Image Classification
And
Object Detection

Deep Learning in Earth Science
Lecture 2
By Xiao Zhuowei

For researchers interested in studying Earth science with deep learning.

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

OUTLINES

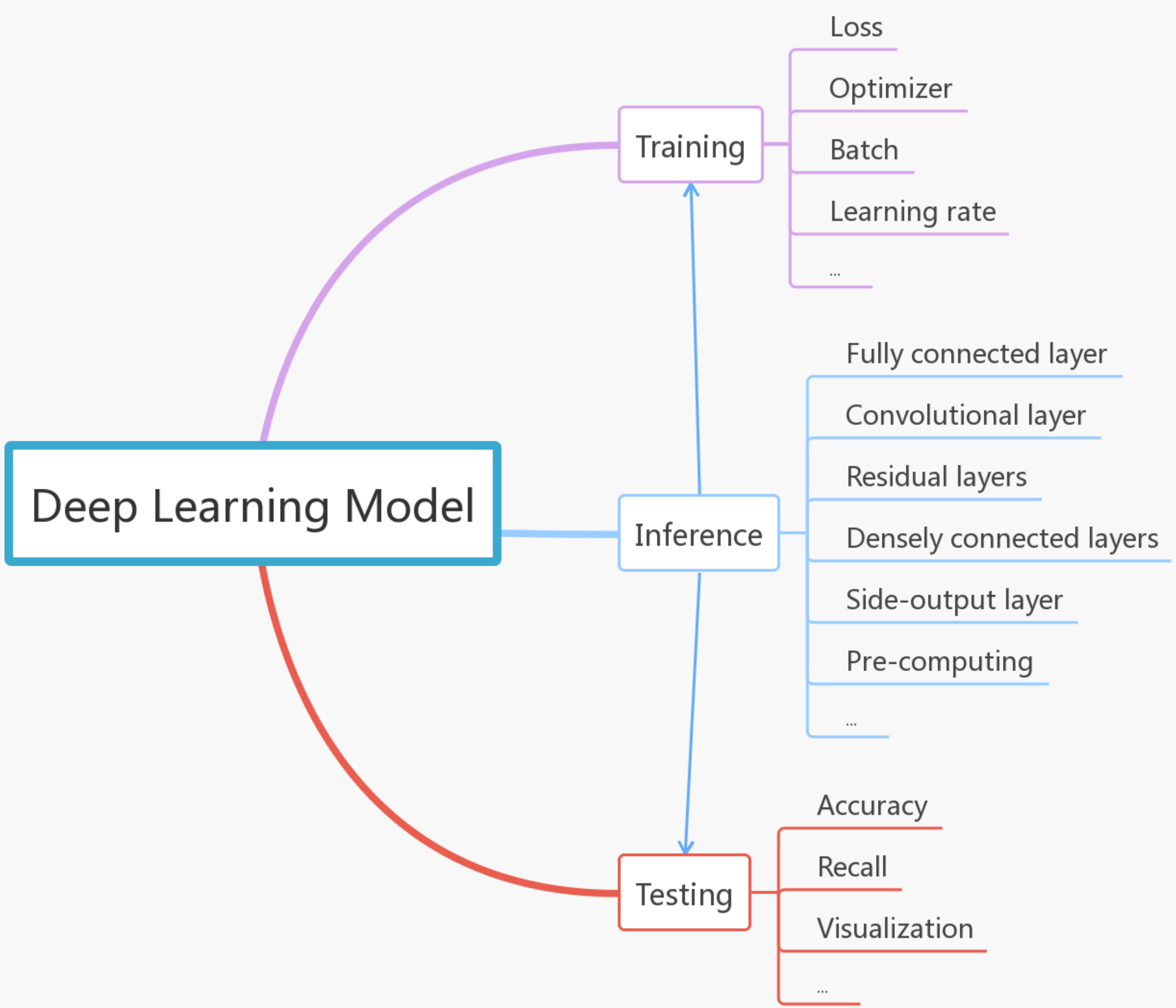
1 Build Networks with High-level API

2 Classification with Convolutional Neural Networks

3 Object Detection with Bounding Box

4 Discussions

Build Networks with High-level API



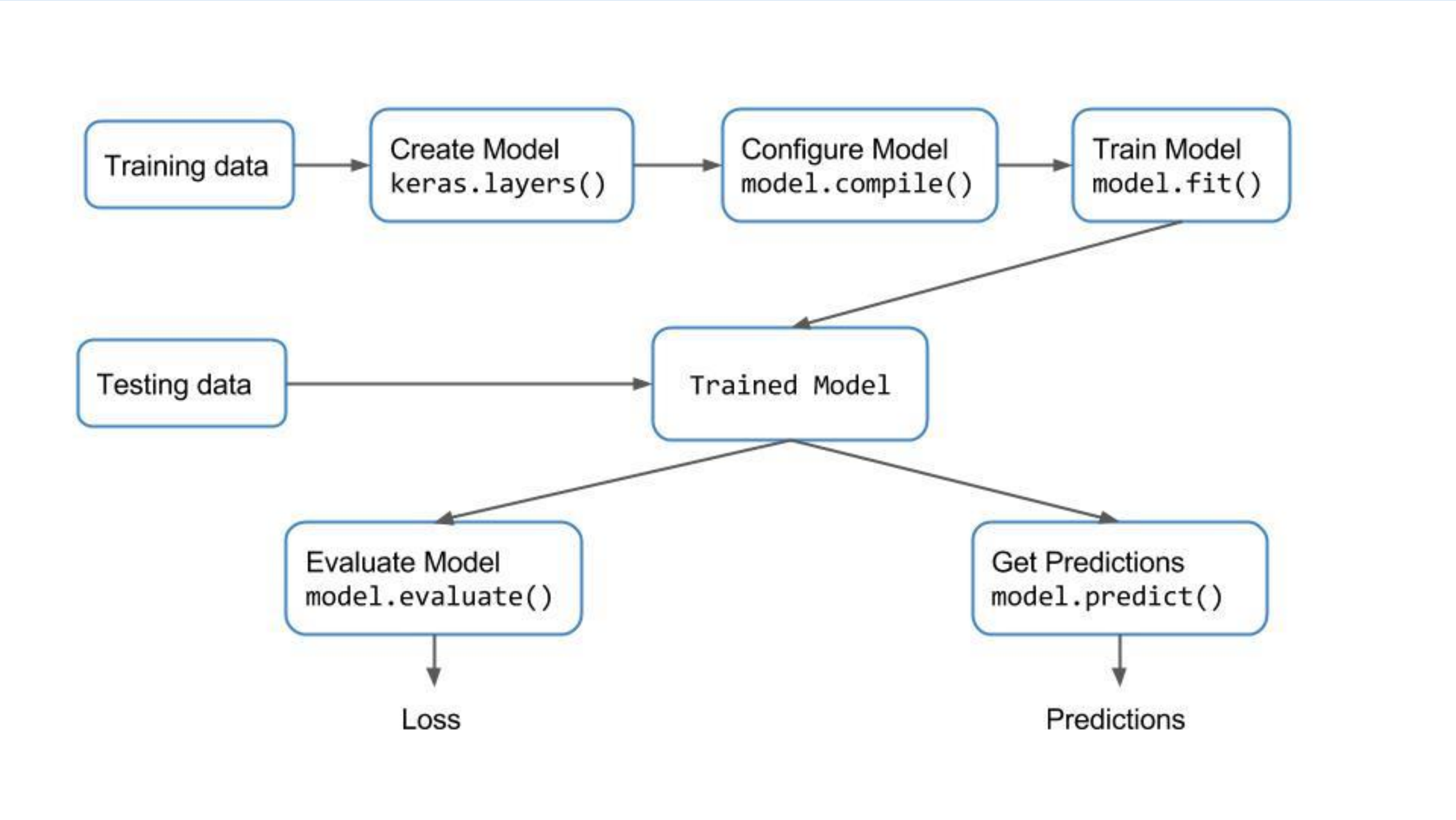
Keras

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

Create a model

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

Add layers

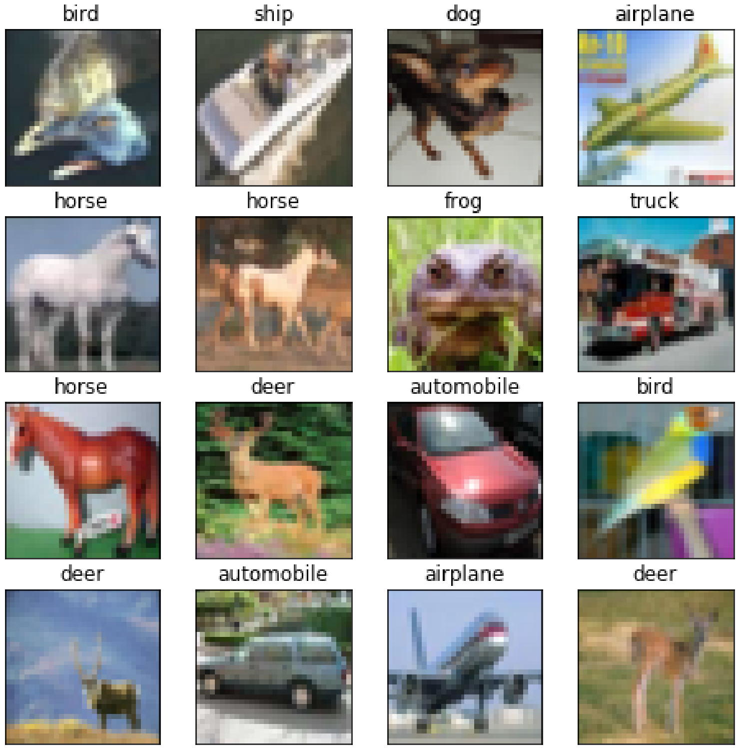
```
In [5]: from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
        from tensorflow.keras.layers import Conv2D, MaxPooling2D
        #You can also simply add layers via the .add() method
        model.add(Conv2D(32, (3, 3), padding='same',input_shape=x_train.shape[1:],
      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

Getting started with the Keras

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()
```



Build Networks with High-level API

Getting started with the Keras

Initiate optimizer

In [8]:
```
opt = tf.keras.optimizers.SGD(lr = 0.1, decay=1e-6, momentum=0.9 ,nesterov=True)
```

Configure model

In [9]:
```
#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'])
```

Train model

In [10]:
```
#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=100,
                    shuffle=True)
```

```
Epoch 1/100
  384/50000 [..............................] - ETA: 31:54 - loss: 2.3246 - acc: 0.1016
```

Build Networks with High-level API

Test model

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

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

Check predicts

```
In [ ]: 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(50000)
            im = x_test[img_id,::]
            plt.title(class_names[preds[img_id].argmax()])
            plt.imshow(im)
        plt.show()
```

Save model

```
In [ ]: # Save model and weights
        if not os.path.isdir(save_dir):
            os.makedirs(save_dir)

        model_path = os.path.join(save_dir, model_name)
        model.save(model_path)
        print('Saved trained model at %s ' % model_path)
```

Build Networks with High-level API

Test model

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

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

Check predicts

```
In [ ]: 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(50000)
            im = x_test[img_id,::]
            plt.title(class_names[preds[img_id].argmax()])
            plt.imshow(im)
        plt.show()
```

Save model

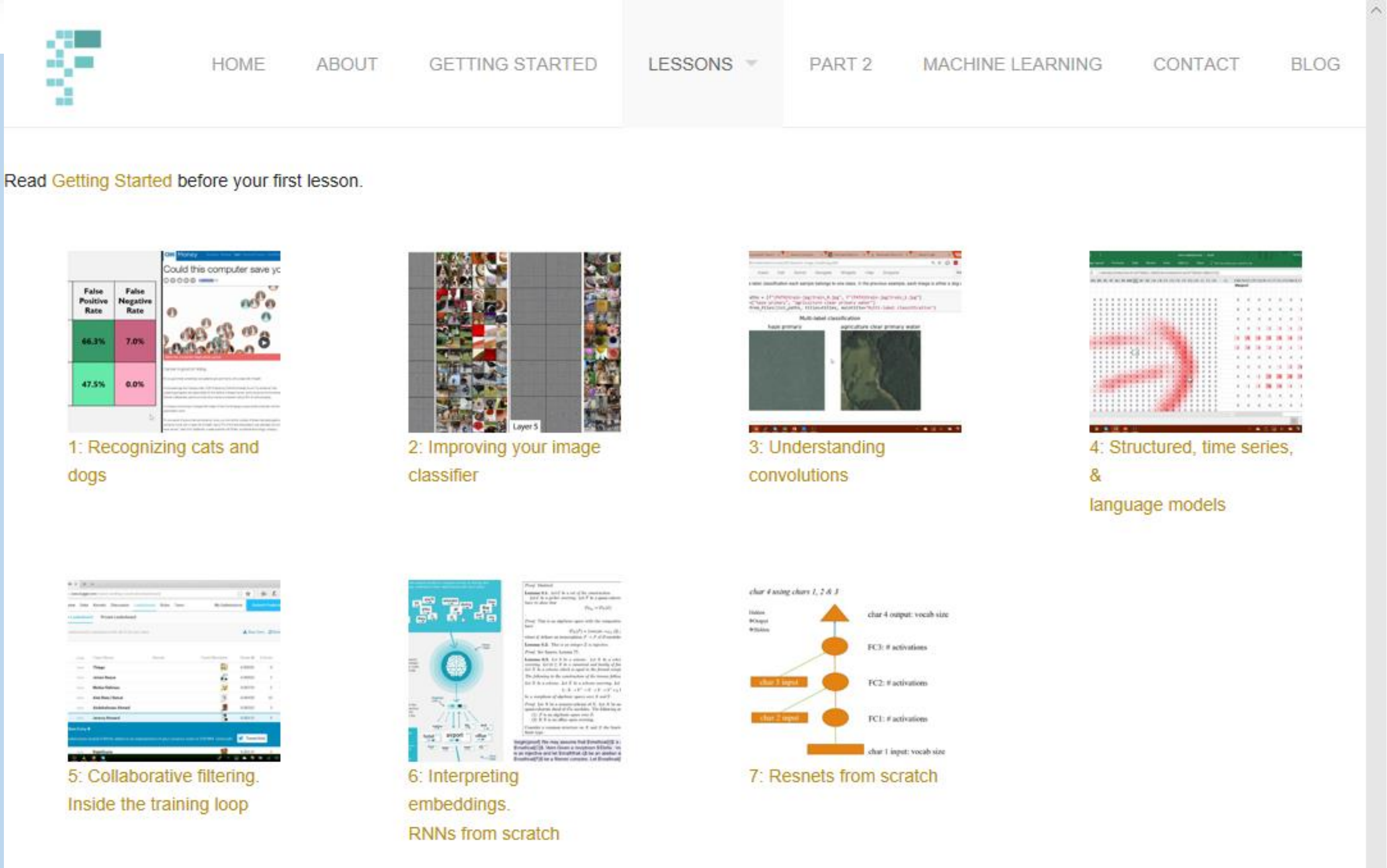
```
In [ ]: # Save model and weights
        if not os.path.isdir(save_dir):
            os.makedirs(save_dir)

        model_path = os.path.join(save_dir, model_name)
        model.save(model_path)
        print('Saved trained model at %s ' % model_path)
```

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

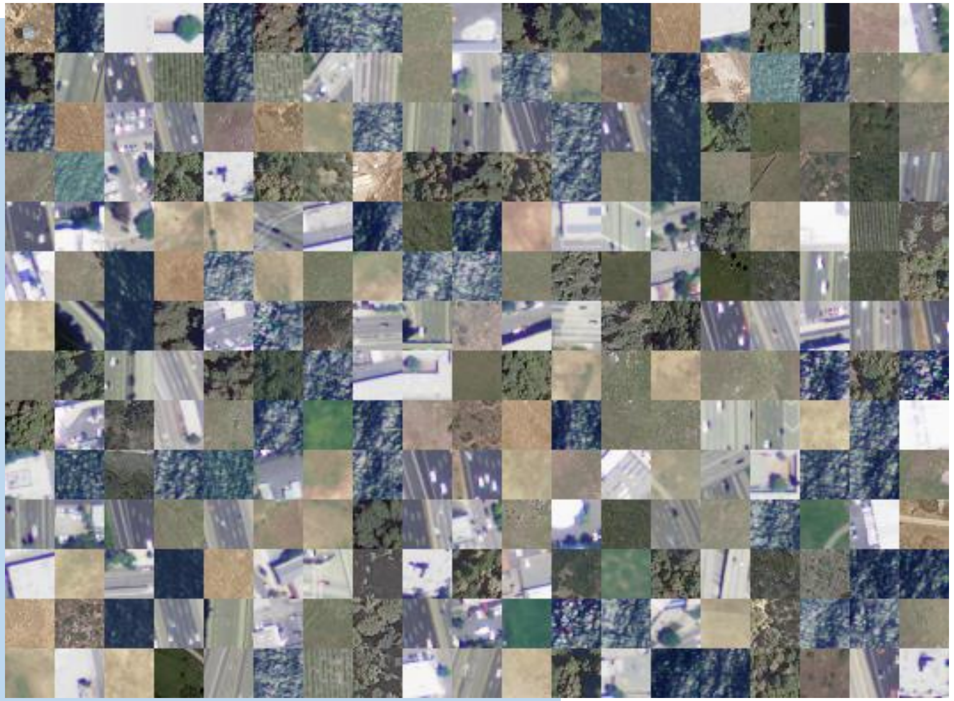
2 Classification with Convolutional Neural Networks

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

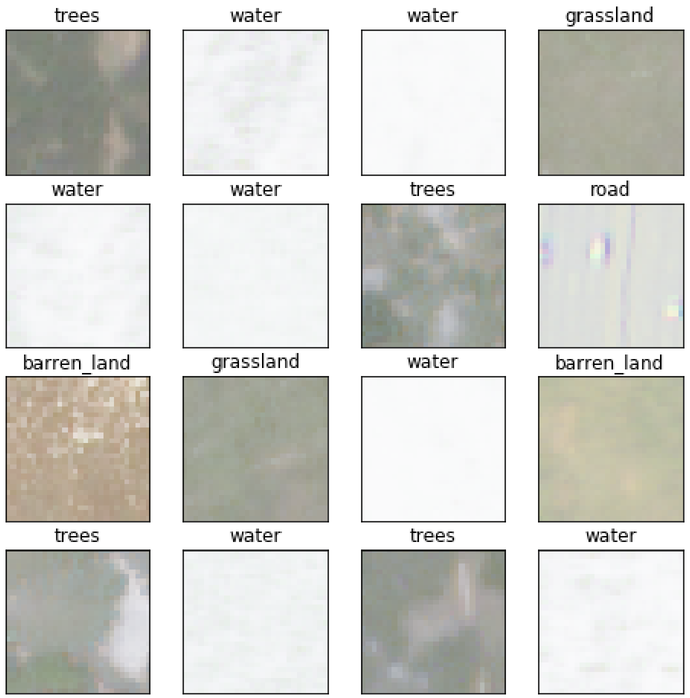
```
#require panadasy
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np

# 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

```
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()
```



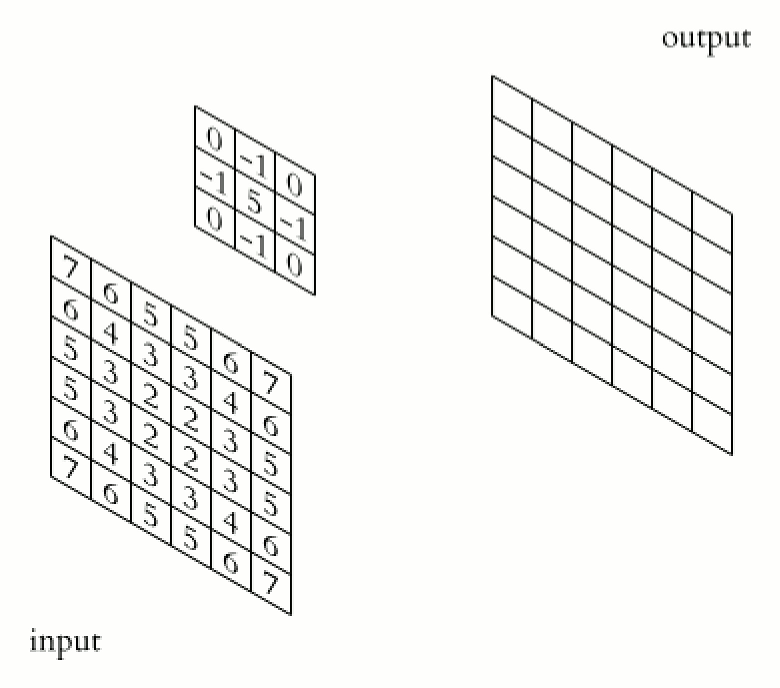
Classification with Convolutional Neural Networks

Classifying Satellite Images

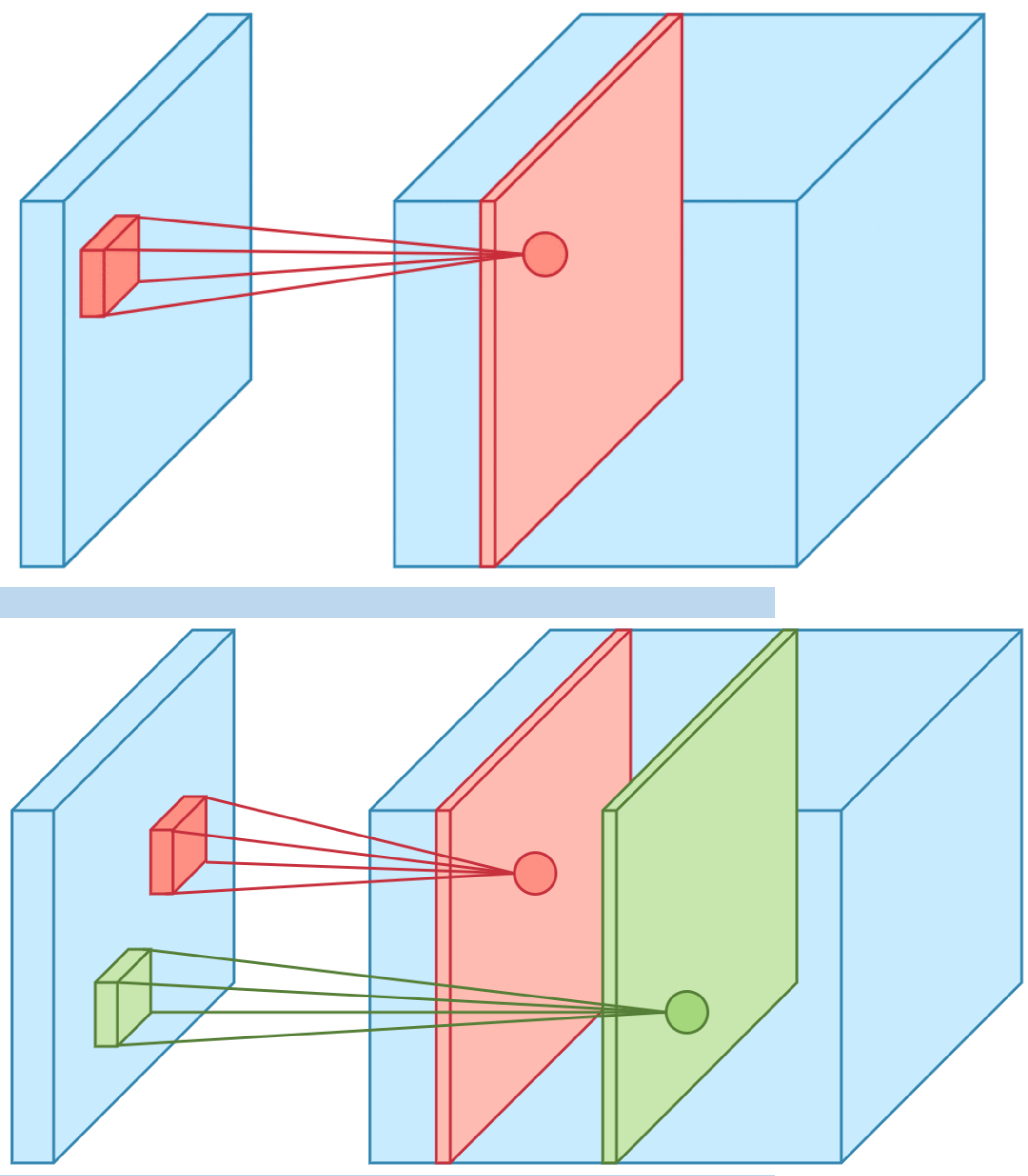
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 16)        592
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 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
```

Classification with Convolutional Neural Networks

Convolutional Layer

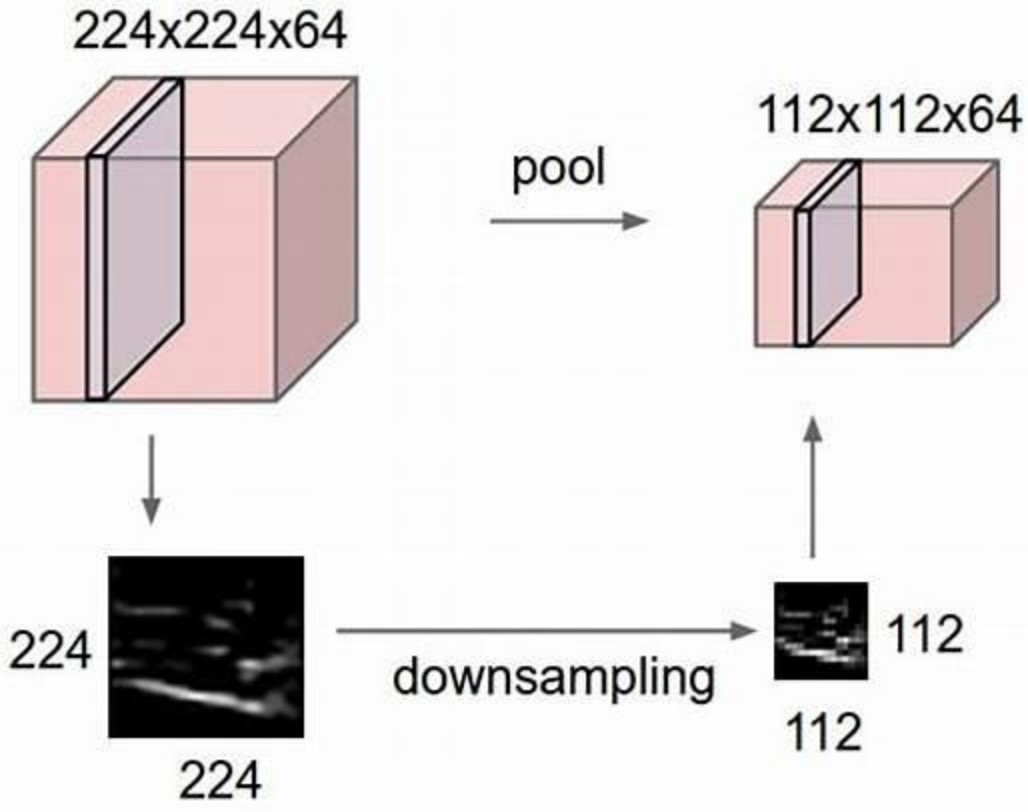


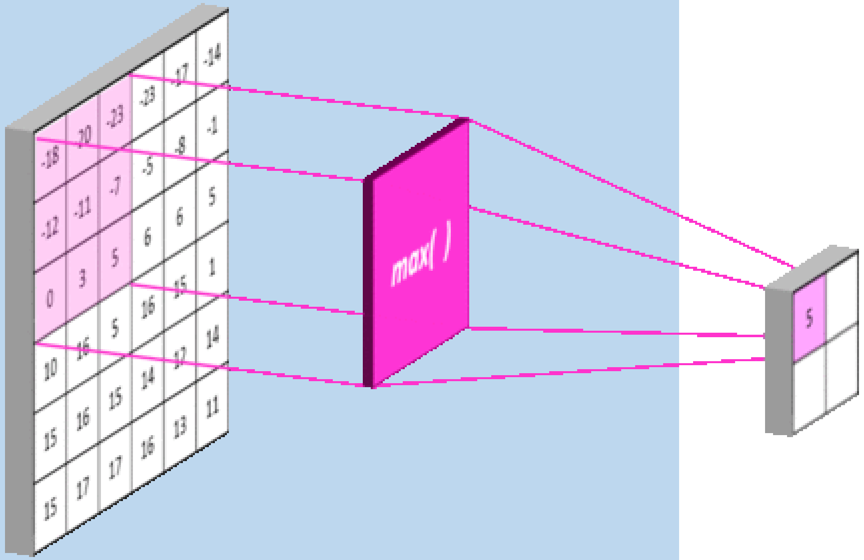
https://de.wikipedia.org/wiki/Convolutional_Neural_Network



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

Pooling Layer

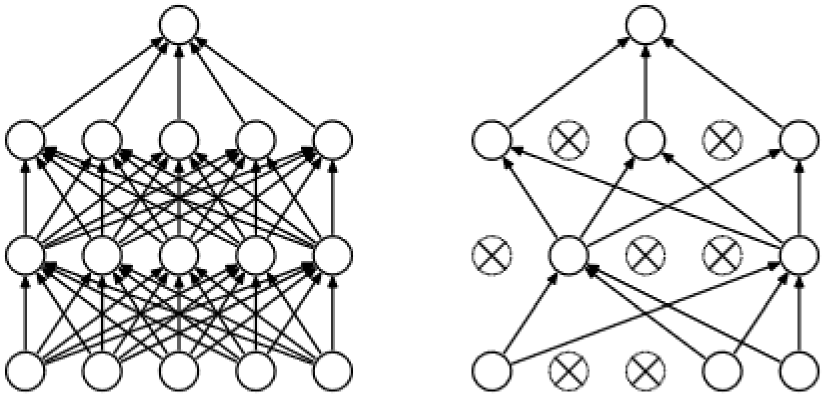




http://cs231n.github.io/convolutional-networks/#case

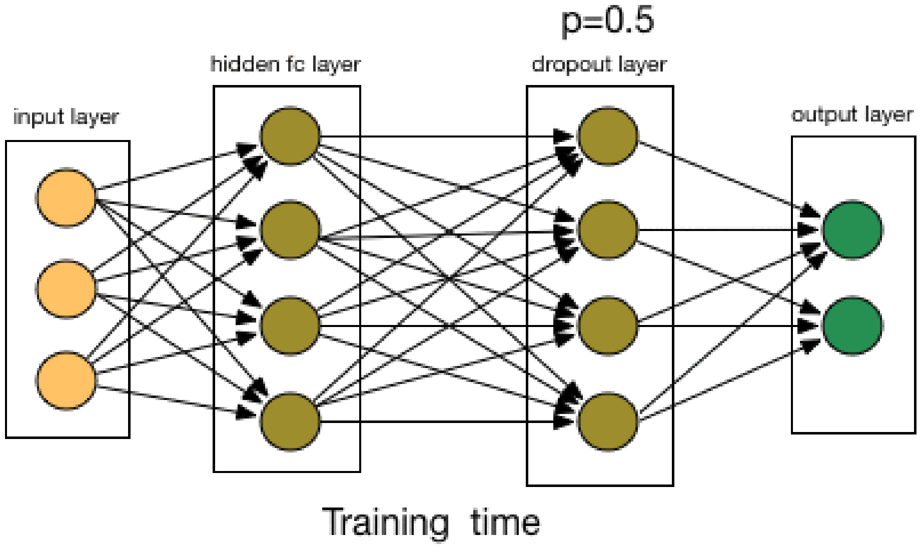
https://mlnotebook.github.io/post/CNN1/

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.



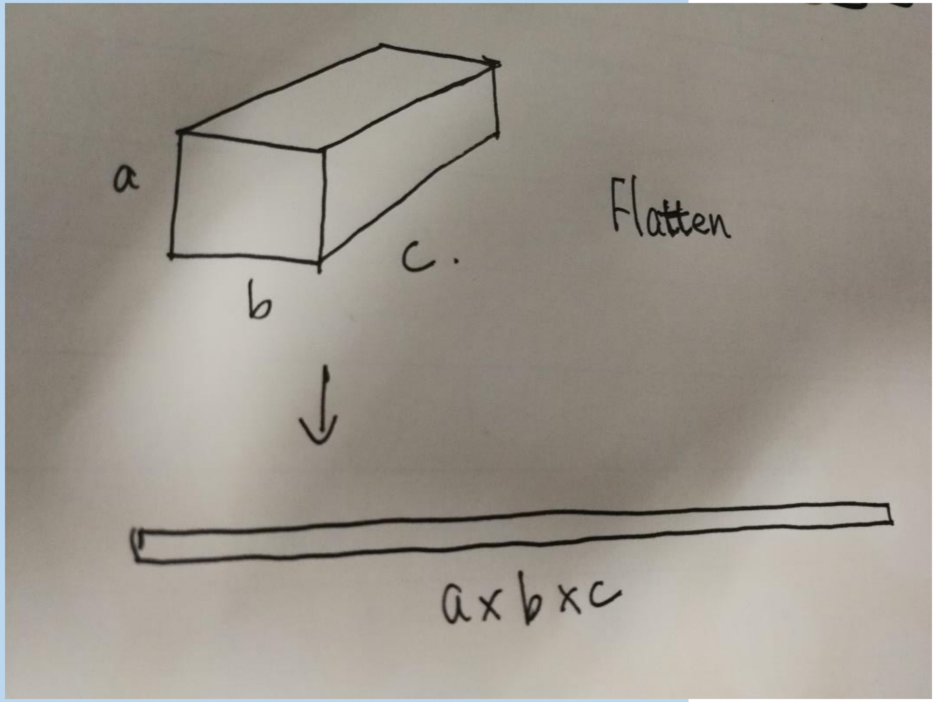
(Srivastava et al., 2014)

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

Flatten Layer

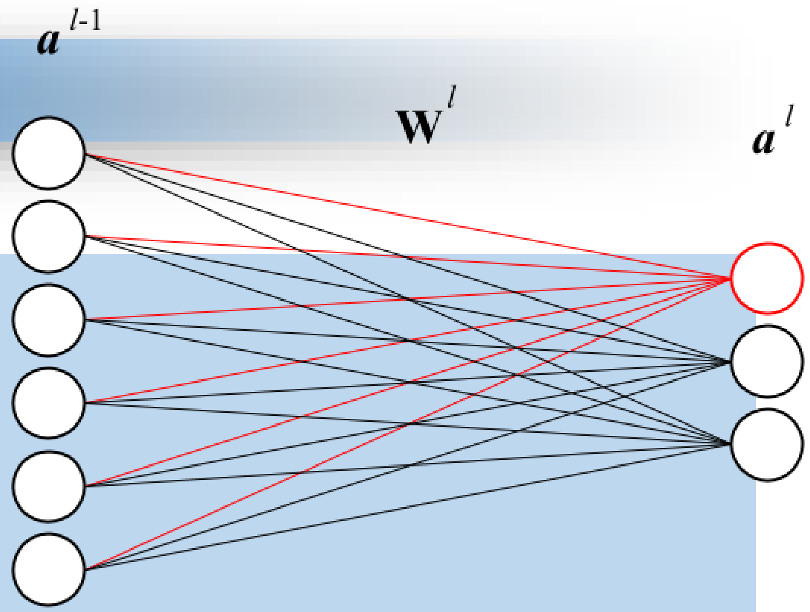
Example:

```
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)
```

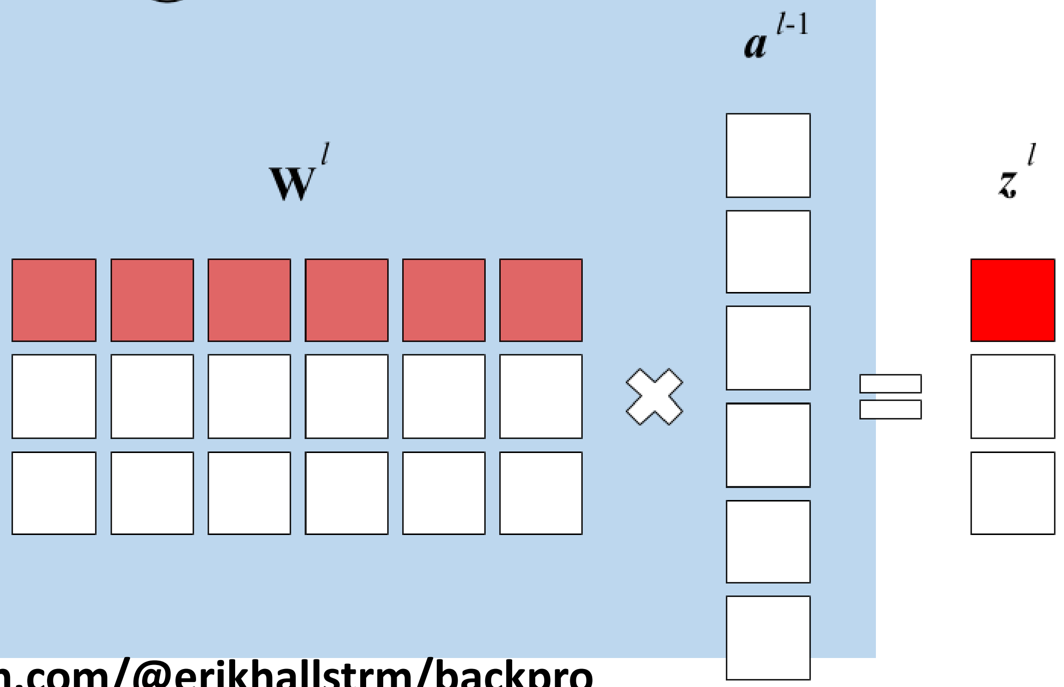


https://www.tensorflow.org/api_docs/python/tf/keras/layers/Flatten

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)

Train model

```
In [ ]: model.fit(X_train, y_train, batch_size=200,
                  epochs=6, verbose=1,
                  callbacks=[tbcallback])
```

Test model

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

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

Check predictions

```
In [ ]: 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(32400)
            im = X_test[img_id,::]
            plt.title(class_names[preds[img_id].argmax()])
            plt.imshow(im)
        plt.show()
```

Classifying Satellite Images

Train model

In [ ]:
```
model.fit(X_train, y_train, batch_size=200,
          epochs=6, verbose=1,
          callbacks=[tbcallback])
```

Test model

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

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

Check predictions

In [ ]:
```
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(32400)
    im = X_test[img_id,::]
    plt.title(class_names[preds[img_id].argmax()])
    plt.imshow(im)
plt.show()
```

Classifying Satellite Images

Train model

In [ ]:
```
model.fit(X_train, y_train, batch_size=200,
          epochs=6, verbose=1,
          callbacks=[tbcallback])
```

Test model

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

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

Check predictions

In [ ]:
```
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(32400)
    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

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

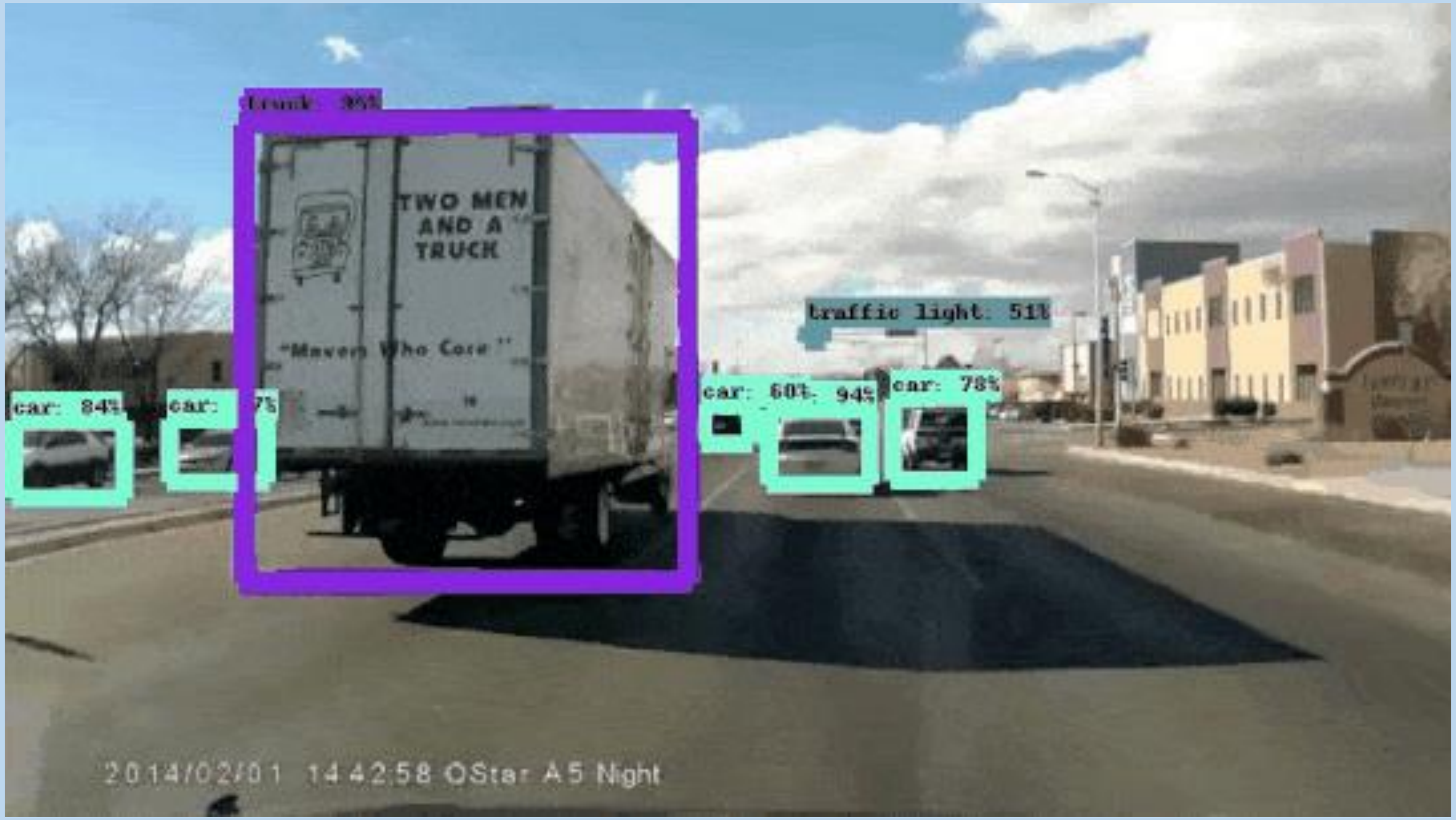
Object Detection with Bounding Box

https://pjreddie.com/darknet/yolo/

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