Image Classification And Object Detection

For researchers interested in studying Earth science with deep learning.

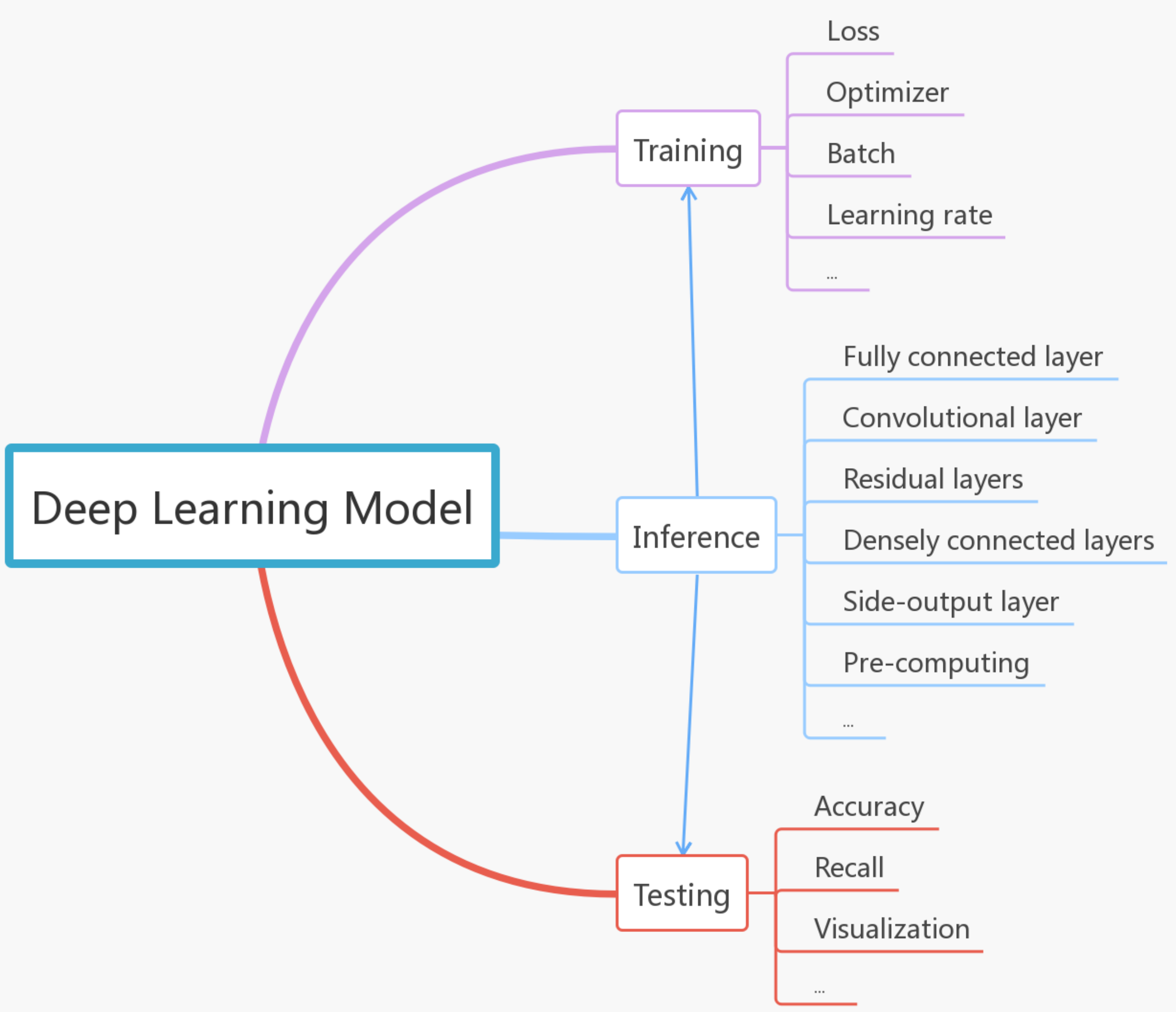
All resources in lectures are available at https://github.com/MrXiaoXiao/DLiES

Deep Learning in Earth Science
Lecture 2
By Xiao Zhuowei

OUTLINES

1. Build Networks with High-level API
2. Classification with Convolutional Neural Networks
3. Object Detection with Bounding Box
4. Discussions

Build Networks with High-level API



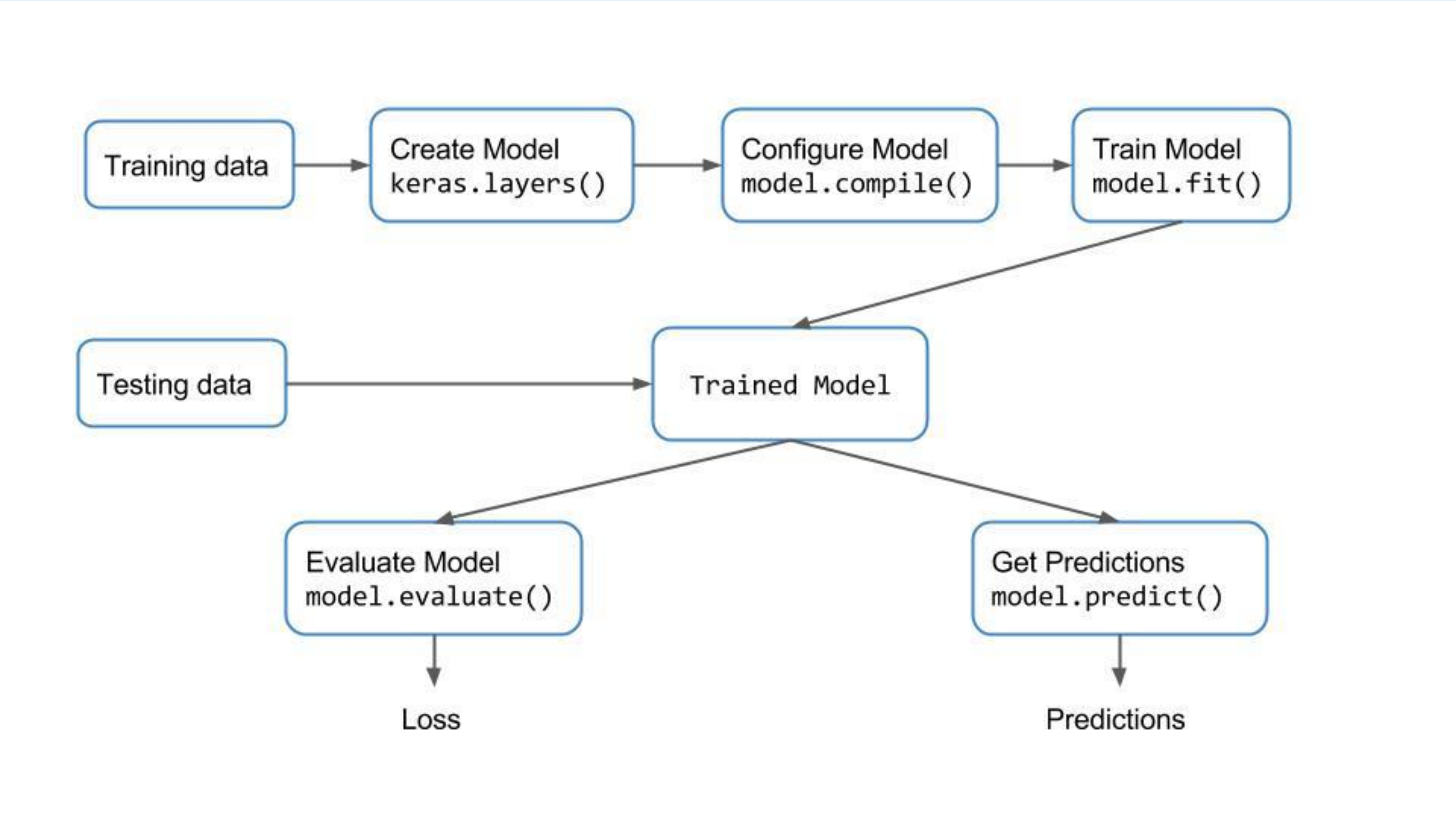
Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

Being able to go from idea to result with the least possible delay is key to doing good research.

Build Networks with High-level API

Keras Workflow



https://www.learnopencv.com/deep-learning-using-keras-the-basics/

Getting started with the Keras

Prepare dataset

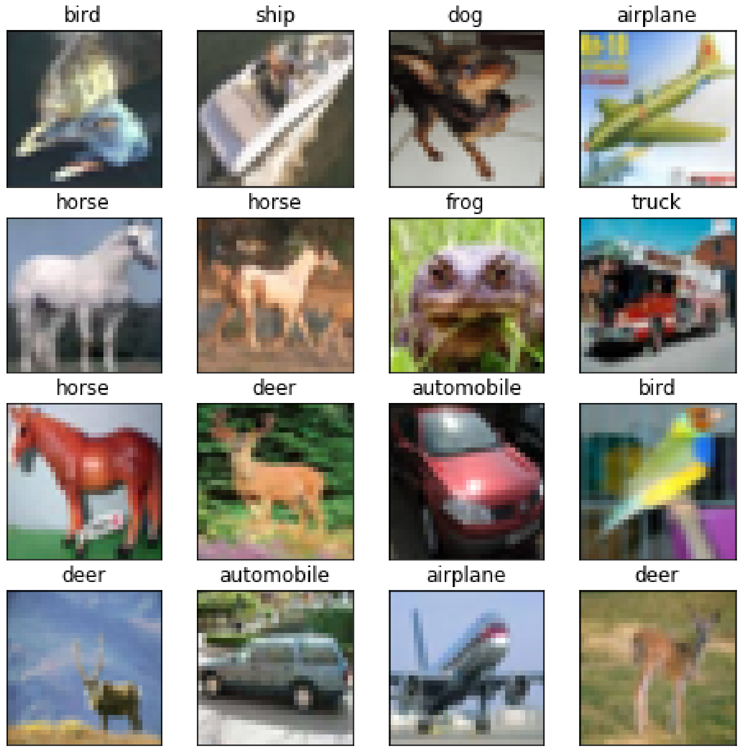
```
import tensorflow as tf
from tensorflow.keras.datasets import cifar10
#train and test using cifar10 dataset
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

#Preprocessing
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

num_classes = 10

# Convert class vectors to binary class matrices.
y_train = tf.keras.utils.to_categorical(y_train, num_classes)
y_test = tf.keras.utils.to_categorical(y_test, num_classes)
```

```
/home/wangj/anaconda3/envs/DLiES/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: numpy.dtype size changed, ma
y indicate binary incompatibility. Expected 88 from C header, got 96 from PyObject
  return f(*args, **kwds)
```

```
x_train shape: (50000, 32, 32, 3)
50000 train samples
10000 test samples
```

Check dataset

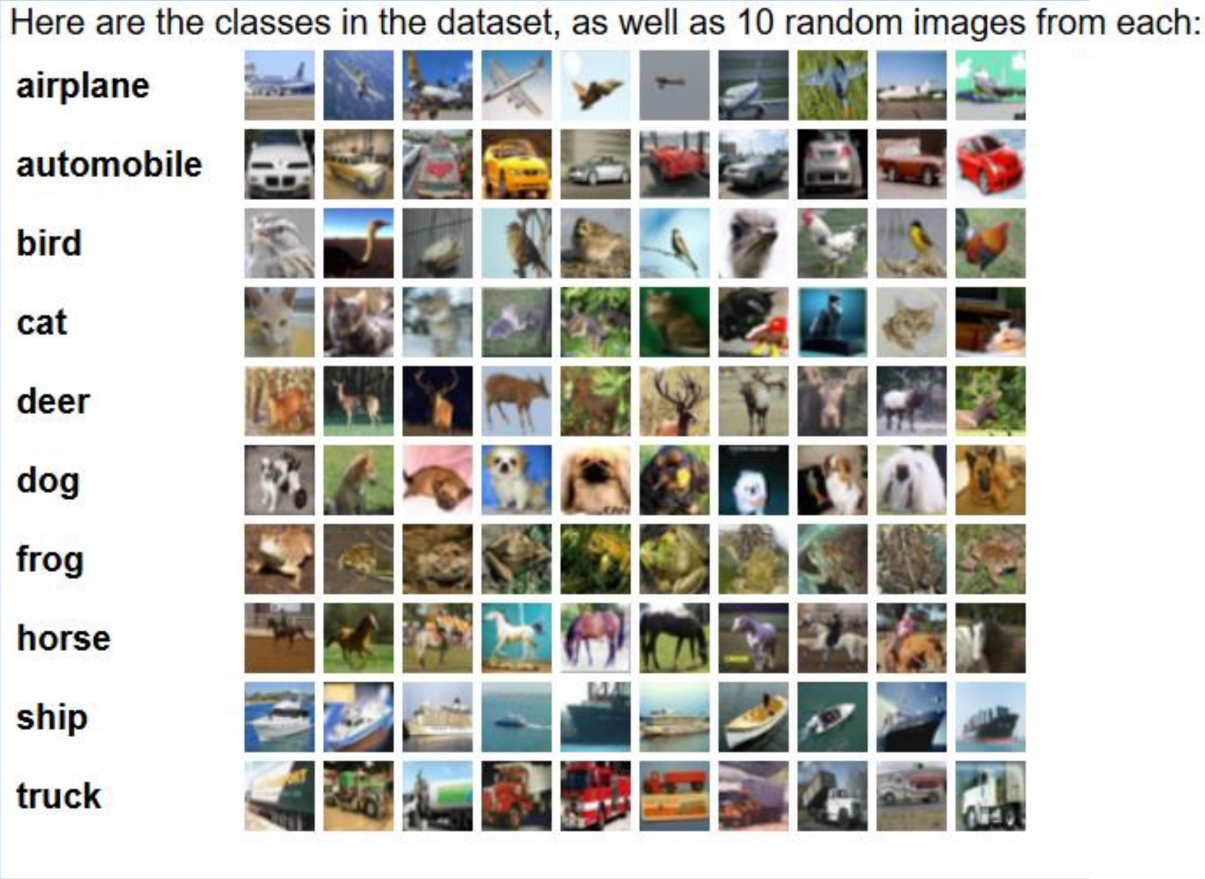
```
import matplotlib.pyplot as plt
import numpy as np
plt.figure(figsize=(8,8))
class_names = ['airplane','automobile','bird','cat','deer',
               'dog','frog','horse','ship','truck']

for i in range(16):
    plt.subplot(4, 4, 1 + i, xticks=[], yticks=[])
    img_id = np.random.randint(50000)
    im = x_train[img_id,::]
    plt.title(class_names[y_train[img_id].argmax()])
    plt.imshow(im)
plt.show()
```



The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

http://www.cs.toronto.edu/~kriz/cifar.html



Getting started with the Keras

Create a model

In [4]:
```
#The Sequential model is a linear stack of layers.
model = tf.keras.Sequential()
```

Add layers

In [5]:
```
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
#You can also simply add layers via the .add() method
model.add(Conv2D(32, (3, 3), padding='same',input_shape=x_train.shape[1:],
                 activation='relu'))
model.add(Conv2D(32, (3, 3), activation='relu') )
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding='same',activation='relu') )
model.add(Activation('relu'))
model.add(Conv2D(64, (3, 3), padding='same',activation='relu') )
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512,activation='relu') )
model.add(Dropout(0.5))
model.add(Dense(10,activation='softmax') )
```

https://keras.io/getting-started/sequential-model-guide/

Build Networks with High-level API

Getting started with the Keras

In [6]:
```
print(model.summary())
```

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 16, 16, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 16, 16, 64)        18496
_________________________________________________________________
activation (Activation)      (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 8, 8, 64)          0
_________________________________________________________________
```

Initiate optimizer

In [7]:
```
opt = tf.keras.optimizers.Adam()
```

Configure model

In [8]:
```
#Before training a model, you need to configure the learning process, which is done via the compile method.
model.compile(loss='categorical_crossentropy',
              optimizer=opt,
              metrics=['accuracy'])
```

Build Networks with High-level API

Train model

In [9]:
```python
#Keras models are trained on Numpy arrays of input data and labels.
#For training a model, you will typically use the fit function.
history = model.fit(x_train, y_train,
                    batch_size=32,
                    epochs=10,
                    shuffle=True)
```

```
Epoch 1/10
50000/50000 [==============================] - 21s 424us/step - loss: 1.5485 - acc: 0.4335
Epoch 2/10
50000/50000 [==============================] - 14s 285us/step - loss: 1.1393 - acc: 0.5926
Epoch 3/10
50000/50000 [==============================] - 14s 283us/step - loss: 0.9782 - acc: 0.6532
Epoch 4/10
50000/50000 [==============================] - 14s 277us/step - loss: 0.8747 - acc: 0.6925
Epoch 5/10
50000/50000 [==============================] - 14s 281us/step - loss: 0.8105 - acc: 0.7142
Epoch 6/10
50000/50000 [==============================] - 14s 277us/step - loss: 0.7452 - acc: 0.7386
Epoch 7/10
50000/50000 [==============================] - 14s 278us/step - loss: 0.7060 - acc: 0.7504
Epoch 8/10
50000/50000 [==============================] - 14s 276us/step - loss: 0.6683 - acc: 0.7654
Epoch 9/10
50000/50000 [==============================] - 13s 266us/step - loss: 0.6360 - acc: 0.7778
Epoch 10/10
50000/50000 [==============================] - 14s 273us/step - loss: 0.6098 - acc: 0.7857
```

Test model

In [10]:
```python
# test trained model.
scores = model.evaluate(x_test, y_test, verbose=1)

print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

```
10000/10000 [==============================] - 1s 114us/step
Test loss: 0.6621272174358368
Test accuracy: 0.7711
```

Build Networks with High-level API

Check predictions

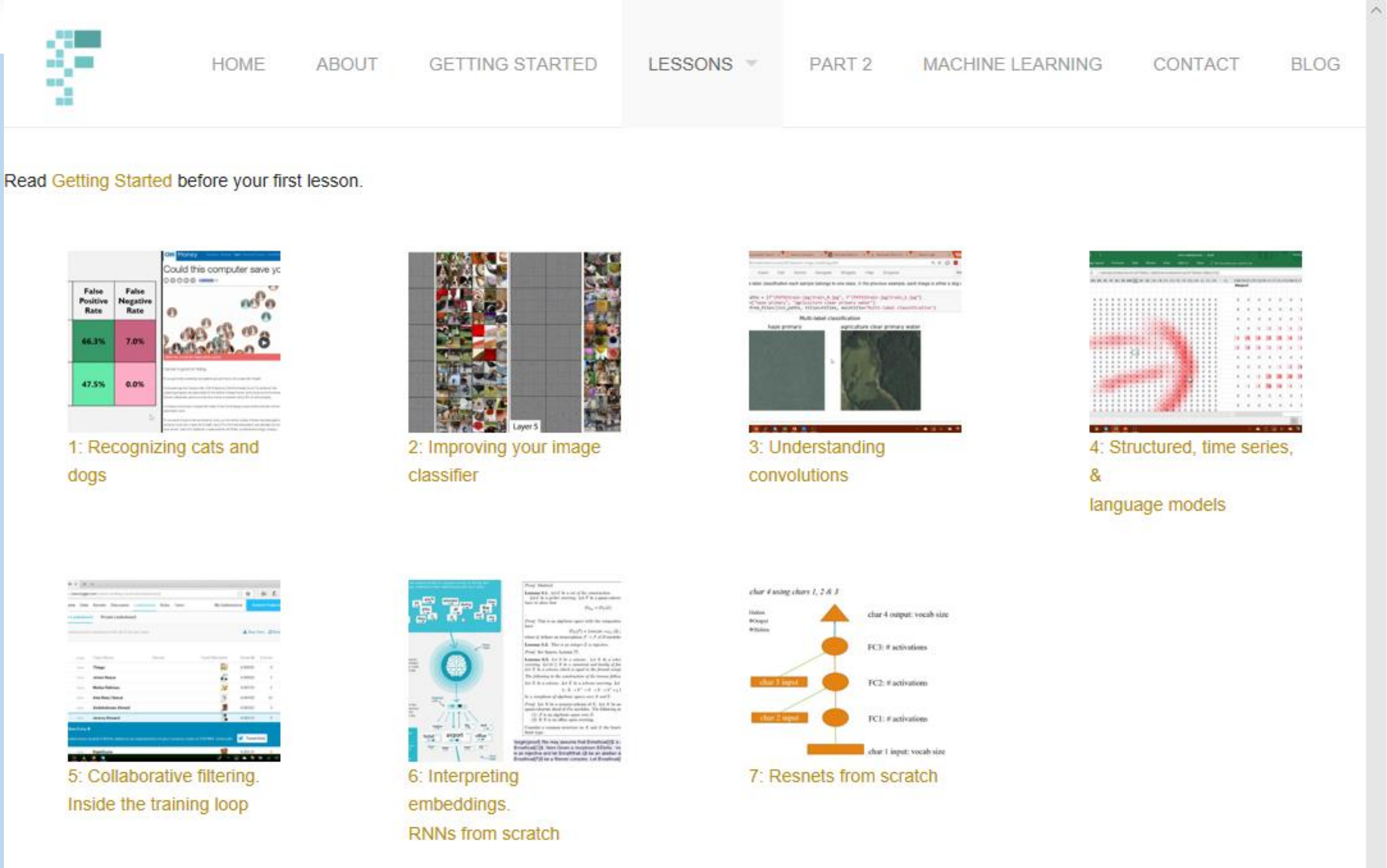
```
In [11]: preds = model.predict(x_test)
         plt.figure(figsize=(8,8))

         for i in range(16):
             plt.subplot(4, 4, 1 + i, xticks=[], yticks=[])
             img_id = np.random.randint(10000)
             im = x_test[img_id,::]
             plt.title(class_names[preds[img_id].argmax()])
             plt.imshow(im)
         plt.show()
```

Build Networks with High-level API

fast.ai

Making neural nets uncool again

https://www.fast.ai/



If you can code,
you can do deep learning.

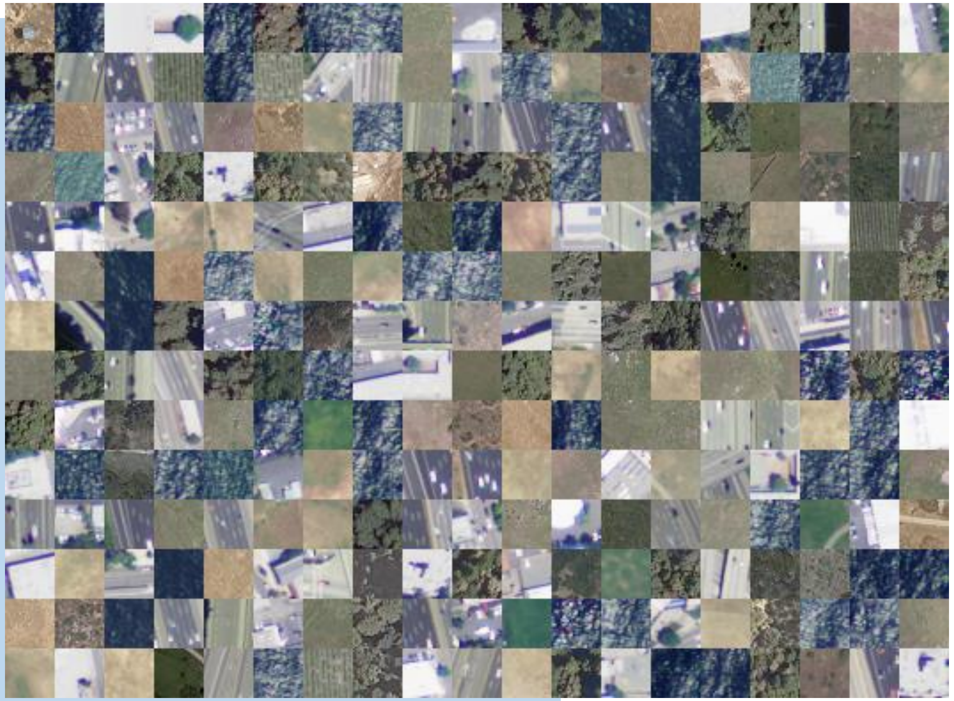
OUTLINES

1. Build Networks with High-level API
2. Classification with Convolutional Neural Networks
3. Object Detection with Bounding Box
4. Discussions

DeepSat (SAT-6) Airborne Dataset

405,000 image patches each of size 28x28 and covering 6 landcover classes

https://www.kaggle.com/crawford/deepsat-sat6



Kaggle is an online community of data scientists and machine learners, owned by Google, Inc. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges

Classifying Satellite Images

Prepare dataset

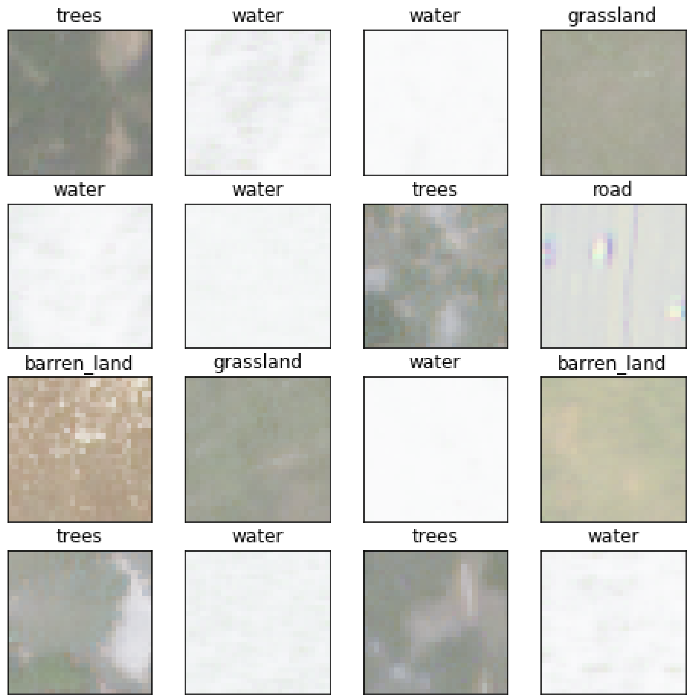
```
#require panadasy
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np

# Method to load data and images
def load_data_and_labels(data, labels):
    data_df = pd.read_csv(data, header=None)
    X = data_df.values.reshape((-1,28,28,4)).clip(0,255).astype(np.uint8)
    labels_df = pd.read_csv(labels, header=None)
    Y = labels_df.values.getfield(dtype=np.int8)
    return X, Y

data_dir = 'F:/deepsat_sat6'

x_train, y_train = load_data_and_labels(data='{}/X_train_sat6.csv'.format(data_dir),
                                        labels='{}/y_train_sat6.csv'.format(data_dir))
x_test, y_test = load_data_and_labels(data='{}/X_test_sat6.csv'.format(data_dir),
                                      labels='{}/y_test_sat6.csv'.format(data_dir))

print(pd.read_csv('{}/sat6annotations.csv'.format(data_dir), header=None))

# Print shape of all training, testing data and labels
# Labels are already loaded in one-hot encoded format
print('x_train_shape : {}'.format(x_train.shape)) # (324000, 28, 28, 4)
print('y_train_shape : {}'.format(y_train.shape)) # (324000, 6)
print('x_test_shape : {}'.format(x_test.shape))   # (81000, 28, 28, 4)
print('y_test_shape : {}'.format(y_test.shape))   # (81000, 6)
```

```
             0  1  2  3  4  5  6
0     building  1  0  0  0  0  0
1  barren_land  0  1  0  0  0  0
2        trees  0  0  1  0  0  0
3    grassland  0  0  0  1  0  0
4         road  0  0  0  0  1  0
5        water  0  0  0  0  0  1
x_train_shape : (324000, 28, 28, 4)
y_train_shape : (324000, 6)
x_test_shape : (81000, 28, 28, 4)
y_test_shape : (81000, 6)
```

Classifying Satellite Images

Check dataset

```python
import matplotlib.pyplot as plt
plt.figure(figsize=(8,8))
class_names = ['building','barren_land','trees',
               'grassland','road','water']

for i in range(16):
    plt.subplot(4, 4, 1 + i, xticks=[], yticks=[])
    img_id = np.random.randint(324000)
    im = x_train[img_id,::]
    plt.title(class_names[y_train[img_id].argmax()])
    plt.imshow(im)
plt.show()
```



Classifying Satellite Images

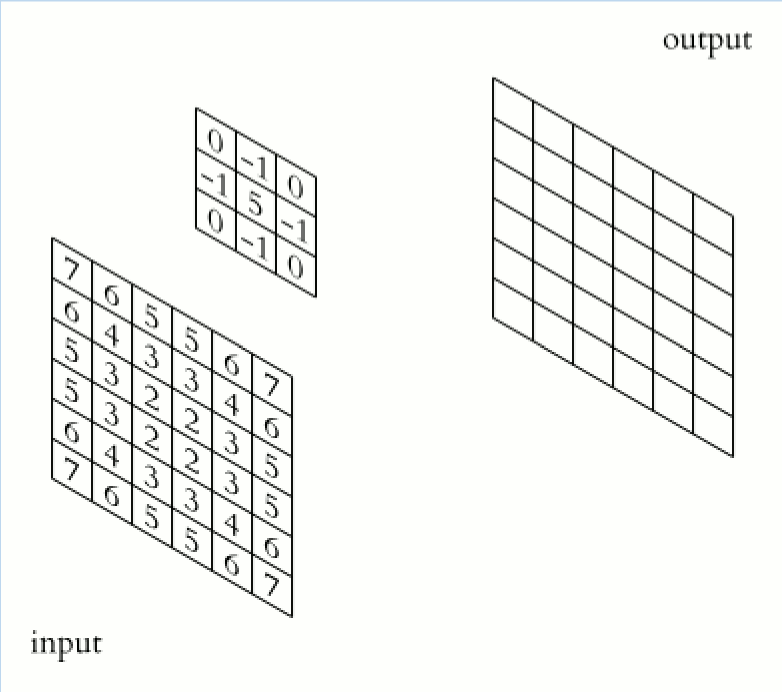
Create model

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Dropout, Flatten
from tensorflow.keras.models import Sequential
#Create model
model = Sequential()

#Add layers
model.add(Conv2D(16, (3,3), activation='relu', input_shape=(28,28,4)))
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.5))
model.add(Conv2D(32, (3,3), activation='relu'))
model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(6, activation='softmax'))

#configure model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

print(model.summary())
```

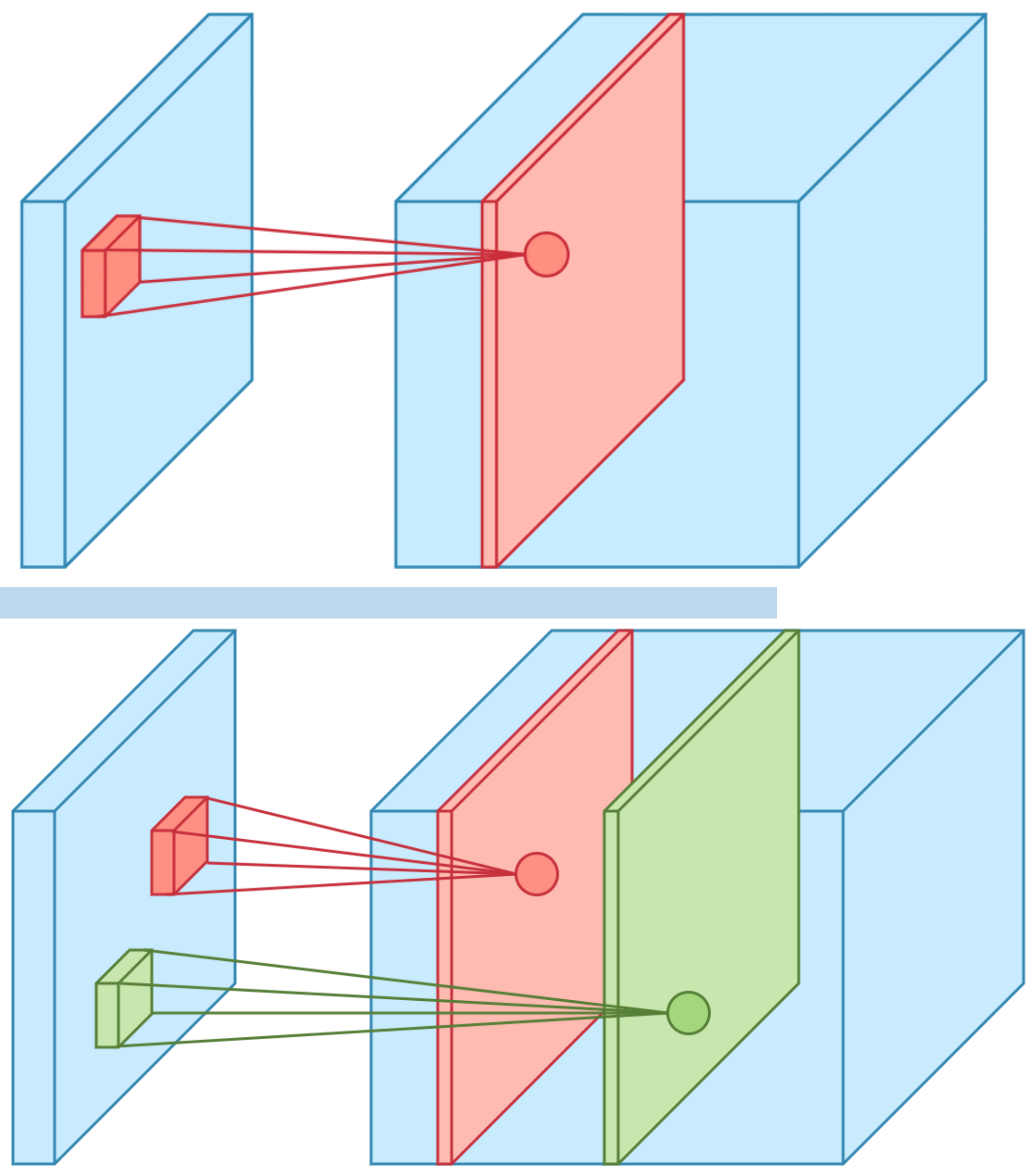
Classifying Satellite Images

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 16)        592
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 32)        4640
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 10, 10, 32)        9248
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 8, 8, 64)          18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 128)               131200
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 774
=================================================================
Total params: 164,950
Trainable params: 164,950
Non-trainable params: 0
_________________________________________________________________
None
```

Convolutional Layer

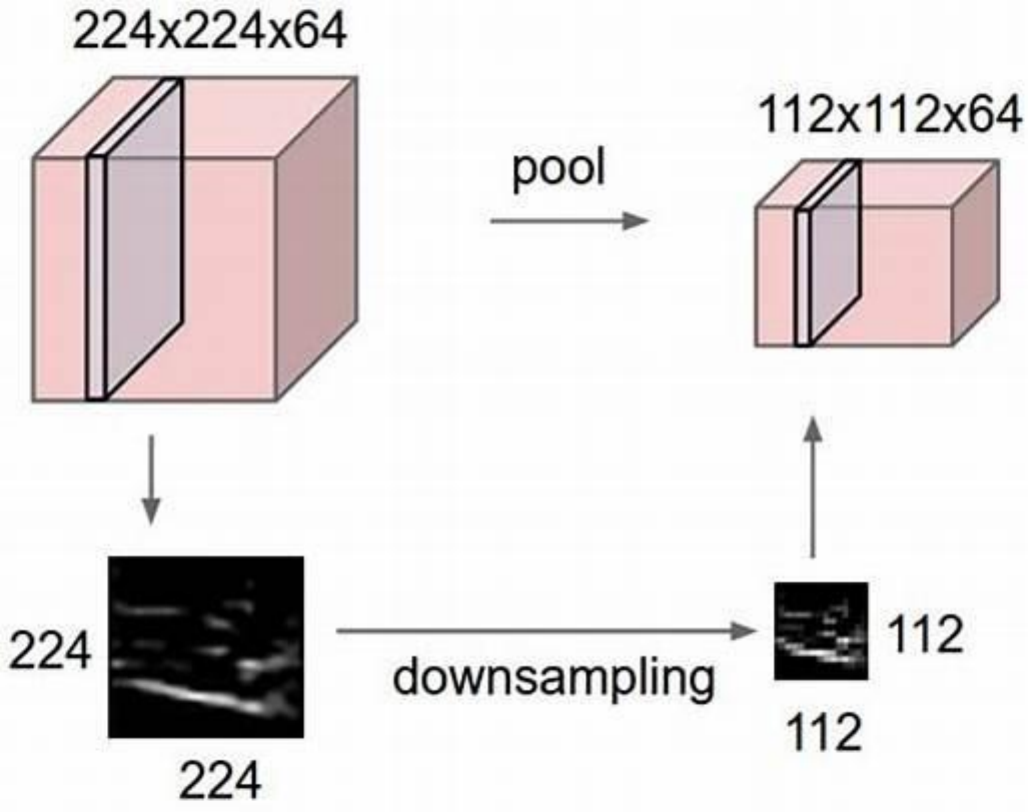


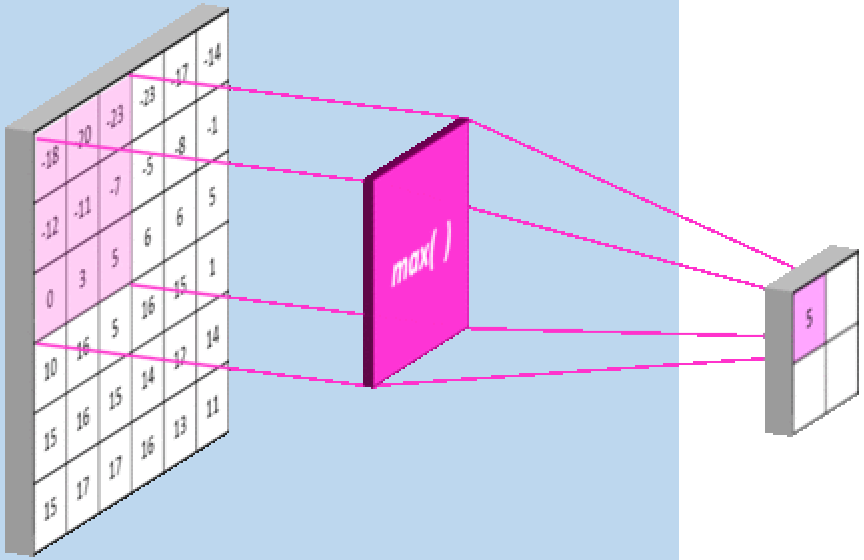
https://de.wikipedia.org/wiki/Convolutional_Neural_Network



https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Pooling Layer

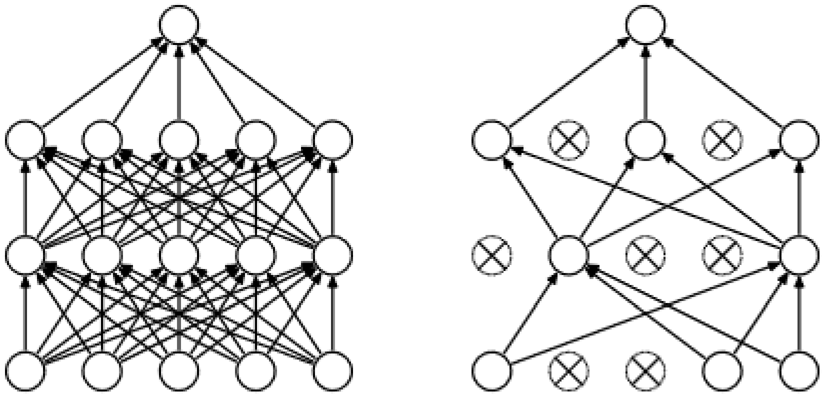




http://cs231n.github.io/convolutional-networks/#case

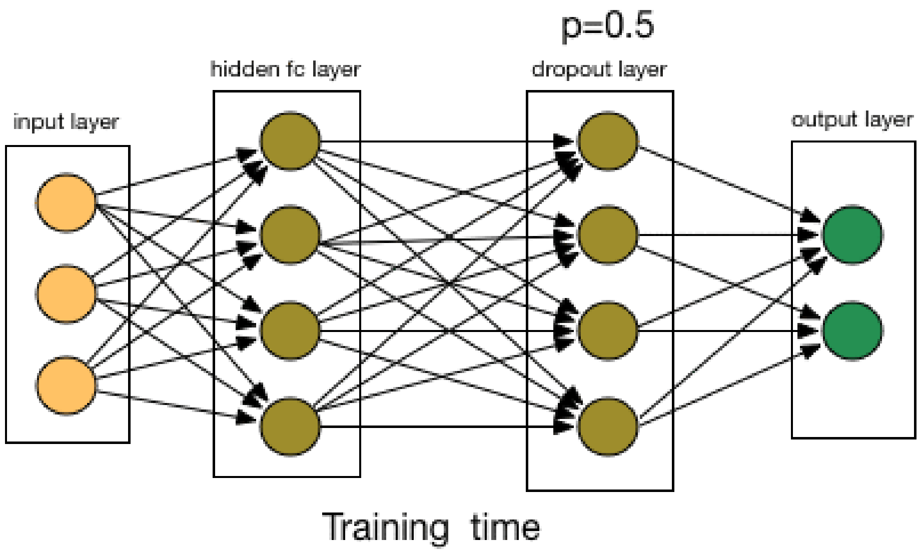
https://mlnotebook.github.io/post/CNN1/

Dropout



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.



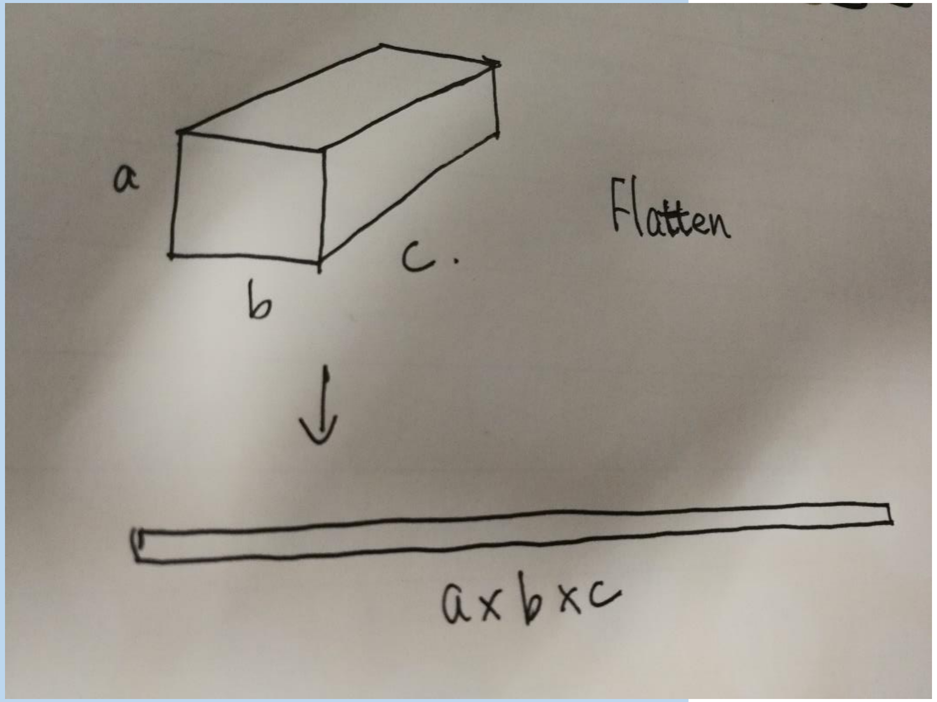
(Srivastava et al., 2014)

https://chatbotslife.com/regularization-in-deep-learning-f649a45d6e0

Flatten Layer

Example:

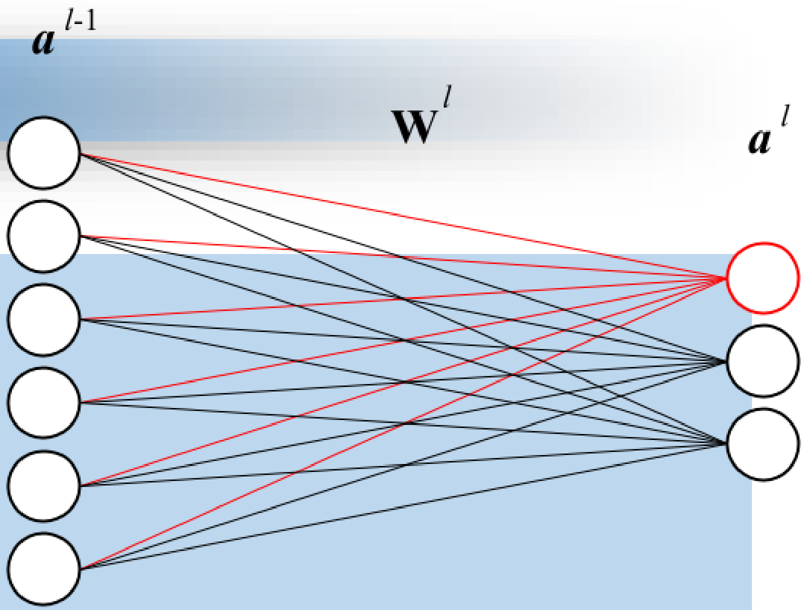
```
model = Sequential()
model.add(Convolution2D(64, 3, 3,
                        border_mode='same',
                        input_shape=(3, 32, 32)))
# now: model.output_shape == (None, 64, 32, 32)

model.add(Flatten())
# now: model.output_shape == (None, 65536)
```

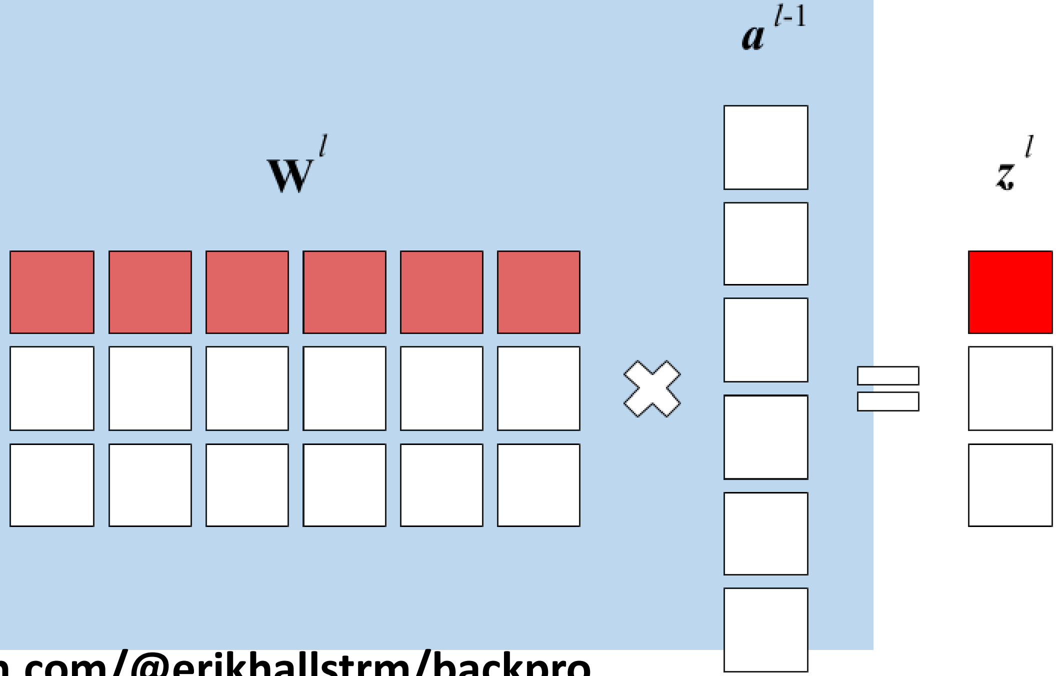


https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten

Densely(Fully) Connected Layer

$$\mathbf{a}^l = \sigma(\mathbf{W}^l \mathbf{a}^{l-1} + \mathbf{b}^l)$$





Images From
(https://medium.com/@erikhallstrm/backpropagation-from-the-beginning-77356edf427d)

Classifying Satellite Images

Train model

In [5]:
```
model.fit(x_train, y_train, batch_size=200,
          epochs=6, verbose=1)
```

```
Epoch 1/6
324000/324000 [==============================] - 23s 72us/step - loss: 0.4426 - acc: 0.9147
Epoch 2/6
324000/324000 [==============================] - 17s 54us/step - loss: 0.1044 - acc: 0.9625
Epoch 3/6
324000/324000 [==============================] - 17s 51us/step - loss: 0.0762 - acc: 0.9736
Epoch 4/6
324000/324000 [==============================] - 17s 52us/step - loss: 0.0676 - acc: 0.9767
Epoch 5/6
324000/324000 [==============================] - 17s 52us/step - loss: 0.0636 - acc: 0.9781
Epoch 6/6
324000/324000 [==============================] - 17s 52us/step - loss: 0.0597 - acc: 0.9797
```

Out[5]:
```
<tensorflow.python.keras.callbacks.History at 0x7f308842a438>
```

Test model

In [7]:
```
# test trained model.
scores = model.evaluate(x_test, y_test, verbose=1)

print('Test loss:', scores[0])
print('Test accuracy:', scores[1])
```

```
81000/81000 [==============================] - 6s 80us/step
Test loss: 0.1474194964328666
Test accuracy: 0.9246296296296296
```

Classifying Satellite Images

Check predictions

```
preds = model.predict(x_test)
plt.figure(figsize=(8,8))

for i in range(16):
    plt.subplot(4, 4, 1 + i, xticks=[], yticks=[])
    img_id = np.random.randint(81000)
    im = x_test[img_id,::]
    plt.title(class_names[preds[img_id].argmax()])
    plt.imshow(im)
plt.show()
```

OUTLINES

1. Build Networks with High-level API
2. Classification with Convolutional Neural Networks
3. Object Detection with Bounding Box
4. Discussions

Object Detection with Bounding Box

Bounding Box Example

In geometry, the **minimum** or smallest **bounding** or enclosing **box** for a point set (S) in N dimensions is the **box** with the smallest measure (area, volume, or hypervolume in higher dimensions) within which all the points lie.

(Wikipedia)

http://host.robots.ox.ac.uk/pascal/VOC/voc2006/examples/index.html

Object Detection with Bounding Box

SPPNet: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

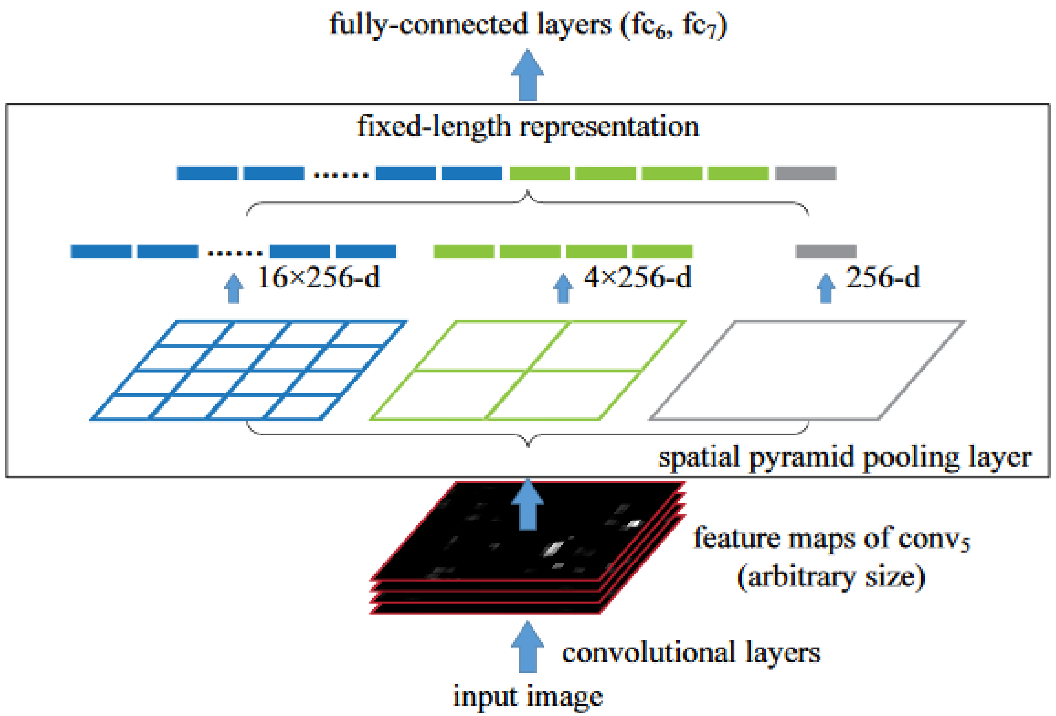
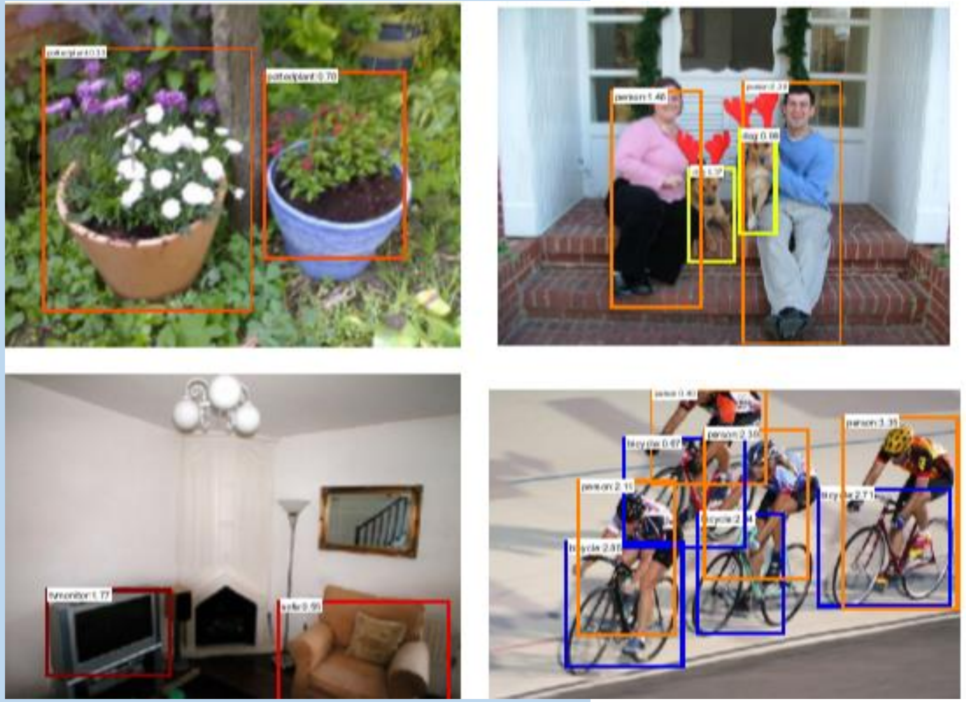


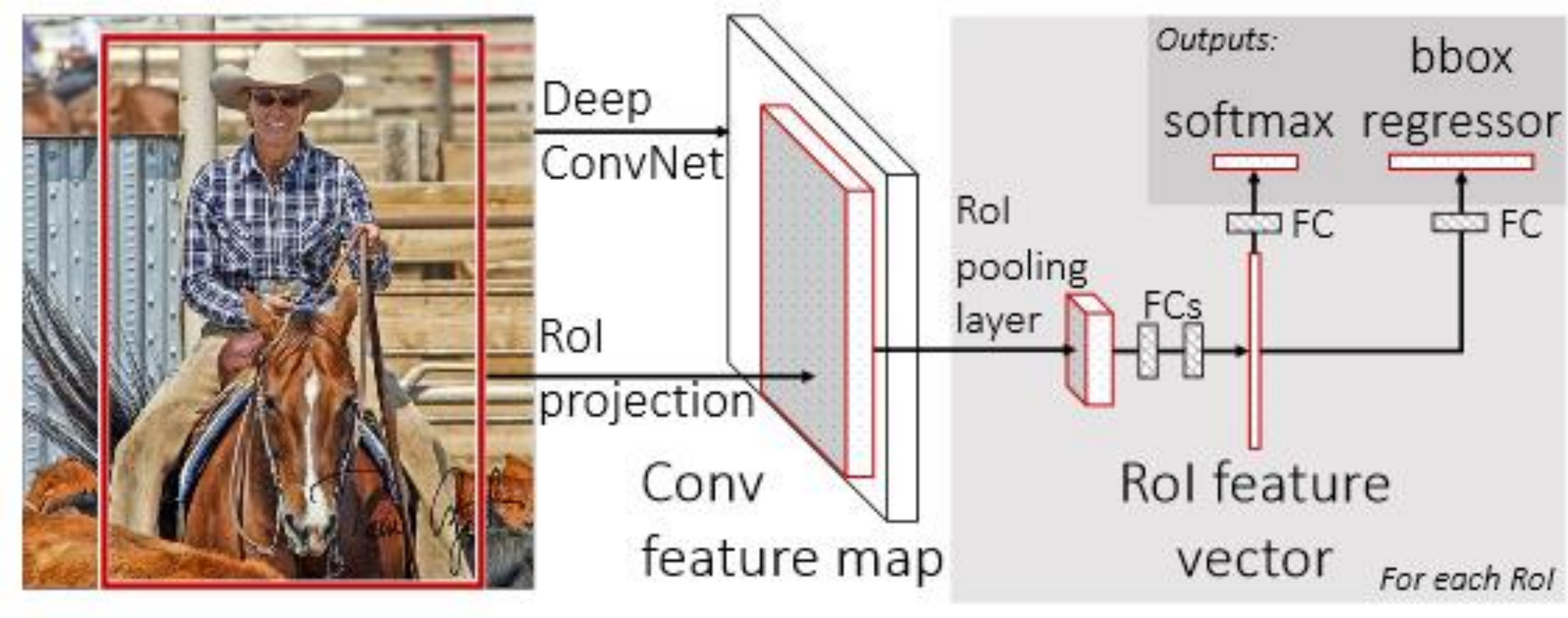
Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.



(He et al. 2014)

Object Detection with Bounding Box

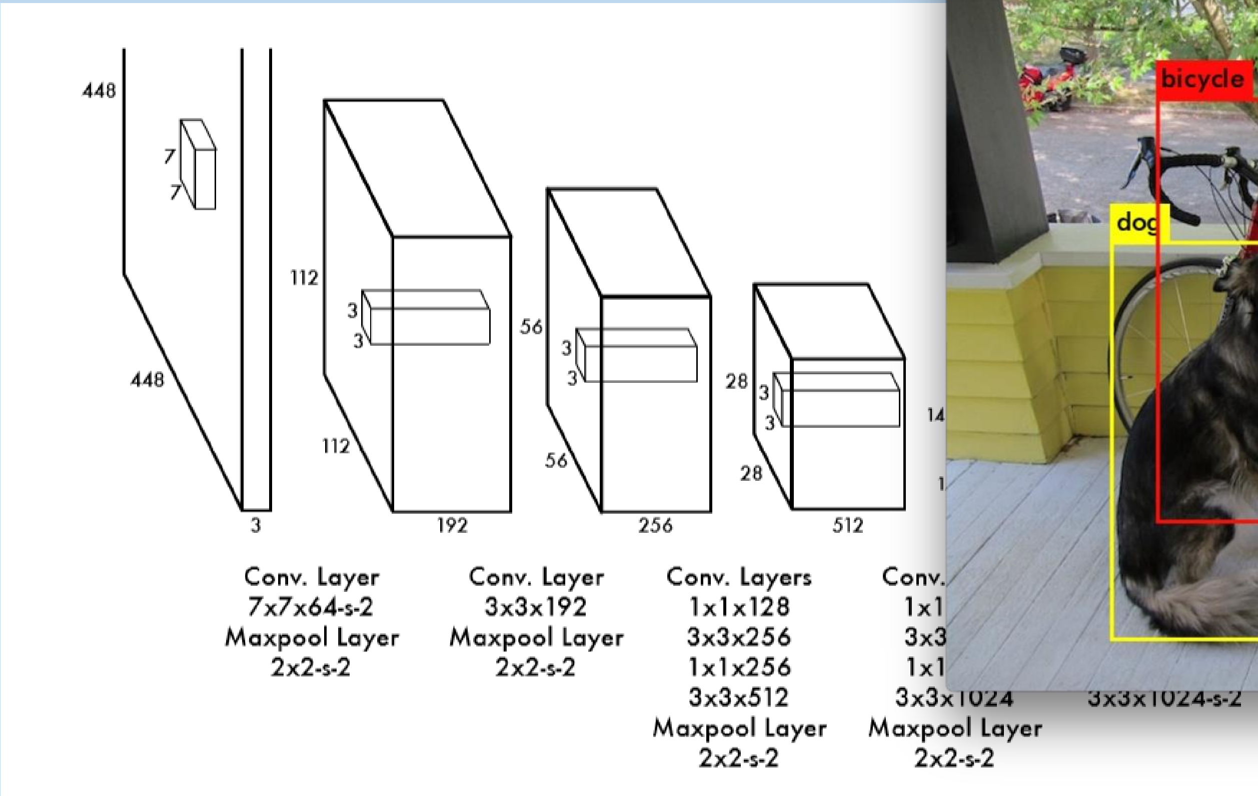
Fast-RCNN: Fast Region-based Convolutional Network method

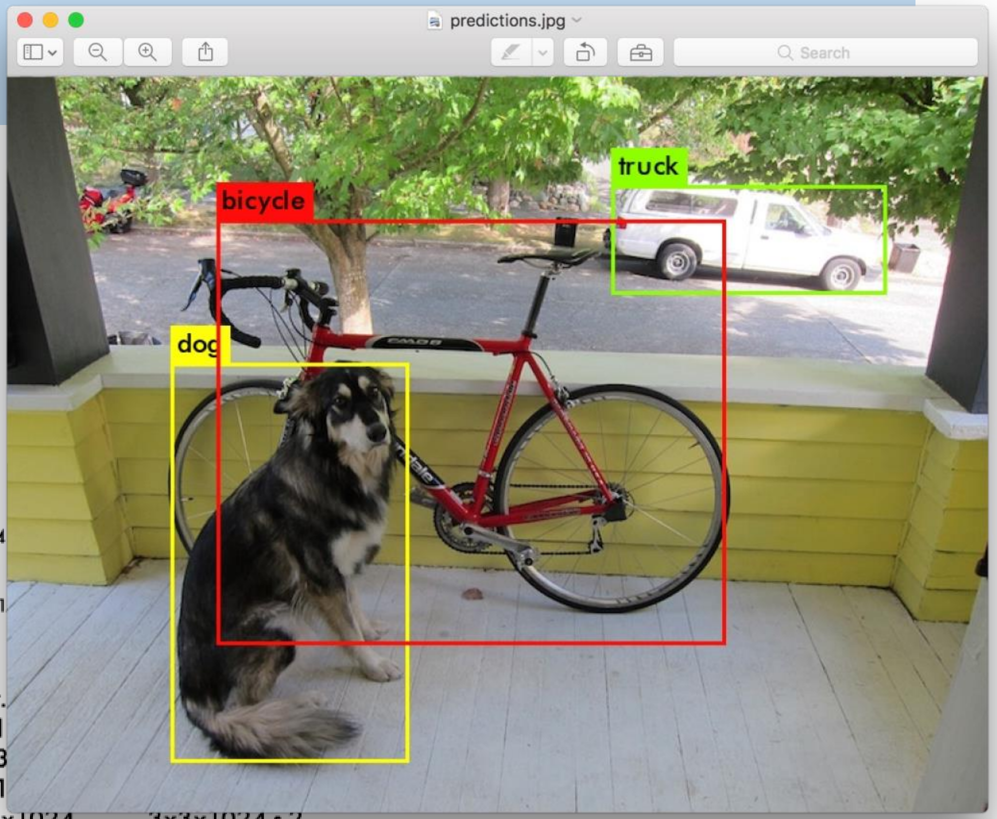


(Girshick, 2015)

Object Detection with Bounding Box

YOLO model: You Only Look Once



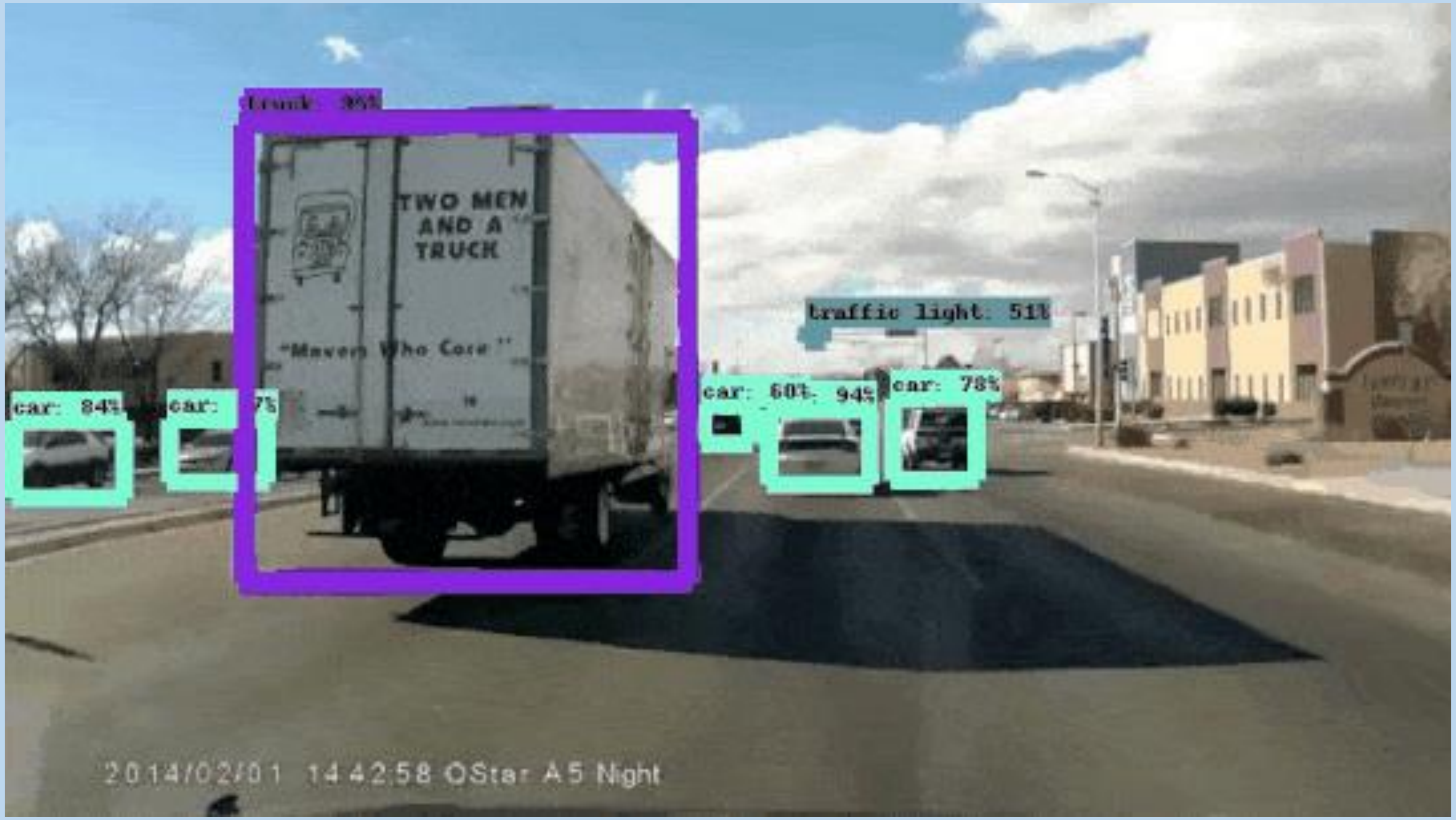


(Redmon et al. 2016)

Object Detection with Bounding Box

https://towardsdatascience.com/is-google-tensorflow-object-detection-api-the-easiest-way-to-implement-image-recognition-a8bd1f500ea0

Object Detection with Bounding Box



https://towardsdatascience.com/is-google-tensorflow-object-detection-api-the-easiest-way-to-implement-image-recognition-a8bd1f500ea0

OUTLINES

1. Build Networks with High-level API
2. Classification with Convolutional Neural Networks
3. Object Detection with Bounding Box
4. Discussions

Discussions

References

[1] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *arXiv:1406.4729 [cs]*, vol. 8691, pp. 346–361, 2014.

[3] R. Girshick, "Fast R-CNN," *arXiv:1504.08083 [cs]*, Apr. 2015.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788.

[5] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," p. 6.