广告推荐排序模型
CNN embedding应用与实践

李满天/王玉
广告部质量部效果组

京东
JD.COM

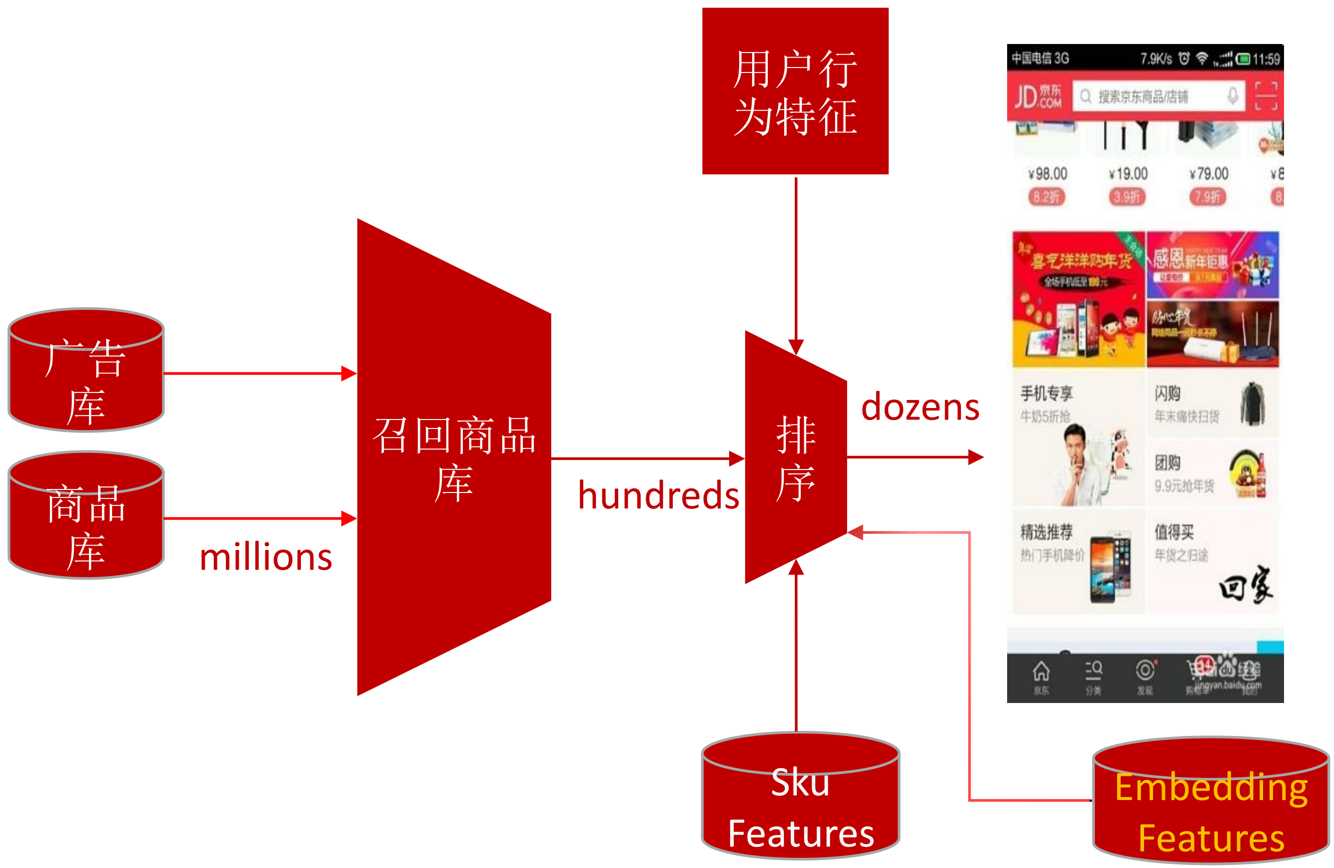
JD.COM 京东

I
- 项目背景

II
- 方案设计 & 技术预研

III
- 应用实践 & 细节优化

项目背景


用户行为特征
广告库
商品库
millions
召回商品库
hundreds
排序
dozens
Sku Features
Embedding Features
手机专享
闪购
团购
精选推荐
值得买

项目背景

JD.COM 京东

- 传统的推荐技术通常遇到新商品/冷门商品的冷启动问题
- 人工特征工程在发掘有效特征和特征组合上费时费力
- 图片等内容特征尚未得到良好的应用

- 深度学习技术从原始内容出发（文本、图像、语音）
- 多层神经网络的丰富表达能力，自动特征学习

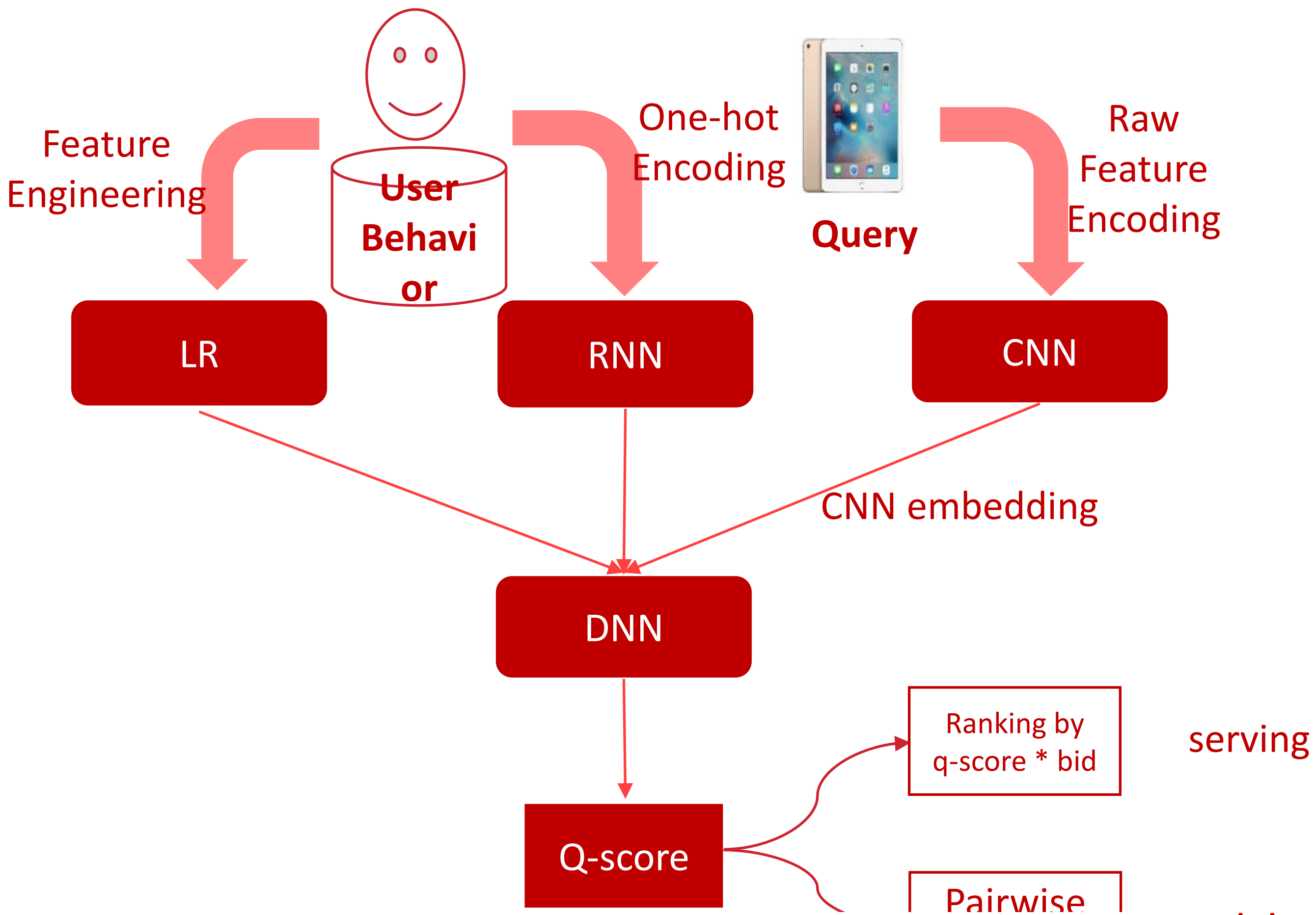
利用深度学习神经网络的学习和表达能力，从原始内容提取有效Embedding特征，帮助排序

I
- 项目背景

II
- 方案设计 & 技术预研

III
- 应用实践 & 细节优化

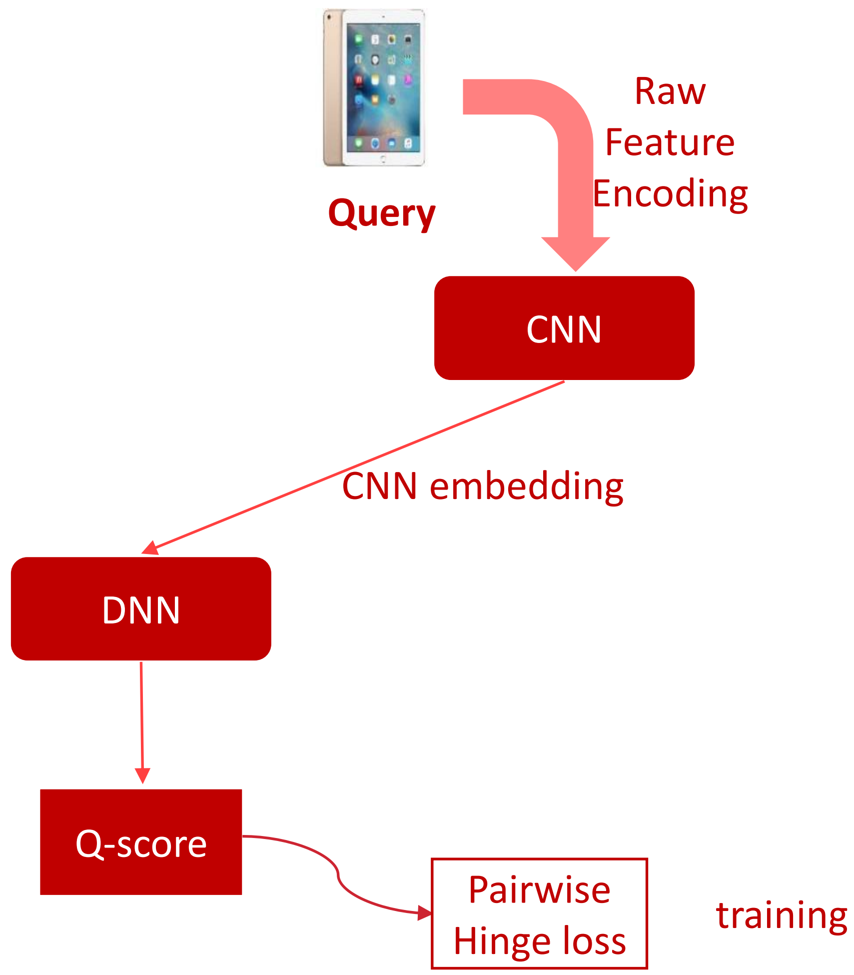
广告推荐统一模型



Problem: Training speed slow down 10x?

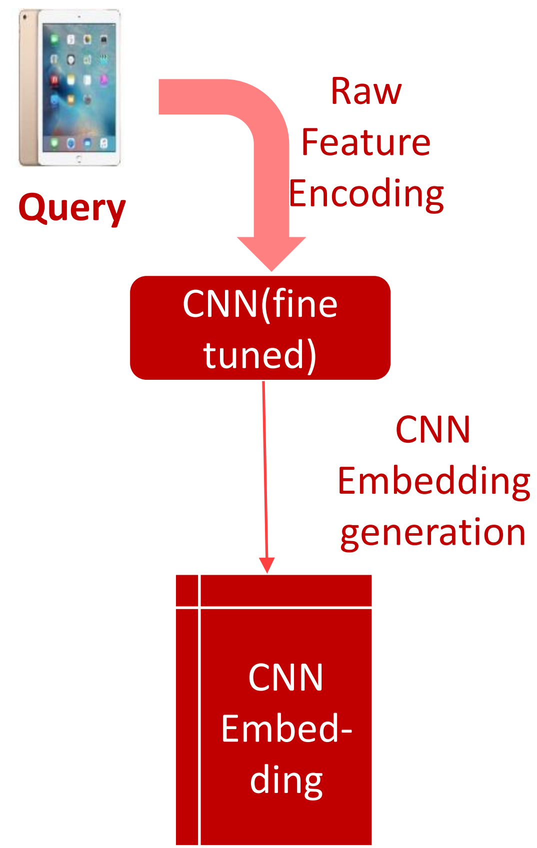
方案设计

■1-Training cnn module alone



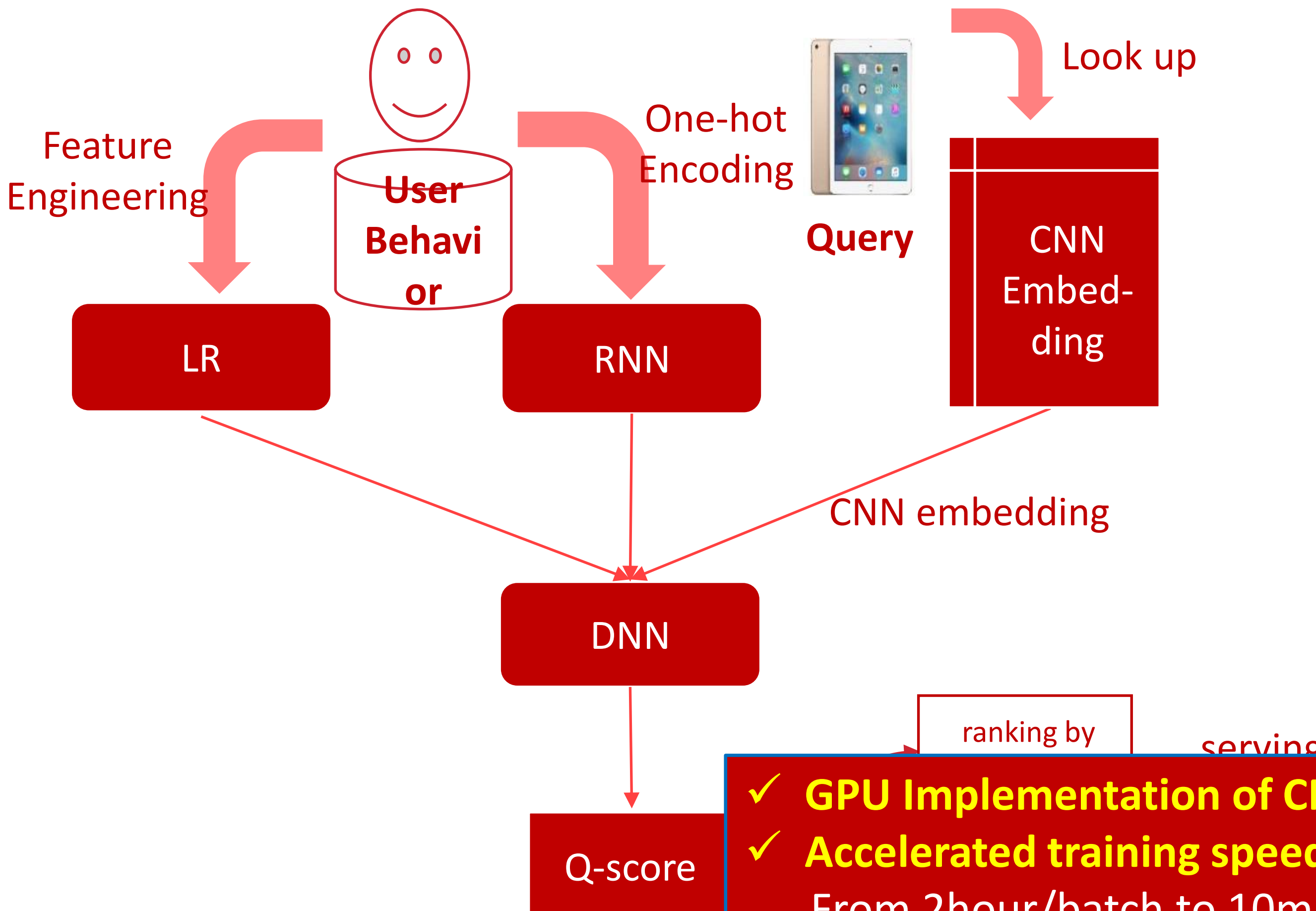
方案设计

- 2-Cnn embedding generation



方案设计

3-Joint training with lr & cnn embedding



- ✓ **GPU Implementation of CNN training**
- ✓ **Accelerated training speed**
  From 2hour/batch to 10min/batch, ↑10x
- ✓ **Cost of performance loss?**

- Inception - Deep convolutional neural network architecture

1989~1998

LeNet, Y. LeCun et al.

Hand-writen zip code recognition

2012

AlexNet, Winner of ISVRC-2012

ReLU activation

Dropout, Data Augmentation

Big Data, GPU implementation

Normalization:LRN

2014

VGGNet

GoogLeNet: Inception-v1

22 layers deep net

stacked Inception module

2015

Inception-v2

Batch normalization

pRelu

2016

Inception-v3

42 layers, 2.5x cost v1

Res-Net

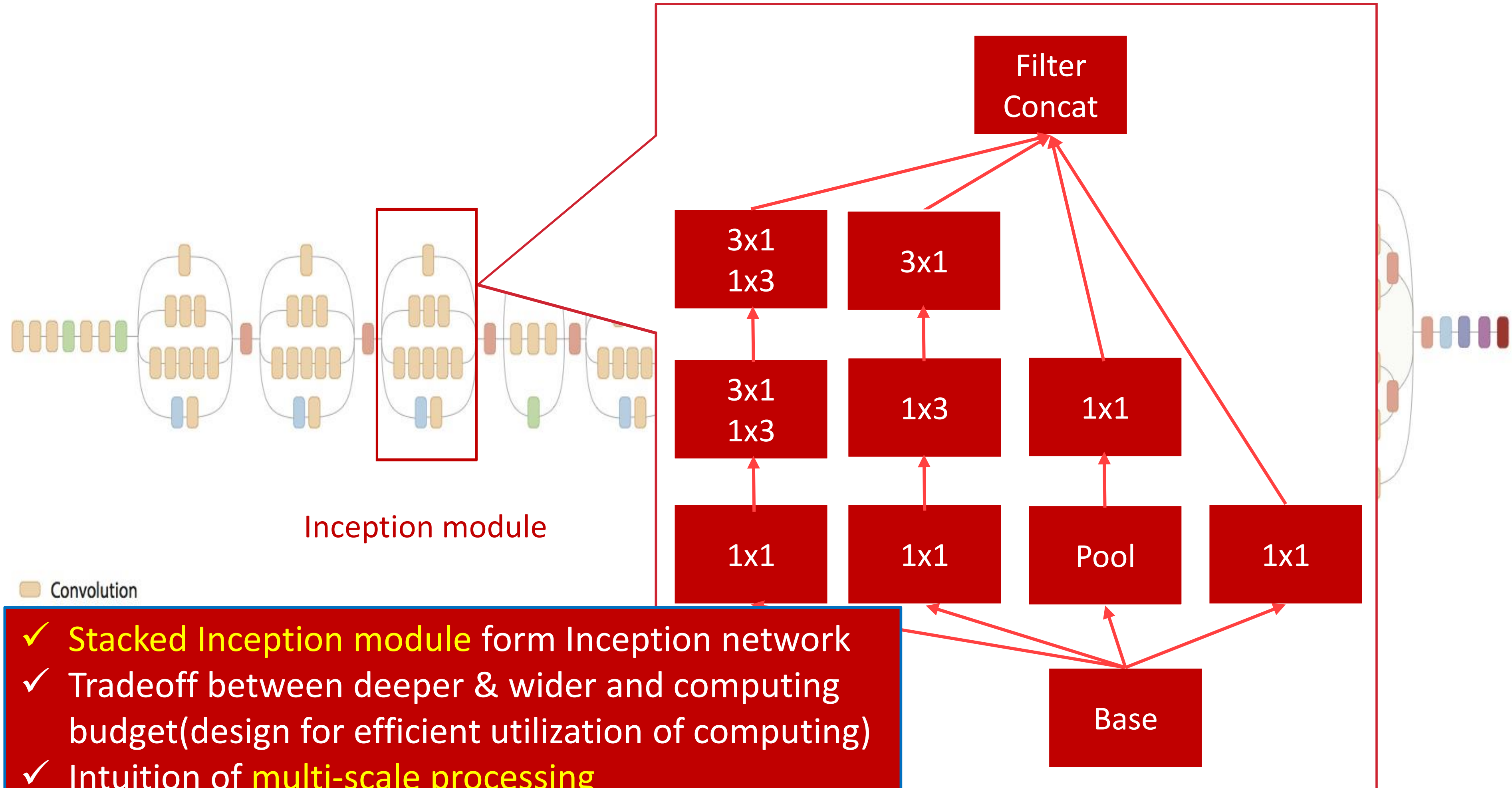
Very deep 152layers

Inception-v4

Res + Inception

Inception

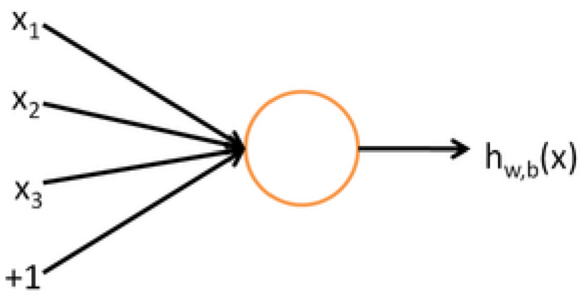
■CNN networks – Inception v3



- ✓ Stacked Inception module form Inception network
- ✓ Tradeoff between deeper & wider and computing budget(design for efficient utilization of computing)
- ✓ Intuition of multi-scale processing
- ✓ Judiciously applying dimension reductions and projections. 1x1 conv before expensive 3x3 or 5x5

he Inception Architecture for Computer Vision, Inception-v3

Batch normalization



$$h(x) = relu(x)$$

$$f(x) = b + \sum_{i=1}^{n} w_i x_i$$

$$h(x) = relu(x)$$

$$hr(x) = \gamma x + \beta$$

$$hn(x) = normalize(x)$$

$$f(x) = b + \sum_{i=1}^{n} w_i x_i$$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \quad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \quad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \quad \text{// normalize}$$

Sergey et al 2015, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift (Inception - v2)

TensorFlow Slim

JD.COM 京东

VGG Net in TF-Slim

```
# VGG16 in TF-Slim.
def vgg16(inputs):
  with slim.arg_scope([slim.ops.conv2d, slim.ops.fc], stddev=0.01, weight_decay=0.0005):
    net = slim.ops.repeat_op(2, inputs, slim.ops.conv2d, 64, [3, 3], scope='conv1')
    net = slim.ops.max_pool(net, [2, 2], scope='pool1')
    net = slim.ops.repeat_op(2, net, slim.ops.conv2d, 128, [3, 3], scope='conv2')
    net = slim.ops.max_pool(net, [2, 2], scope='  ol2')
    net = slim.ops.repeat_op(3, net, slim.ops.conv  , 256, [3, 3], scope='conv3')
    net = slim.ops.max_pool(net, [2, 2], scope='pool3')
    net
    net
    net
    net
    net
    net
    net
    net
    net
    net
  retur
```

odels in TF.

VGG Net in native TF

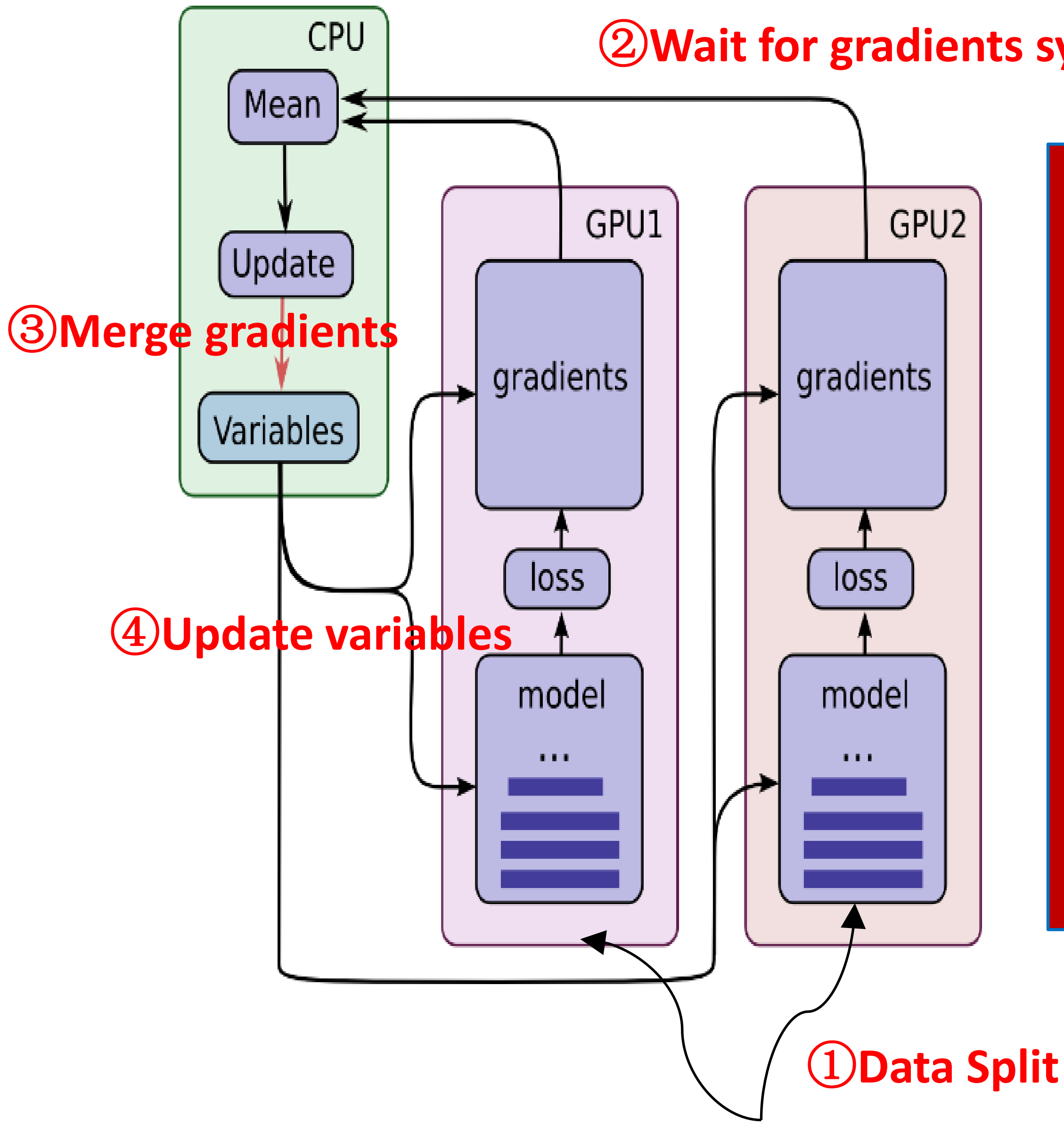
```
# Layers 1-3 (out of 16) of VGG16 in native tensorflow.
def vgg16(inputs):
  with tf.name_scope('conv1_1') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 3, 64], dtype=tf.float32, stddev=1e-1), name='weights')
    conv = tf.nn.conv2d(inputs, kernel, [1, 1, 1, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0, shape=[64], dtype=tf.float32), trainable=True, name='biases')
    bias = tf.nn.bias_add(conv, biases)
    conv1 = tf.nn.relu(bias, name=scope)
  with tf.name_scope('conv1_2') as scope:
    kernel = tf.Variable(tf.truncated_normal([3, 3, 64, 64], dtype=tf.float32, stddev=1e-1), name='weights')
    conv = tf.nn.conv2d(images, kernel, [1, 1, 1, 1], padding='SAME')
    biases = tf.Variable(tf.constant(0.0, shape=[64], dtype=tf.float32), trainable=True, name='biases')
    bias = tf.nn.bias_add(conv, biases)
    conv1 = tf.nn.relu(bias, name=scope)
  with tf.name_scope('pool1')
    pool1 = tf.nn.max_pool(conv1, ksize=[1, 2, 2
```

AUC, Precision_at_k, recall_

- ✓ More concise coding style, efficient development and testing
- ✓ Arg_scope facilitate parameters adjustment experiment

Synchronous Data Parallelism

JD.COM 京东



- ✓Place an individual model replica on each GPU. Split batch across the GPUS.
- ✓Update model parameters synchronously by waiting for all GPUs to finish one batch
- ✓The gradients are combined and averaged across the multiple towers in order to provide a single update of the Variables stored on CPU.
- ✓Using either a single local machine with multiple GPU cards or a cluster of machines.

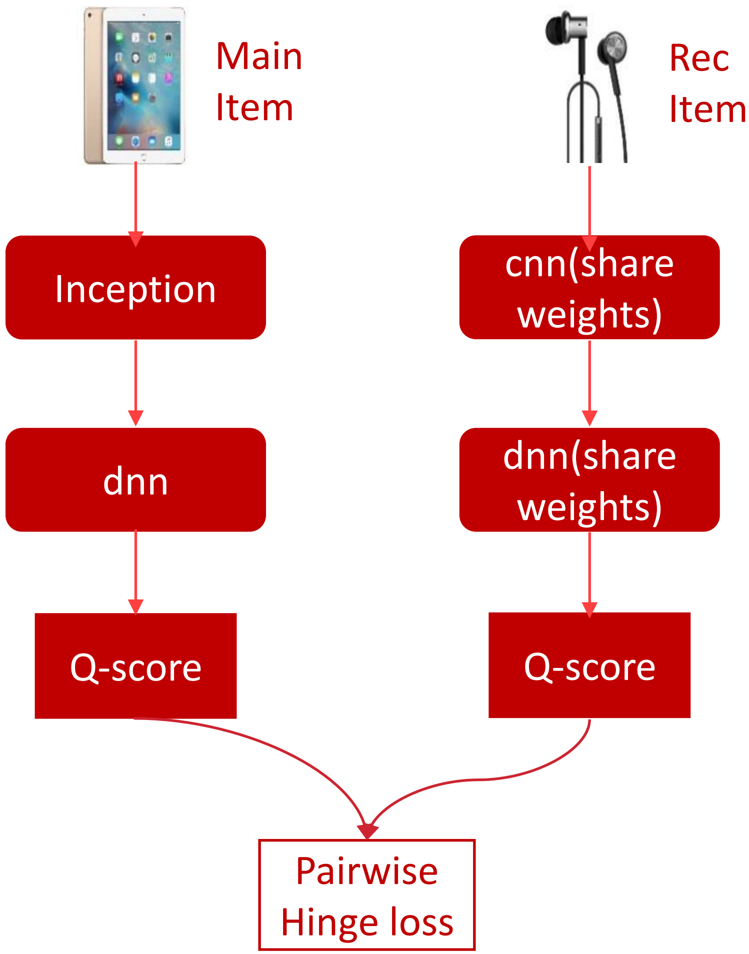
I
- 项目背景

II
- 方案设计 & 技术预研

III
- 应用实践 & 细节优化

应用实践

Pairwise trained inception networks



- ✓ **Share weights by tensorflow api**
  tf.get_variable_scope().reuse_variables()
- ✓ **Batch normalization tricks**
  Group all data in a batch to do batch norm, other than treat positive & negative samples respectively
- ✓ **Pay attention to sparsity issue**
  Substitute pooling layer with stride conv

细节优化

JD.COM 京东

Joint training

embedding module

structure-1

- ✓Dot product as similarity feature.
- ✓No additional params need tuned.
- ✓Acceptable discriminate feature.

Look up

CNN Embed -ding

Emb-main

Emb-rec

Ft

1 Dimension Feature

Dot Product Similarity

$$sim\_ft(\vec{x},\vec{y}) = \sum_{i=1}^{n} x_i\, y_i$$

细节优化

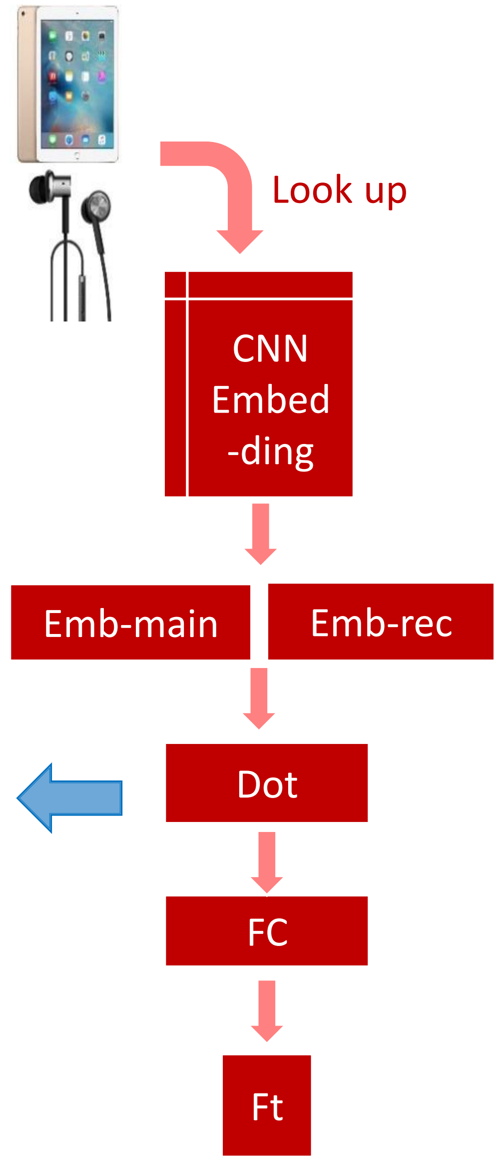
Joint training

embedding module

structure-2

- ✓Separate dot product as two phase
- ✓Add params for fine training
- ✓Better performance for unify model ranking
- ✓NN approximate arbitrary function

$$\overrightarrow{dot\_ft}(\vec{x},\vec{y}) = \vec{x} \odot \vec{y}$$



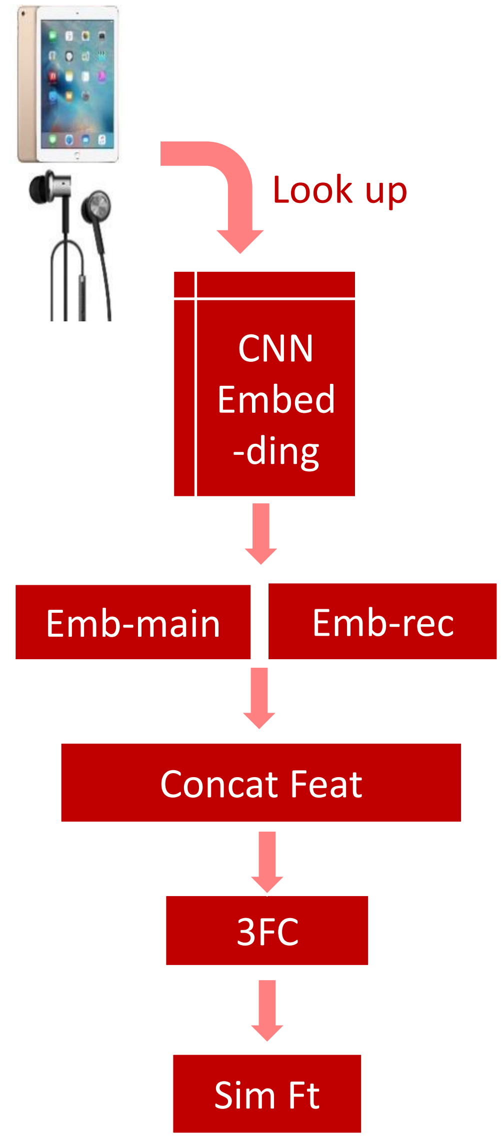
细节优化

Joint training

embedding module

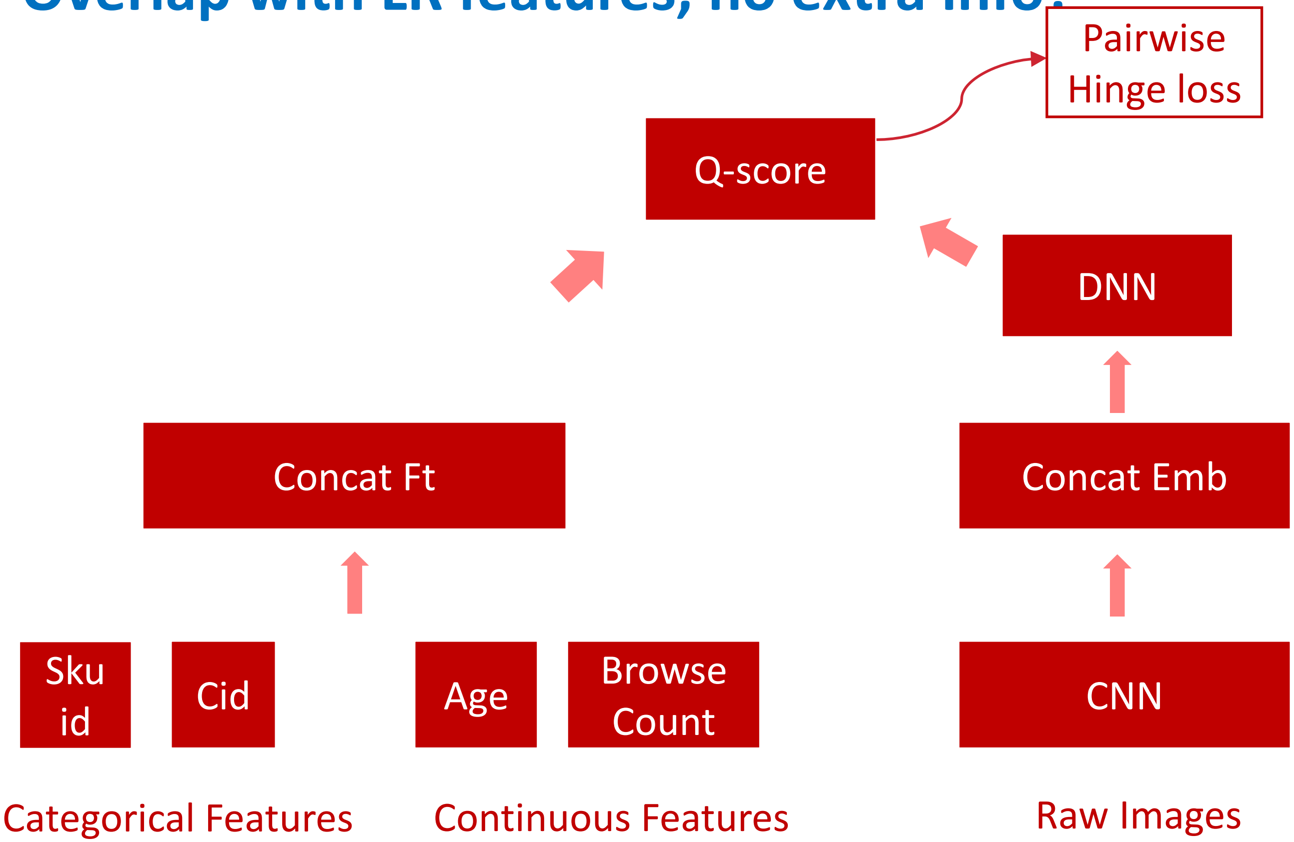
structure-3

- ✓Training DNN as a better similarity function for specific problem.
- ✓Output multi dimension vector for joint training.



- Embedding training, **complementary training**

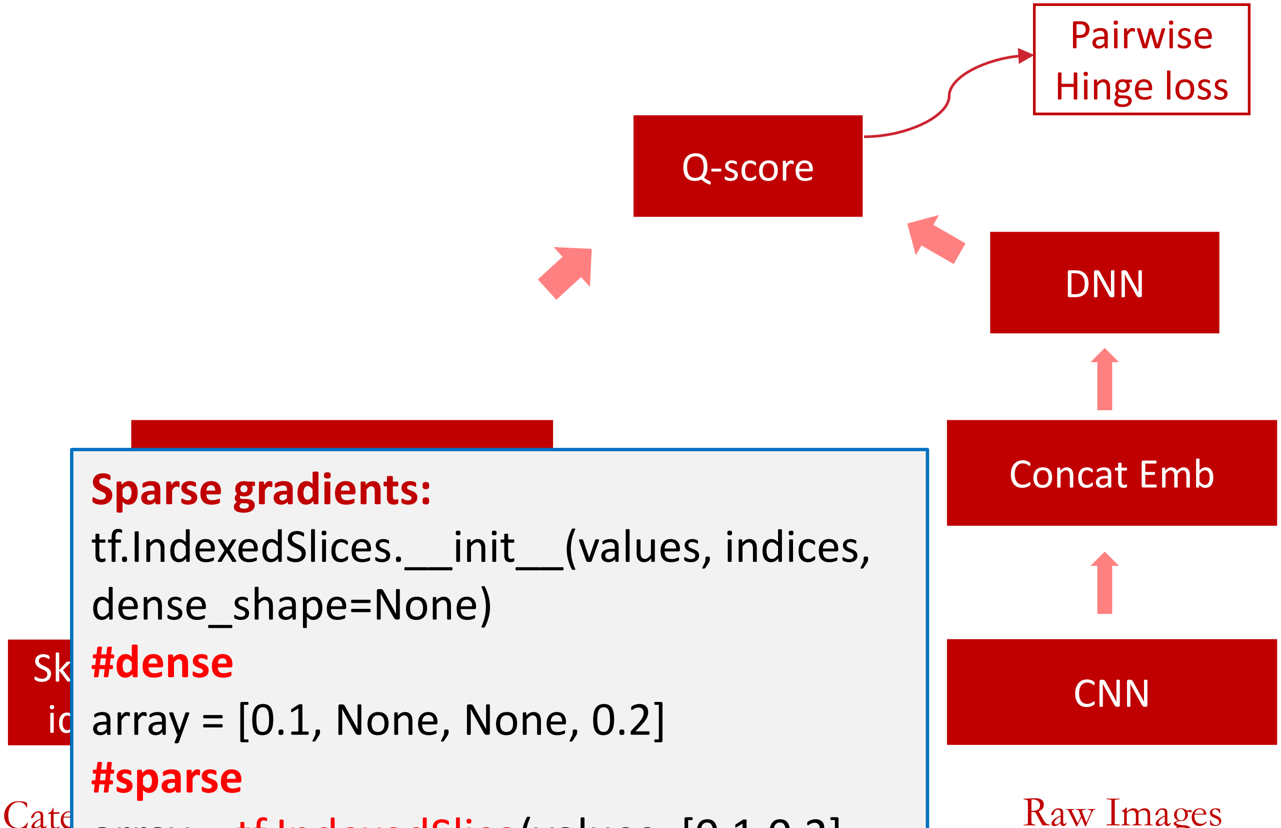
Overlap with LR features, no extra info?



Problem: Training speed slow down 3x?

细节优化

■ Embedding training, **complementary training – diagnose**



Sparse gradients:

tf.IndexedSlices.__init__(values, indices, dense_shape=None)

#dense

array = [0.1, None, None, 0.2]

#sparse

array = tf.IndexedSlice(values=[0.1,0.2], indices=[0,3],dense_shape=(4,))

细节优化

JD.COM 京东

Embedding training, complementary training – diagnose

```
average_grads = []
for grad_and_vars in zip(*tower_grads):
  # Note that each grad_and_vars looks
  #   ((grad0_gpu0, var0_gpu0), ... ,
  grads = []
  for g, _ in grad_and_vars:
    # Add 0 dimension to the gradients
    expanded_g = tf.expand_dims(g, 0)

    # Append on a 'tower' dimension whi
    grads.append(expanded_g)

  # Average over the 'tower' dimension.
  grad = tf.concat(0, grads)
  grad = tf.reduce_mean(grad, 0)

  # Keep in mind that the Variables are redundant because they are shared
  # across towers. So .. we will just return the first tower's pointer to
  # the
  v = gr
  grad_a
  averag
return a
```

```
# Add Op to graph
if output_structure:
  op = g.create_op(op_type_name, inputs, output_types, name=scope,
                   input_types=input_types, attrs=attr_protos,
                   op_def=op_def)
  outputs = op.outputs
  return _Restructure(ops.convert_n_to_tensor(outputs), output_structure)
else:
  return g.create_op(op_type_name, inputs, output_types, name=scope,
                     input_types=input_types, attrs=attr_protos,
                     op_def=op_def)
```

```
def _apply_dense(self, grad, var):
  rms = self.get_slot(var, "rms")
  mom = self.get_slot(var, "momentum")
  return training_ops.apply_rms_prop(
      var, rms, mom,
      self._learning_rate_tensor,
      self._decay_tensor,
      self._momentum_tensor,
      self._epsilon_tensor,
      grad, use_locking=self._use_locking).op

def _apply_sparse(self, grad, var):
  raise NotImplementedError()
```

- ✓ **tf.expand_dims()** op will expand sparse tensors to dense expression which will slow down the training speed.
- ✓ **RmsProp** (default optimizer) not support sparse operation in version 0.8.0. This issue is resolved after tensorflow 0.10.0

细节优化

JD.COM 京东

Embedding training, complementary training – solution

```
average_grads = []
for grad_and_vars in zip(*tower_grads):
  # Note that each grad_and_vars looks like the following:
  #   ((grad0_gpu0, var0_gpu0), ... , (grad0_gpuN, var0_gpuN))
  if isinstance(grad_and_vars[0][0], ops.IndexedSlices):
    dense_shape = grad_and_vars[0][0].dense_shape
    s = dict()
    ia = []
    va = []
    j = 0
    for g, _ in grad_and_vars:
      indexs = g.indices
      values = g.values
      #print ("IndexedSlice size = %d" %indexs.get_shape()[0])
      for i in range(indexs.get_shape()[0]):
        if indexs[i] not in s.keys():
          s[indexs[i]] = j
          j += 1
          #s.add(indexs[i])
          ia.append(indexs[i])
          va.append(values[i,:])
        else:
           tmp = s[indexs[i]]
           va[tmp] = tf.add(va[t
    ia = tf.pack(ia)
    #print ("grad_and_vars.size =
    va = tf.pack(va) / (1.0 * len
    grad = ops.IndexedSlices(va,
```

```
# Create an optimizer that performs gradient descent.
#-----------------------------------------------------------------
opt = tf.train.RMSPropOptimizer(lr, RMSPROP_DECAY,
                                momentum=RMSPROP_MOMENTUM,
                                epsilon=RMSPROP_EPSILON)
opt_2 = tf.train.FtrlOptimizer(lr, l1_regularization_strength=1.0)
```

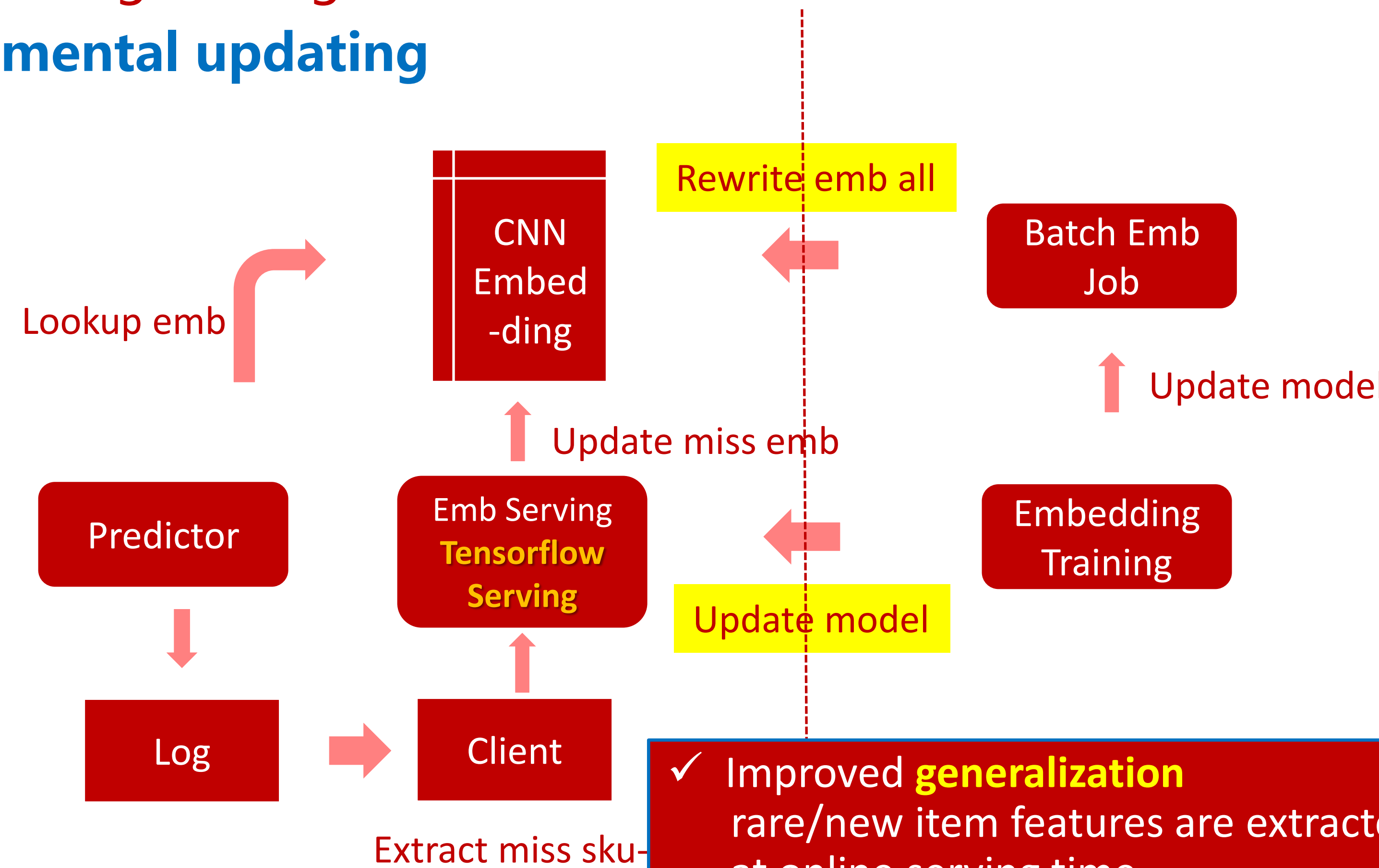
- ✓ **Split average gradient function** to two parts: dense & sparse.
- ✓ **Use RMSProp optimizer for dense part.** RMSProp proven to be one of the best algorithms for deep models like DNN.
- ✓ **Use Ftrl optimizer for sparse part.** Ftrl proven to be a better choice for shallow models like Logistic Regression.

细节优化

JD.COM 京东

- Embedding serving:

Incremental updating



ONLINE SERVING & EMBEDDING UPDATE

- ✓ Improved **generalization**
  rare/new item features are extracted at online serving time
- ✓ **Quasi real-time** updating
  offline update(1 day) vs. online(5 min)

应用场景

- 商品详情页推荐 - 主商品触发推荐结果
  - CNN -> SKU embedding
  - 主商品embedding，推荐商品embedding
  - Sim(主商品，推荐商品A)>Sim(主商品，推荐商品
- 无类目引流推荐 - 用户标签/行为触发推荐结
  - CNN -> SKU embedding
  - 用户浏览行为embedding，推荐商品embedding
  - Sim(用户行为，推荐商品A)>Sim(用户行为，推荐
- 搜索页推荐 - 搜索query触发搜索结果
  - Q_CNN -> Query embedding, SKU_CNN -> SKU em
  - Query embedding, 推荐商品embedding
  - Sim(用户query，推荐商品A)>Sim(用户query，推

未来计划 & 项目成员

JD.COM 京东

- 未来计划
  - Sku title embedding
  - User behavior embedding/modeling
- 项目成员
  - 王玉
  - 李满天、徐吉兴、吴敖寒、李季冬
- 参考文献

  - https://www.tensorflow.org/
  - https://github.com/tensorflow/models/blob/master/inception/
  - LeCun et al, Backpropagation applied to handwritten zip code recognition
  - Sergey et al 2016, Rethinking the Inception Architecture for Computer Vision
  - Sergey et al 2015, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
  - Szegedy et al 2014, Going deeper with convolutions
  - McMahan et al, Ad Click Prediction: a View from the Trenches
  - Cheng et al, Wide & Deep Learning for Recommender Systems
  - Krizhevsky et al, ImageNet Classification with Deep Convolutional Neural Networks
  - He et al, Deep Residual Learning for Image Recognition
  - He et al, Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification

谢　谢！