fpro. ru

cuneiform:
Coaoaeru ffpacwet ie cebfeaabie
N AourMSHTBJlbubie rpaflbMbr
Bbr e roaeuoa pooa


www sfpro ru
```

Результат OCR

	Id	label	tesseract	txtlen
16668	106555603_200745813	0	АМП TAVE T	10
15992	107731247_203283026	0	| Ад! ‹ шита ЪЁЁІ АМ! (WWW/g 2’45	36
12124	109780842_207678716	0	Рапазопбс п ь▷ ...	151
238	109875184_207885561	0	у % ПЛИТ ;V е02 “NT ’4	22
7074	110132756_208424448	0	5000руб. 50006 {бкВт „ ЛУИ Т©5000ру...	82

Всего **30%** изображений с нарушениями

Всего **21%** изображений с обнарежнием текста

На примерах без нарушений **15%** обнаружений текста

На примерах с нарушениями **36%** обнаружений текста

Из изображений с обнаружением текста **49%** с нарушениями

Ищем паттерны со 100% Precision:

- Номера: (XX, XX), -XX, XXX + ['8(', '8 (', '(8', '8-8', '+79']
- Части слов из корпусов (без крайних букв)
- Паттерны, придуманные вручную

```
urls = ['.ru', 'vk.', 'http', ':/', '.net', u'.рф', '.ua']
emails = ['@ma', 'e-ma', '@gm']
words = [u'монта', u'тел.', u'звонит', u'кред', u'автом', u'стоим', u'адрес',
         u'конт', u'москва', u'бург', u'красн', u'продае', u'окна', u'окон',
         u'shop', u'под ключ', u'слуг', u'недв']

test = ['abcde']
patterns = test + urls + emails + words # + telephones

scores = []
for pattern in patterns:
    scores.append(find(pattern))
scores = np.array(scores)
for tup in sorted(scores, key=lambda score: float(score[1]), reverse=True):
    print "{}\t{}\t{}".format(tup[0], tup[1], tup[2])
```

```
.ru     262     1.0
тел.    76      1.0
vk.     73      1.0
звонит  65      1.0
:/      54      1.0
монта   46      1.0
http    45      1.0
слуг    43      1.0
недв    38      1.0
@ma     26      1.0
```

Находим около 20% нарушений (100% Precision, 20% Recall)

```
patterns = urls + emails + words + parts + telephones
patterns = list(set(patterns))

ytrue = train['label']
ypred = np.zeros_like(train['label'])
for pattern in patterns:
    ypred[np.array(train['tesseract'].apply(lambda s:
                                            s.lower().encode("utf-8")
                                            .find(pattern.encode("utf-8")) > 0))] += 1
print sum(ypred > 0)
print auc(ytrue, ypred)
print acc(ytrue[ypred > 0], ypred[ypred > 0]*0 + 1)
```

```
2377
0.59878646829
1.0
0.594981166825
```

А сколько (не)реально вообще найти?

```
ytrue = train['label']
ypred = np.zeros_like(train['label'])
ypred[np.array((train['label'] == 1) & (train['txtlen'] > 0))] = 1
print sum(ypred)
print auc(ytrue, ypred)
print acc(ytrue[ypred > 0], ypred[ypred > 0])
```

```
4408
0.683193417006
1.0
```

http://fastml.com/yesterday-a-kaggler-today-a-kaggle-master-a-wrap-up-of-the-cats-and-dogs-competition/

2014-02-02

Yesterday a kaggler, today a Kaggle master: a wrap-up of the cats and dogs competition

Out of 215 contestants, we placed 8th in the Cats and Dogs competition at Kaggle. The top ten finish gave us the master badge. The competition was about discerning the animals in images and here's how we did it.

We extracted the features using pre-trained deep convolutional networks, specifically *decaf* and *OverFeat*. Then we trained some classifiers on these features. The whole thing was inspired by Kyle Kastner's *decaf + pylearn2* combo and we expanded this idea. The classifiers were linear models from *scikit-learn* and a neural network from *Pylearn2*. At the end we created a voting ensemble of the individual models.

OverFeat features

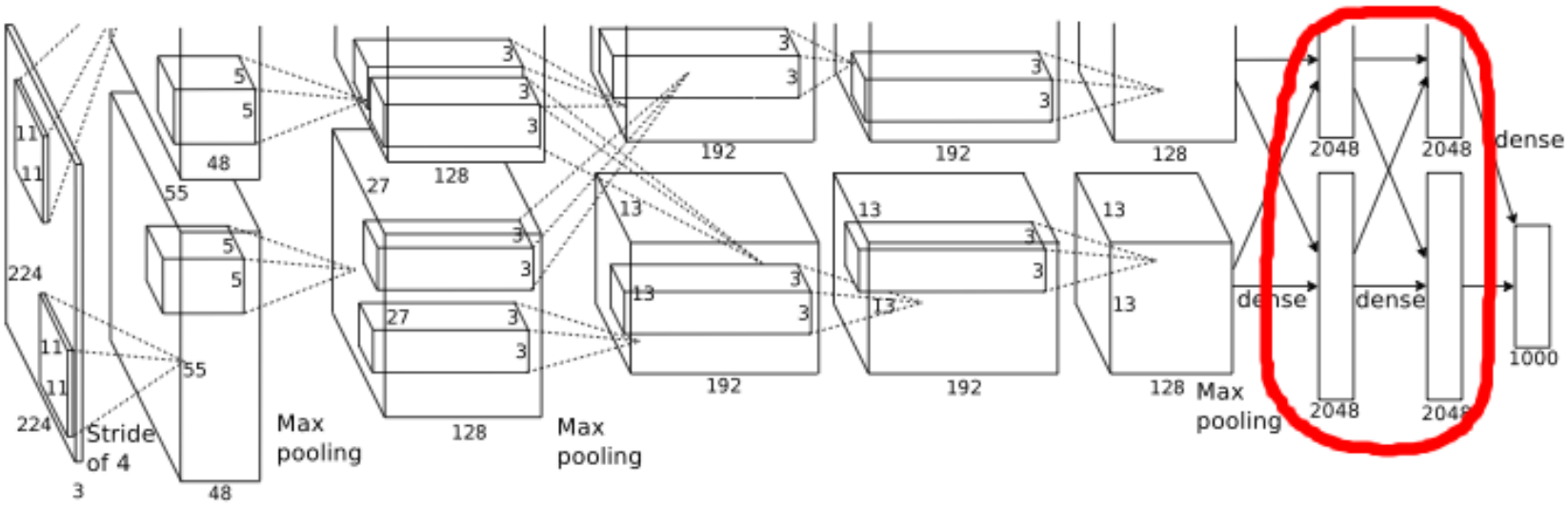
We touched on OverFeat in Classifying images with a pre-trained deep network. A better way to use it in this competition's context is to extract the features from the layer before the classifier, as Pierre Sermanet suggested in the comments.

ImageNet Classification with Deep Convolutional Neural Networks [NIPS 2012]

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca



```
# returns (8192,)-shaped vector
def feature(impath):
    scores = net.predict([caffe.io.load_image(impath)])
    y = np.zeros((10,0,1,1))
    y = np.concatenate((y, net.blobs['fc6'].data), axis=1)
    y = np.concatenate((y, net.blobs['fc7'].data), axis=1)
    y = np.max(y, axis=0)
    return y.flatten()

def extract(data, folder):
    X = np.zeros((data.shape[0], 8192))
    y = np.array(data['label'])
    for i, Id in enumerate(data['Id']):
        X[i,:] = feature('/home/ugin/Science/Avito/b' + folder + Id + '.jpg')
    return X, y
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

Crange = np.logspace(-4.75, -4.25, 10)

scores = np.zeros((len(Crange), 2))
for i, C in enumerate(Crange):
    model = LogisticRegression(C=C)
    model.fit(Xtrain, ytrain)
    print "%.8f  %.4f  %.4f  %.4f" % (C, roc_auc_score(ytrain, model.predic
                                       roc_auc_score(yvalid[:5000], model
                                       roc_auc_score(yvalid[5000:], model
```

```
0.00001778  0.9282  0.8906  0.9060
0.00002021  0.9297  0.8910  0.9063
0.00002297  0.9313  0.8915  0.9065
0.00002610  0.9327  0.8918  0.9066
0.00002966  0.9342  0.8920  0.9067
0.00003371  0.9356  0.8922  0.9066
0.00003831  0.9369  0.8924  0.9066
0.00004354  0.9382  0.8924  0.9065
0.00004948  0.9395  0.8924  0.9062
0.00005623  0.9408  0.8924  0.9060
```

```
print "tanh:"
for alpha in np.linspace(0,1,11):
    pred = alpha * cpred + (1-alpha) * np.tanh(0.25*tpred)
    print "%.3f  %.4f %.4f" % (alpha, roc_auc_score(yvalid[:5000], pred[:5000]),
                                     roc_auc_score(yvalid[5000:], pred[5000:]))
print

print "prod:"
for alpha in np.logspace(-1.5,0.5,10):
    pred = (alpha + tpred) * cpred
    print "%.3f  %.4f %.4f" % (alpha, roc_auc_score(yvalid[:5000], pred[:5000]),
                                     roc_auc_score(yvalid[5000:], pred[5000:]))
```

```
max:   0.8922 0.9077

tanh:
0.000  0.5840 0.5973
0.100  0.8926 0.9083
0.200  0.8927 0.9083
0.300  0.8937 0.9093
0.400  0.8941 0.9096
0.500  0.8944 0.9099
0.600  0.8945 0.9099
0.700  0.8945 0.9096
0.800  0.8941 0.9090
0.900  0.8934 0.9080
1.000  0.8923 0.9066
```

```
print "train: %.4f" % roc_auc_score(np.array(train_df['label']), tr
print "valid: %.4f" % roc_auc_score(np.array(valid_df['label'].head
print "valid: %.4f" % roc_auc_score(np.array(valid_df['label'].tail
```

```
train: 0.9389
valid: 0.8944
valid: 0.9099
```

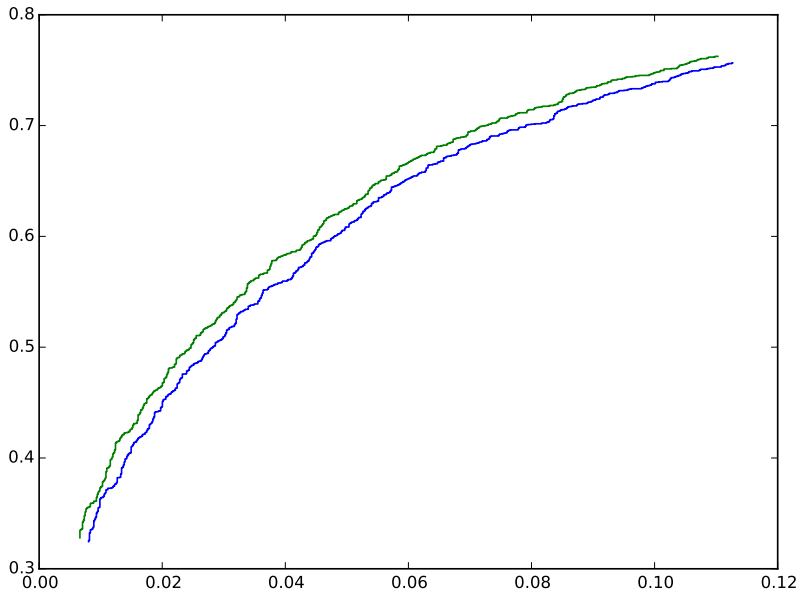
#	Никнейм участника	Результат
1	Alexandra Fenster	0.9495
2	Ульянов Дмитрий	0.9374
3	Нижибицкий Евгений	0.9202
4	Sukhinov	0.9164
5	Signneuro	0.8897

Итоговый алгоритм

На основе выхода caffe-регрессии производится смешение с вектором частот срабатываний паттерн-матчинга на tesseract-паттернах преобразованием вида

$$y_{pred} = \alpha \cdot y_{caffe} + (1 - \alpha) \cdot \tanh(0.25 \cdot freq_{tesseract}), \alpha = 0.5,$$

которое добавляет еще пару-тройку тысячных к AUC-ам:

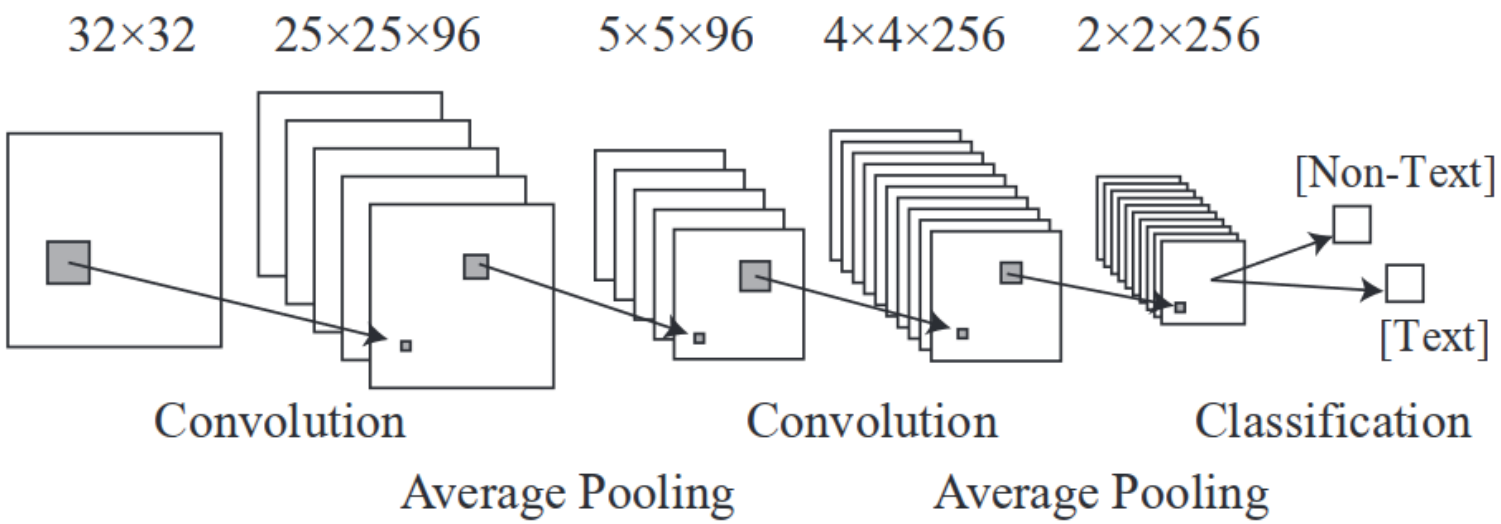


Что еще можно было попробовать

Обучение нейросети для поиска символов

End-to-End Text Recognition with Convolutional Neural Networks

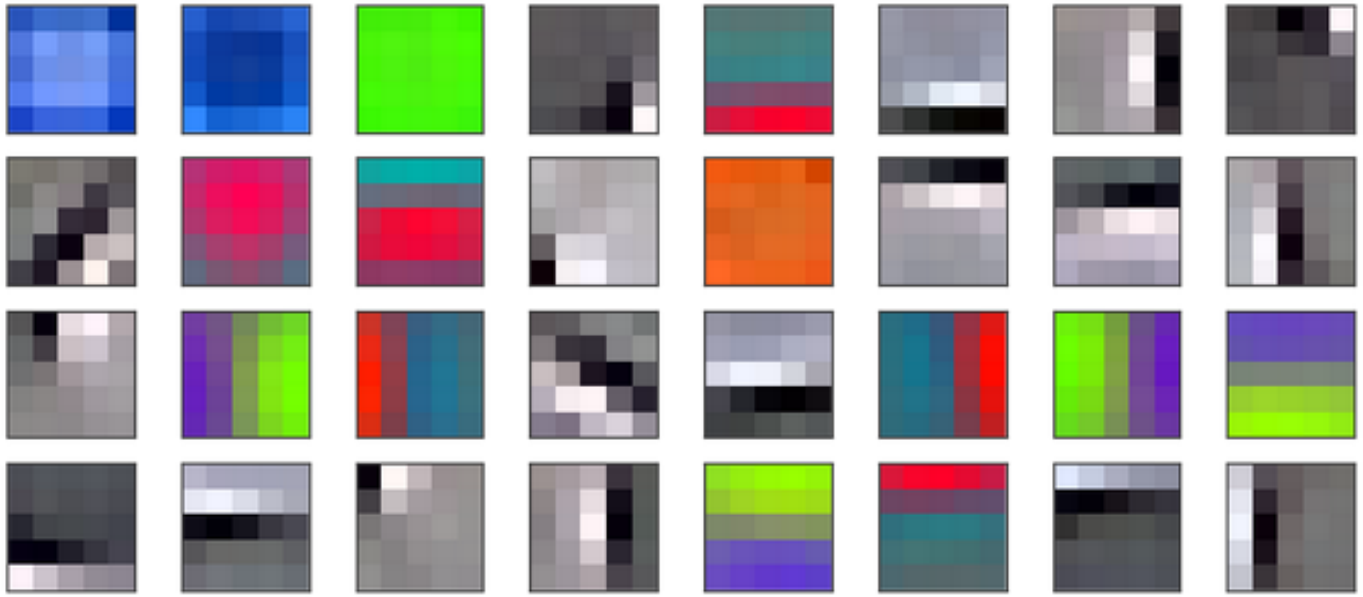
Tao Wang* David J. Wu* Adam Coates Andrew Y. Ng
Stanford University, 353 Serra Mall, Stanford, CA 94305
{*twangcat, dwu4, acoates, ang*}*@cs.stanford.edu*



Что еще можно было попробовать

Обучение первого слоя с помощью K-Means

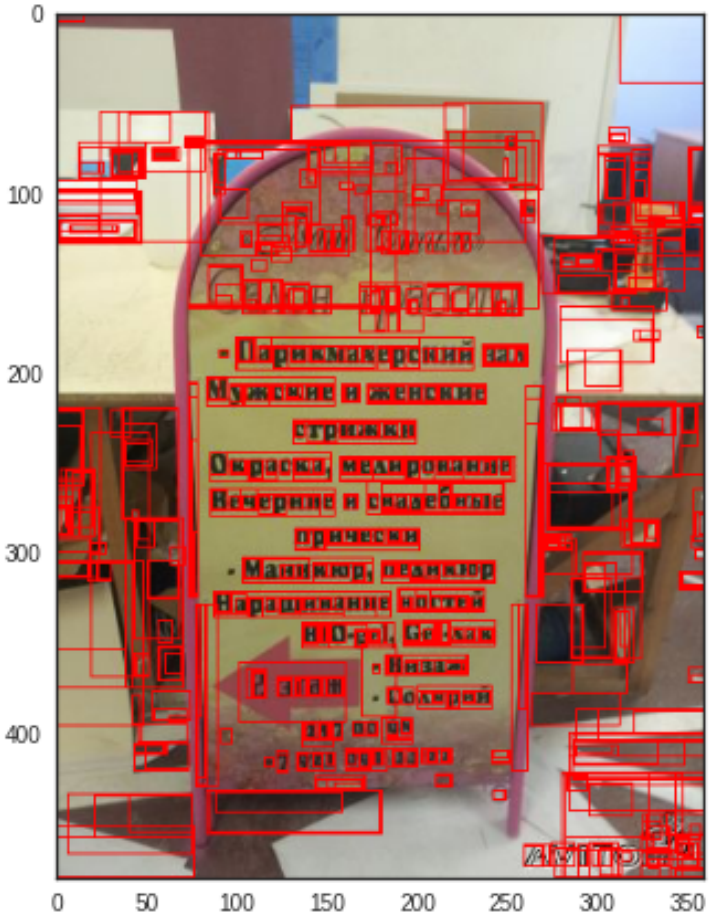
```
# display
fig = plt.figure(figsize=(14, 6))
num_col = int(np.ceil(float(NUM_FILTERS)/4))
for i in xrange(NUM_FILTERS):
    ax = fig.add_subplot(4, num_col, i+1)
    filter_ = filters[i,...]
    filter_ -= filter_.min()
    filter_ /= filter_.max()
    ax.imshow(filter_, interpolation='none')
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()
```



Что еще можно было попробовать

Регионы для проверки ищем с помощью MSER

```
im = imread("0505.jpg")
mim = mser(im, debug=True)
```



Привет!