Convolutional Networks and Their Applications

Google AI ML Winter Camp
谷歌 AI 机器学习应用冬令营

Liangliang Cao
http://www.llcao.net
Staff Research Scientist, Google NYC

Many slides came from my class at Columbia University

Quick Survey

1. What is your career goal in 1-4 years?

(A) Ph.D. study in AI/ML
(B) Professor in universities
(C) Join big companies
(D) Join or co-found a startup
(E) None of Above

2. How can you use GPUs/TPUs?

(A) Google Cloud
(B) Other cloud service
(C) Resource from your university
(D) Resource from internship
(E) No GPUs yet or will build GPU machines by your own

Overview: Questions to discuss today

- How long did it take CNNs to become popular? Why?
- What can you do with CNNs? What can't do?
- What is the gap between "research" and "products"?

A glance of deep CNNs

- Convolutions are widely used for decades.
- The first popular deep CNN: LeNet in 1998
- The second popular deep CNN: AlexNet in 2012

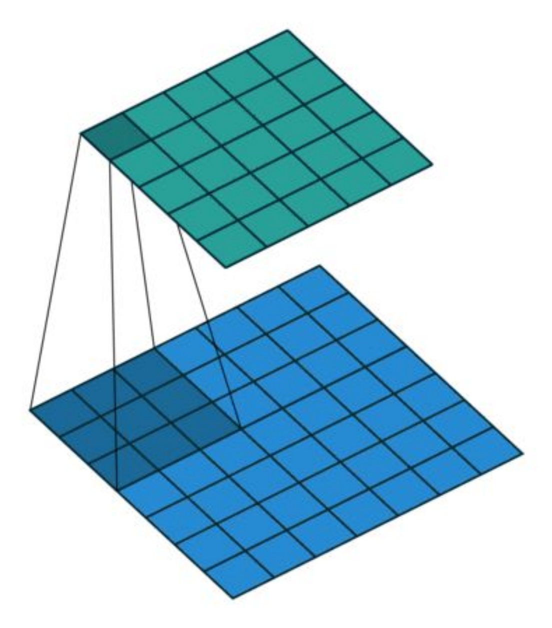
Why 14 years?

A brief history of convolutional filters

What is convolutional filter?

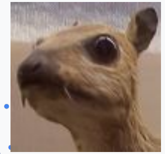
Image filtering are usually represented by the convolution between an image and a mask.

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot h[m-i, n-j]$$

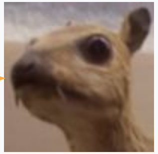


Usage of image filters

Image filtering are usually represented by the convolution between an image and a mask.



$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ →  Edge detection

$* \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ →  Blurring

$* \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ →  Sharpen

Usage of image filters

Image filtering are usually represented by the convolution between an image and a mask.

Image filters are powerful for many vision applications. We can use filters for recognition, synthesis, enhancement...

But convolutions are expensive

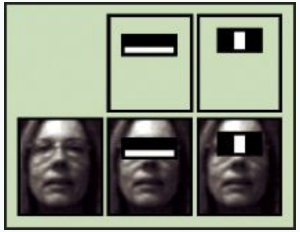
Despite its powerfulness for many vision applications, convolution is expensive: at every pixel we need do multi-multiplication with its neighborhood values

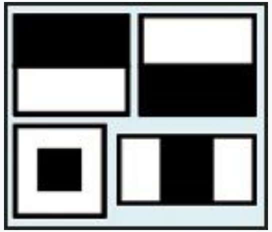
Algorithms for speedup: integral image, separable filters, convert time domain convolution to frequency multiplication...

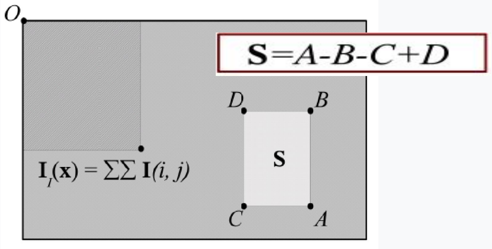
Hardware for speedup: GPU, TPU

An example of simple filters

Viola and Jones propose to learn face detection by selecting **millions of** simple filters called Harr filters.







Efficient computation:

Given adjacent rectangular regions,

- sums up the pixel intensities in each location
- calculates region integration via the difference

Convolutional filters for general tasks

Learning millions of filter or more

Modeling complexity over a larger neighborhood

- Both small and larger receptive fields (i.e., filter size)
- Stack multi convolutional layers together

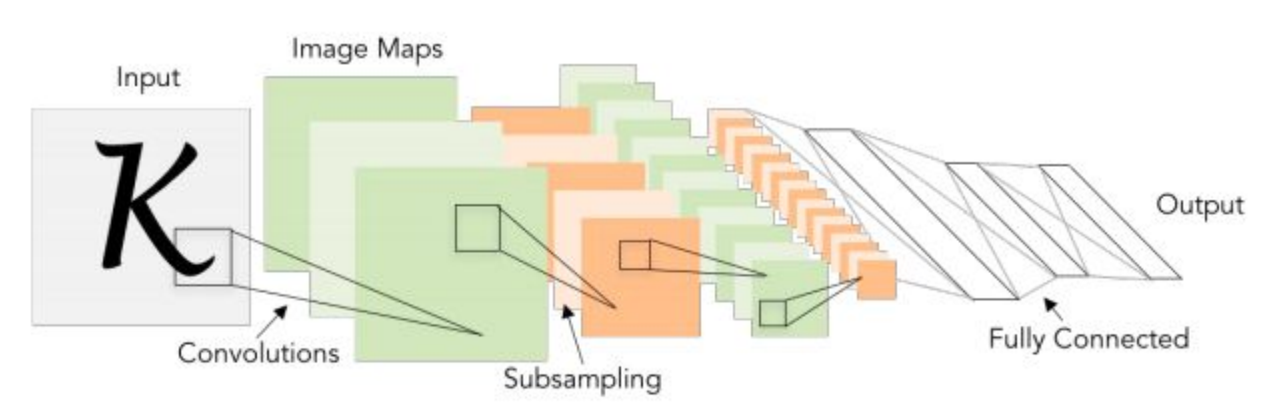
About two decade ago, some pioneers have realized the potential of such models:

Multi-layer CNNs (deep CNNs)

Google AI ML Winter Camp
谷歌 AI 机器学习应用冬令营

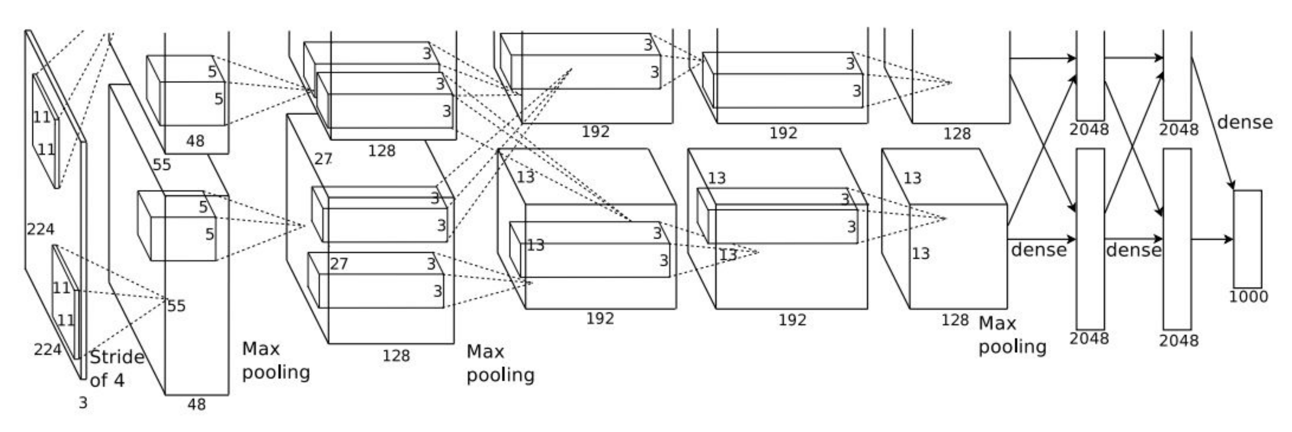
The first popular deep CNN

LeCun, Bottou, Bengio, Haffner, Gradient-based learning applied to document recognition, Proc. IEEE, 1998.



The second popular deep CNN

Krizhevsky, Sutskever, Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012



Why took 14 years?

1. People do not trust local minimum and are often annoyed by SGD failures.

Why took 14 years?

Which of the following will fail CNNs on MNIST?

- Use the raw pixel values between [0, 255]
- Initialize all the CNN weights as 0
- Use no intercept (i.e., Wx instead of Wx+b) in the fully connect layer
- The batch size is too small (i.e., one sample per batch)
- Use the whole dataset as one batch
- Do not shuffle the data before training

Will it fail CNNs (on MNIST)?

- “Use the raw pixel values between [0, 255]”

Correct. Almost all CNNs prefer to normalize pixel value normalized between [-1,1]

Will it fail CNNs (on MNIST)?

- “Initialize all the CNN weights as 0”

Correct. Network weights should be initialized randomly

Google AI ML Winter Camp
谷歌 AI 机器学习应用冬令营

Will it fail CNNs (on MNIST)?

- “Use no intercept (i.e., Wx instead of Wx+b) in the fully connect layer”

No. Network with zero intercepts will still work.

Will it fail CNNs (on MNIST)?

- “The batch size is too small (i.e., one sample per batch)”

  No.

- “The batch size is too big (i.e.,use the whole dataset as one batch)”

  Correct.

Will it fail CNNs (on MNIST)?

- “The batch size is too small (i.e., one sample per batch)”

  No. Small batch size will still work, but make the optimization slower

- “The batch size is too big (i.e.,use the whole dataset as one batch)”

  Correct. We will lose the “stochastic” factor by taking whole dataset as one batch, and the optimization will fall into bad local minimum.

Will it fail CNNs (on MNIST)?

- “Do not shuffle the data before training”

Usually correct. Random shuffling improves CNN a lot.

Why takes 14 years?

Which of the following will fail CNNs on MNIST?

- **Use the raw pixel values between [0, 255]**
- **Initialize all the CNN weights as 0**
- Use no intercept (i.e., Wx instead of Wx+b) in the fully connect layer
- The batch size is too small (i.e., one sample per batch)
- **Use the whole dataset as one batch**
- **Do not shuffle the data before training**

Why takes 14 years?

1. People do not trust local minimum and are often annoyed by SGD failures.

2. On MNIST CNN is not significant better than others

Model	Testing Error
KNN, subsample 16 x 16	1.1%
Boosted tree	1.53%
Non-linear SVM by LeCun'98	1.0%
Non-linear SVM by DeCoste'02	0.56%
2-layer MLP	2.45%
CNN LeNet-5	0.95%

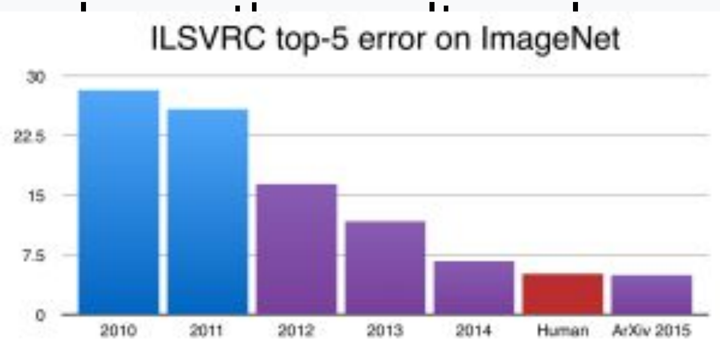
Results from http://yann.lecun.com/exdb/mnist/

Why takes 14 years?

1. People do not trust local minimum and are often annoyed by SGD failures.

2. On MNIST CNN is not significant better than others

3. Until pe … Net.



MNIST vs ImageNet

	MNIST	ImageNet LSVRC
Winner	SVMs	deep CNNs
Image size	28 x 28 x 1	224 x 224 x 3*
Num of images	60K	1,200K
Num of category	10	1000
In-class variation	small	large

*Resized size. Can be as large as 512 x 512

MNIST vs ImageNet

	MNIST	ImageNet LSVRC
Winner	SVMs	deep CNNs
Image size	28 x 28 x 1	224 x 224 x 3*
Num of images	60K	1,200K
Num of category	10	1000
In-class variation	small	large

SVMs → **More data** → **Bigger model** → **CNNs**

Small model → More data → Bigger model

Many product development follows such a pattern:

(1) annotate more and more data

(2) try different models

keep iterating (1) and (2) while deploy the current model to production

Let’s implement these popular CNNs

To implement LeNet is easy ...

- Download MNIST data and load them into memory
- Build a 5 layer CNN model

You can even do it on a laptop without GPUs

```
model = Sequential()
model.add(Conv2D(filters = 6, kernel_size = 5, strides = 1, activation = 'relu',
        input_shape = (32,32,1)))
model.add(MaxPooling2D(pool_size = 2, strides = 2))
model.add(Conv2D(filters = 16, kernel_size = 5, strides = 1, activation = 'relu',
        input_shape = (14,14,6)))
model.add(MaxPooling2D(pool_size = 2, strides = 2))
model.add(Flatten())
model.add(Dense(units = 120, activation = 'relu'))
model.add(Dense(units = 84, activation = 'relu'))
model.add(Dense(units = 10, activation = 'softmax'))
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
        ['accuracy'])
model.fit(X_train ,Y_train, steps_per_epoch = 10, epochs = 40)
```

Implement with Keras
a high level interface on top
of Tensorflow

But to implement AlexNet is hard ...

Alex Krizhevsky recalls his work of AlexNet on ImageNet:

“Ilya convinced me that with **an additional week** of effort, we could get equally good results on ImageNet. It actually took **five months** to match the 2010 state-of-the-art, and **several more months** to improve on it convincingly.”

“Time scales aside, his intuition was correct.”

But to implement AlexNet is hard ...

Challenges:

1. load 1.2M images into memory
2. do convolution via GPUs
3. two stream model using 2 GPUs

Ask yourself: how to handle these challenges if you were the chief architect?

Challenge 1
Load all ImageNet data into memory

Size of all images: 1.2M x 224 x 224 x 3 = 180G

Solution:

- use data iterator for Keras
  - easy to write, see example next page
- TF.data.Dataset for Tensorflow
  - hard to learn, but good for large scale

Challenge 1
Load all ImageNet data into memory

```
class NaiveImageNetIterator:
    def __init__(self, total_batches):
        self.ib, self.nb = 0, total_batches
    def __iter__(self):  return self
    def next(self): # Python 3: def __next__(self)
        if self.ib >= self.nb: raise StopIteration
        else:
            self.ib += 1
            return Load_Batch_from_Disk(self.ib)
```

Example use of data iterator:

```
data_iterator = NaiveImageNetIterator(120)
model.fit_generator(data_iterator, sample_per_epoch=1000)
```

Challenge 1
Load all ImageNet data into memory

Size of all images: 1.2M x 224 x 224 x 3 = 180G

Solution:

- use data iterator for Keras
  easy to write
- TF.data.Dataset for Tensorflow
  hard to learn,
  Tensorflow’s low level API
  Good for multi GPUs and TPUs

Challenge 2: Convolution via GPUs

It is not easy to write efficient GPU codes for convolution with different kernels.

See Alex's dizzying code
https://code.google.com/archive/p/cuda-convnet/

Fortunately now we can utilize Nvidia's Library
- cuBLAS
- cuDNN
- and also Google's TPU library

Challenge 3: Implement deep CNNs

Two approach

- Keras is easy to implement and understand

- Tensorflow estimator is more scalable

Google AI ML Winter Camp
谷歌 AI 机器学习应用冬令营

Challenge 3: Implement deep CNNs

```
# layer 1
alexnet.add(Conv2D(96, (11, 11),
        input_shape=img_shape, padding=valid',
        kernel_regularizer=l2(l2_reg)))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
alexnet.add(MaxPooling2D(strides=(4, 4)))

# layer 2
alexnet.add(Conv2D(256, (5, 5), padding='same'))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
alexnet.add(MaxPooling2D(pool_size=(2, 2)))
```

What is the number of para. in Layer 1

What is the output size of layer 1?

What is the number of para in layer 2?

What is the output size of layer 2?

Challenge 3: Implement deep CNNs

```
# layer 1
alexnet.add(Conv2D(96, (11, 11),
        input_shape=img_shape, padding=valid',
        kernel_regularizer=l2(l2_reg)))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
alexnet.add(MaxPooling2D(strides=(4, 4)))

# layer 2
alexnet.add(Conv2D(256, (5, 5), padding='same'))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
alexnet.add(MaxPooling2D(pool_size=(2, 2)))
```

What is the number of para. in Layer 1

- (11 x 11 x 3) * 96 = 35K

What is the output size of layer 1?

- (224-11) /4 = 55
- Output size (55 x 55 x 96)

What is the number of para in layer 2?

- (5 x 5 x 96) * 256 = 710K

What is the output size of layer 2?

- 55/2 = 27
- Output size (27 x 27 x 256)

Challenge 3: Implement deep CNNs

```
# layer 1
alexnet.add(Conv2D(96, (11, 11),
      input_shape=img_shape, padding=valid',
      kernel_regularizer=l2(l2_reg)))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
alexnet.add(MaxPooling2D(strides=(4, 4)))

# layer 2
alexnet.add(Conv2D(256, (5, 5), padding='same'))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
alexnet.add(MaxPooling2D(pool_size=(2, 2)))
```

```
# layer 3
alexnet.add(ZeroPadding2D((1, 1)))
      alexnet.add(Conv2D(512, (3, 3),
      padding='same'))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
alexnet.add(MaxPooling2D(pool_size=(2, 2)))

# layer 4
alexnet.add(ZeroPadding2D((1, 1)))
alexnet.add(Conv2D(1024, (3, 3), padding='same'))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
```

Challenge 3: Implement deep CNNs

```
# layer 5
alexnet.add(ZeroPadding2D((1, 1)))
alexnet.add(Conv2D(1024, (3, 3), padding='same'))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
alexnet.add(MaxPooling2D(pool_size=(2, 2)))

# layer 6
alexnet.add(Flatten())
alexnet.add(Dense(3072))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
alexnet.add(Dropout(0.5))
```

```
# layer 7
alexnet.add(Dense(4096))
alexnet.add(BatchNormalization())
alexnet.add(Activation('relu'))
alexnet.add(Dropout(0.5)) #

# layer 8
alexnet.add(Dense(n_classes))
alexnet.add(BatchNormalization())
alexnet.add(Activation('softmax'))
```

Challenge 3: Implement deep CNNs

Use Tensorflow estimator:

- TF Estimator can use Keras' layers
- TF Estimator can be scaled to large scale

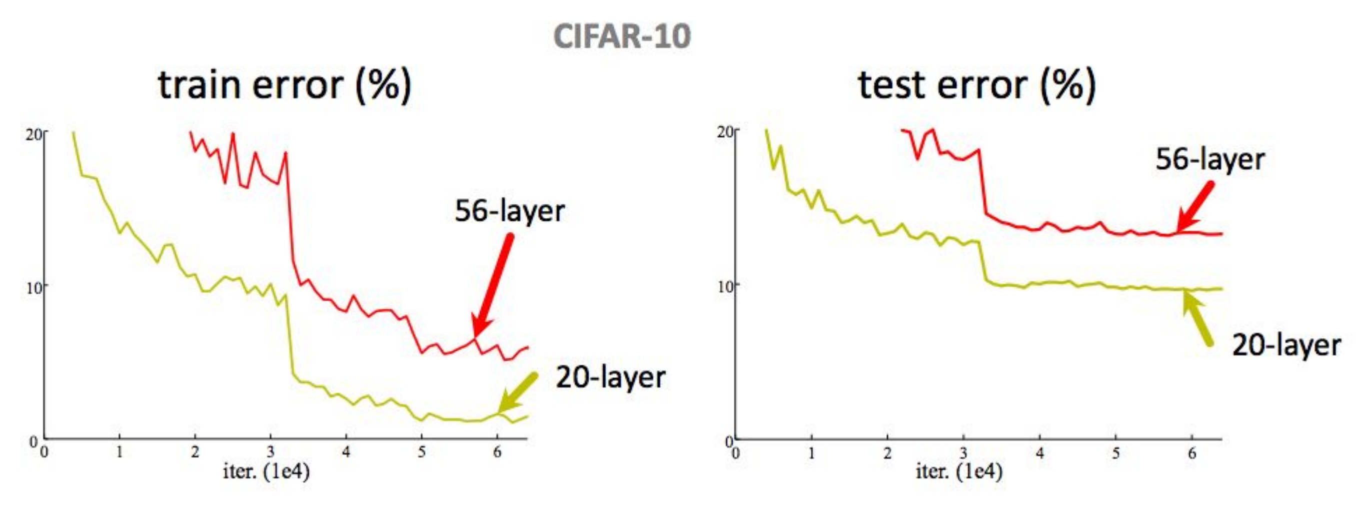
Challenge 3: Implement deep CNNs

How to imrove further?

- Smaller receptive fields, more filters, with more layers (see Matt Zeiler Network and VggNet)
- Concatenate multiple size of filters (see GoogLeNet)
- Residual Network

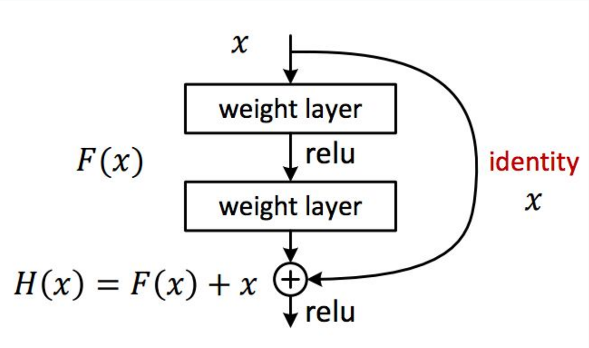
ResNet

Problem: Stacking more layers do not always improve performance.



ResNet

Solution



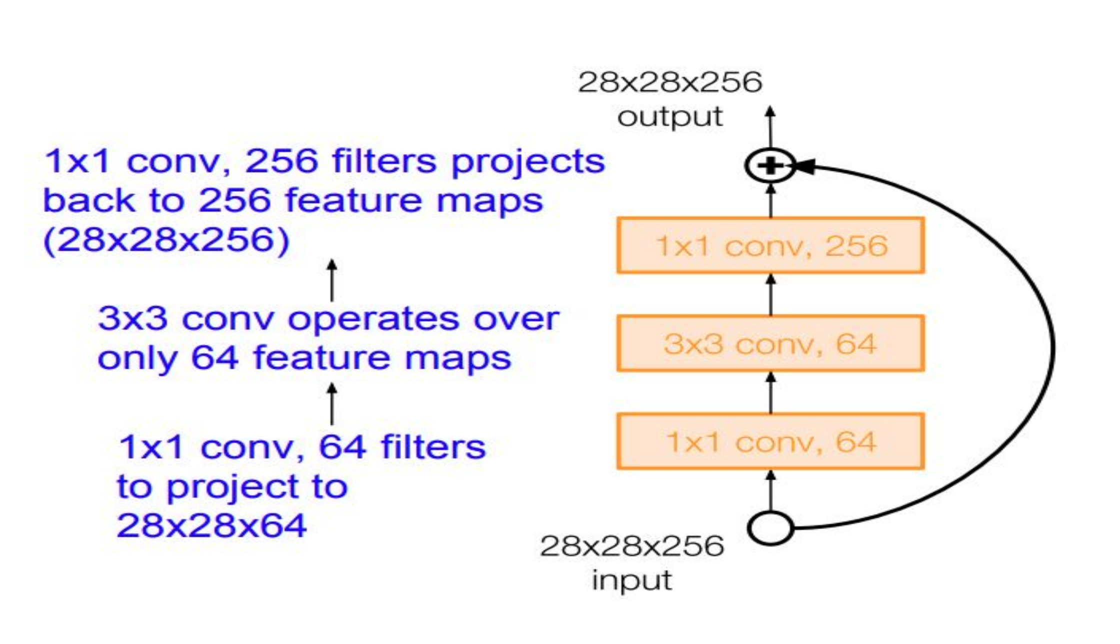

```
from keras.layers import Conv2D, Input

# input tensor for a 3-channel 256x256 image
x = Input(shape=(3, 256, 256))
# 3x3 conv with 3 output channels (same as input channels)
y = Conv2D(3, (3, 3), padding='same')(x)
# this returns x + y.
z = keras.layers.add([x, y])
```

Kaiming He et al, Residual Network 2015

ResNet with bottleneck structure

Further improvement



What CNN can do?

CNN has pushed forward the state of the art in
- Classification
- Detection
- Segmentation
- Synthesize images and videos

It has been used widely in different tasks:

Game:
- AlphaGo
- Poker

Applications:
- Face recognition
- Surveillance
- Fashion
- Medical image

What CNN can't do?

We shall be careful to claim CNNs outperform human vision systems in real life.

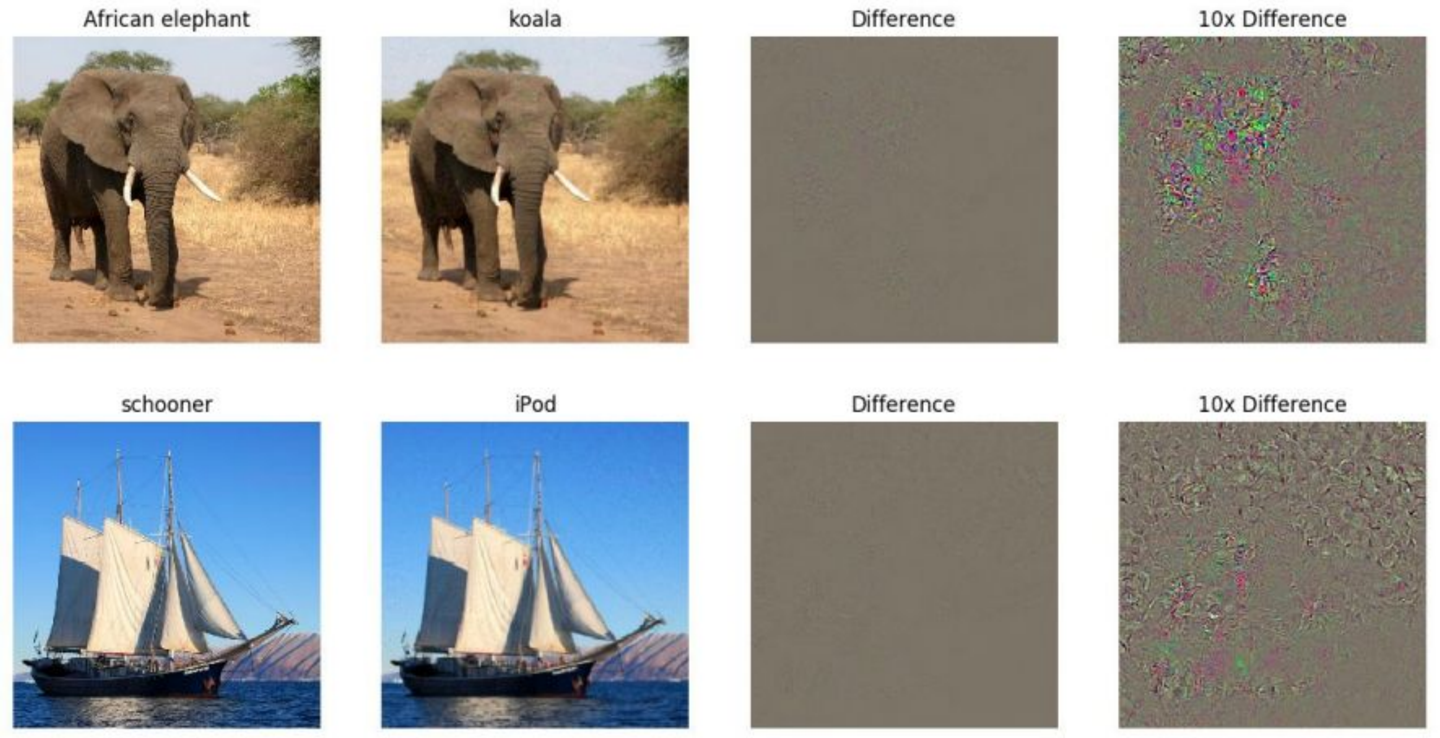
CNNs are not always reliable

Guess what an ImageNet model will predict the following pics

CNNs are not always reliable

Guess what an ImageNet model will predict the following pics

Adversarial attack



How to compute adversarial examples

$$J(\tilde{\boldsymbol{x}}, \boldsymbol{\theta}) \approx J(\boldsymbol{x}, \boldsymbol{\theta}) + (\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x})$$

Maximize $(\tilde{\boldsymbol{x}} - \boldsymbol{x})^\top \nabla_{\boldsymbol{x}} J(\boldsymbol{x})$

Subject to $||\tilde{\boldsymbol{x}} - \boldsymbol{x}||_\infty \leq \epsilon$

Thus we can generate adversarial examples

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \epsilon \text{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{x})\right)$$

This is called Fast Gradient Sign method.

How to implement adversarial attack

Using exiting toolboxes (e.g., cleverhans, Foolbox), we can find adversarial examples to fool most given classifier

```
# example of find adversarial attack
from cleverhans.attacks import FastGradientMethod
fgsm = FastGradientMethod(model, sess=sess)
fgsm_params = {'eps': 0.3, 'clip_min': 0., 'clip_max': 1.}
adv_x = fgsm.generate_np(orgin_x, **fgsm_params)
```

Refer to [NIPS 2017 adversarial attack competition]() fore more adversarial approaches

How serious is the problem

Most of existing adversarial attacks require either model parameters (white-box attack) or frequent evaluating the model (black-box attack). There are still a lot of room to make the adversarial attack more practical.

But the phenomenon of adversarial attacks suggest there are “holes” at least in many machine learning models. Take home message is: deep CNNs make make mistakes that human vision will not make.

The gap between “research” and “products”

What are research and “products”?

“Research” aims to develop a **new** approach or solve a **new** problem or explain a **new** discovery. Research problems are often defined in a **clear and simple** manner.

“Products” are usually made to **serve the requirement of the customers**. The customers are diversified with different requirements. A mature product is usually **complicated**.

Gaps

1. Many problems in products are not as well defined as in research. For example,
    - Photo management app that attract online users
    - Fashion suggestion based on users purchase history

Such products are complicated!

2. Research is evaluated with one or two metrics, while products consider multiple metrics, such as
- Accuracy/Precision/Recall
- Memory usage, power consumption
- Speed, user experience
- Easy to maintain or upgrade

I believe researchers can learn the perspectives of customers and products.

Lessons

1. Try more projects and gain experiences!

   - try develop a product using your AI

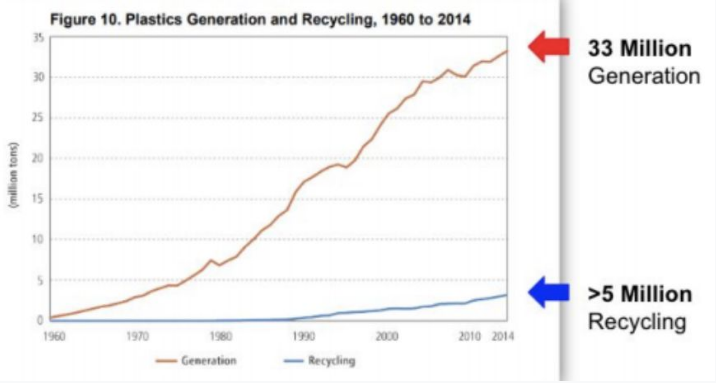
2. Project development via continuously testing and improvement

   - Let’s see two examples.

Example 1

Problem: classifying garbage for environment restoration: recyclable, non-recyclable, food, container, and etc.

by Rachel Jordan (rcj2118@columbia.edu)



Challenge: not enough training labels!

Solution for Example 1

Try different models

- borrow ImageNet models directly
- fine-tune the last layer using liblinear
- fine-tune CNN layers with small rate
- generate more examples using GANs

Collect data:

- no training
- a few labels
- hundreds of labels

Continuously improving
All the models are served by Tensorflow Extended (TFX)

Detailed explanation on fine tuning models

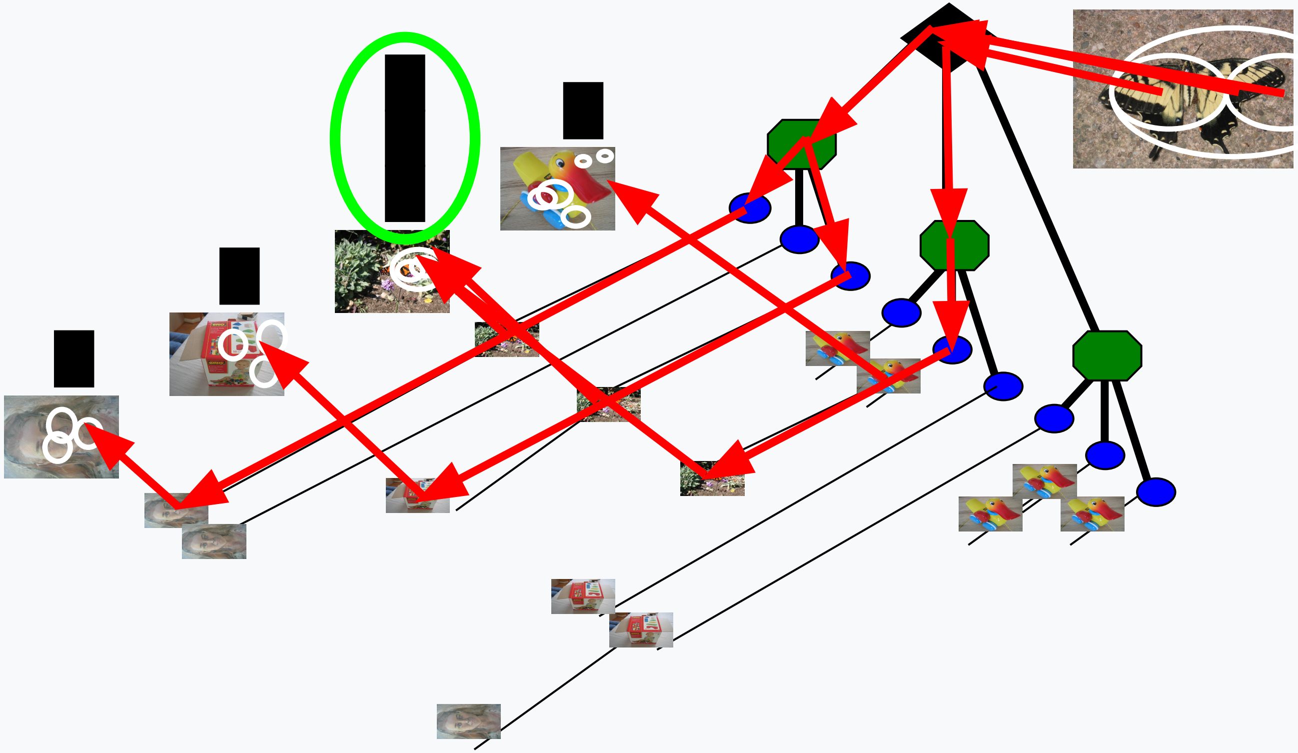
Example 2

Problem: Large scale visual search system

This task has rich applications such as

- Diving into personal albums
- Recommending Youtube/news/TV shows
- Searching and recommending clothes and fashion
- Organizing social media

Traditional approach for visual search



Nister and Stewenius, Scalable recognition with a vocabulary tree, CVPR 2006

How to improve the traditional approach

- Add new local detectors and descriptors
- Learning better rank functions
- Add spatial verification

Disadvantages:

- mostly limited to exact object search

- no semantics or user preferences

A modern approach

Quick prototype:

- Learn a compact embedding for every image

- Similarity search for nearest neighbor search

Using open source libraries:

- flann (search using hierarchical tree)
- lucene (search via inverted index)

Refer to: Visual Search at Pinterest, by Y. Jing et al, KDD’15

A modern approach

Quick prototype:

- Learn a compact embedding for every image

- Similarity search for nearest neighbor search

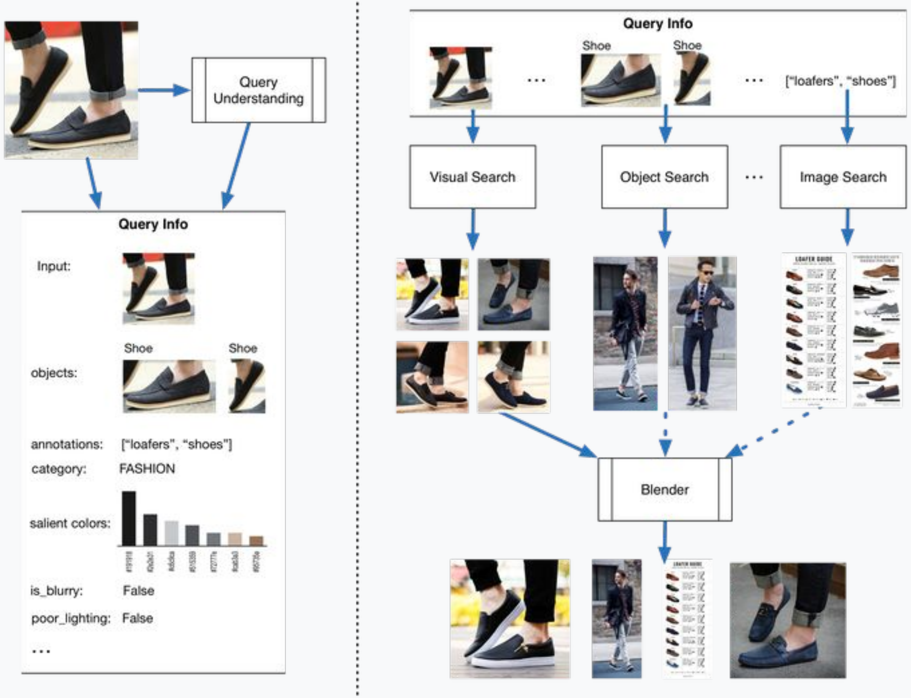
Using open source libraries:

- flann (search using hierarchical tree)
- lucene (search via inverted index)

Refer to: Visual Search at Pinterest, by Y. Jing et al, KDD'15

Continuously improving

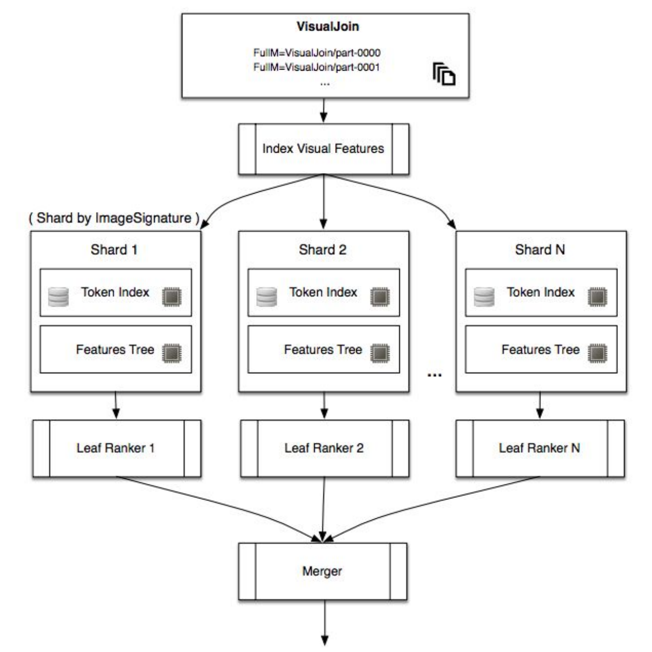
Represent an image using multiple clues: objects, labels, etc



Visual Search at Pinterest, by Y. Jing et al, KDD'15

Continuously improving (2)

Scalable systems and more complicated ranking



Visual Search at Pinterest, by Y. Jing et al, KDD'15

Continuously improving (3)

Learn better embedding model.

Recommendation with fashion and popularity measure.

Li, Cao, Zhu and Luo, Mining fashion outfit composition using an end-to-end deep learning approach on set data, TMM 2017

Topics covered this morning:

- LeNet, AlexNet, ResNet
- Adversarial Attack
- Fine tuning
- Visual Search

Questions we have discussed

- How long did it take CNNs to become popular? Why?
- What can you do with CNNs? What can't do?
- What is the gap between "research" and "products"?

Thank you!

Liangliang Cao
llcao@google.com
www.llcao.net