Know Your Bands:
A Content-Based Image Retrieval Mobile App

Final Presentation

14252392

SUN Jingxuan

Supervisor: Prof. Yuen Pongchi

Observer: Prof. Xu Jianliang

2018.4.21

Outline

- Introduction
  - Motivation
  - Functionality
  - Pipeline
- Sem 1: Building pipeline
  - Data Collection
  - Object Recognition
  - Feature Extraction
  - Instance Matching
  - Information Retrieval
  - Initial test and problems
- Sem 2: Optimization
  - Network optimization
  - Feature fusion experiments
- Mobile development
- Q&A

Introduction

1. Motivation

- Heavy metal as personal interest
- Distinctive characteristics in visual representation
  - Logos: highly textured patterns, less pictorial
  - Cover arts: without name of band/album; SNS as source
- Visually complex even for human beings → Deep learning

2. Functionality and Problem Domain

- Input: 1 real-life photo w/ logo(s) and/or album(s)
- Output: related album/band information (finally returned Metal-archive band profile link)

- Object recognition
- Feature extraction
- Instance matching

2. Functionality and Pr

- Object recognition
- Feature extraction
- Instance matching

2. Functionality an

- Object recognition
- Feature extraction
- Instance matching

2. Functio

- Obje
- Featu
- Insta

```
qimg: mastoeos.jpg
len(crop): 3
('album', 'mastoeos_cropped0.jpg')
[[ 1.36864418  1.37014265  1.37916231  1.38287845  1.38346592  1.38397321
```

Mastodon

report an error

Country of origin:	United States	Genre:	Progressive/Sludge Metal (early), Progressive Metal/Rock (later)
Location:	Atlanta, Georgia	Lyrical themes:	Life, Hardships, Heroic tales, Mythical beasts, Literature, Spirituality
Status:	Active	Current label:	Reprise Records
Formed in:	2000		
Years active:	2000-present		

Officially formed on January 13, 2000, a few days after Brann Dailor and Bill Kelliher had met the other members at a High on Fire show.

Founded as a five-piece with Eric Saner on vocals, who left the band just before going on tour. The rest of the band members completed the tour sharing vocal duties, and eventually decided to continue as a four-piece.

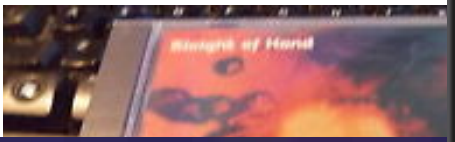
Scott Kelly from Neurosis has appeared as ...

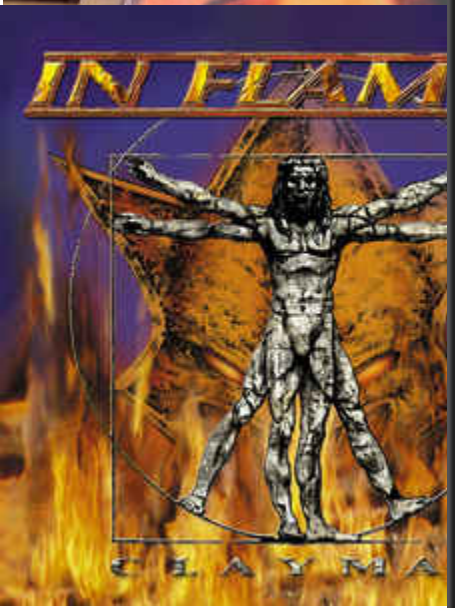
Read more


```
Путь Наверх Часть 1
BAND: https://www.metal-archives.com/bands/%D0%9A%D0%B8%D0%BF%D0%B5%D0%BB%D0%BE%D0%B2/8649
Nightblade
Servant To Your Lair
https://www.metal-archives.com/bands/Nightblade/29763
https://www.metal-archives.com/bands/Nightblade/3540353095
```

Sem 1: Building Pipeline

1






```
band_html
band_id
band_stats
    country
    location
    status
    formed_in
    genre
    theme
    label
    years_active
    current_members
    similar_artists
    related_links
    comment
    logo_url
```

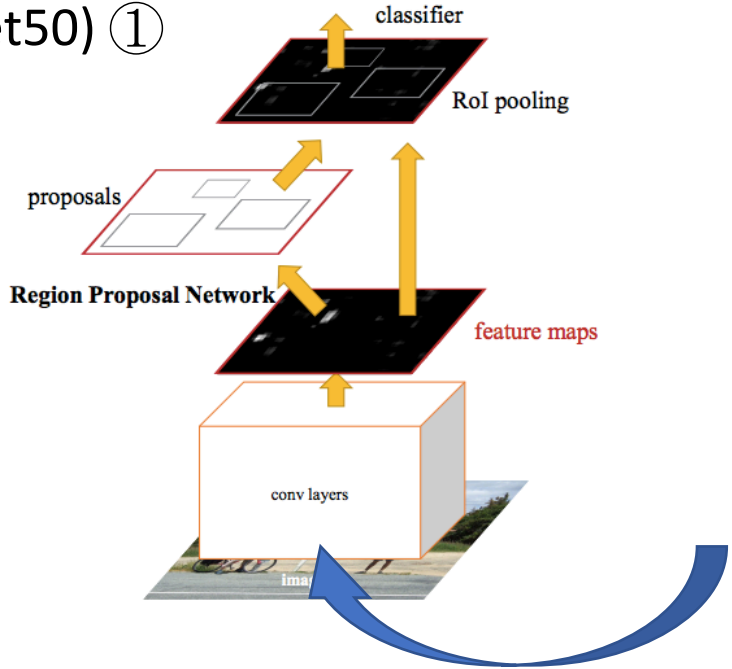
cogs ma

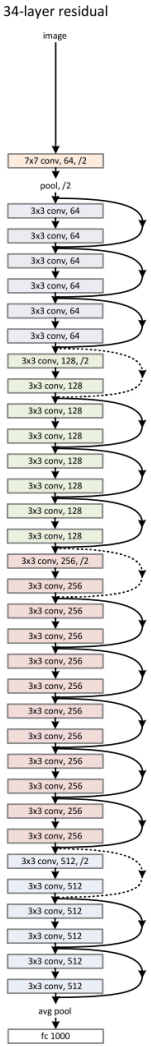
l festiva

odon/13

2. Object Recognition

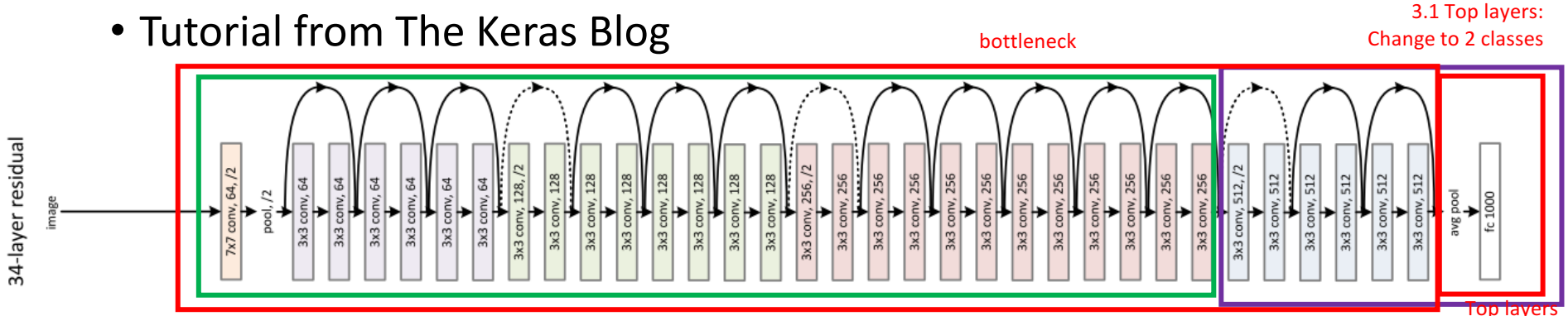
- Detection – new FRCNN end-to-end ②
- Classification – pre-train CNN (Resnet50) ①





Pre-train Resnet50

- Fine-tune top layers of pre-trained ImageNet networks
- Tutorial from The Keras Blog



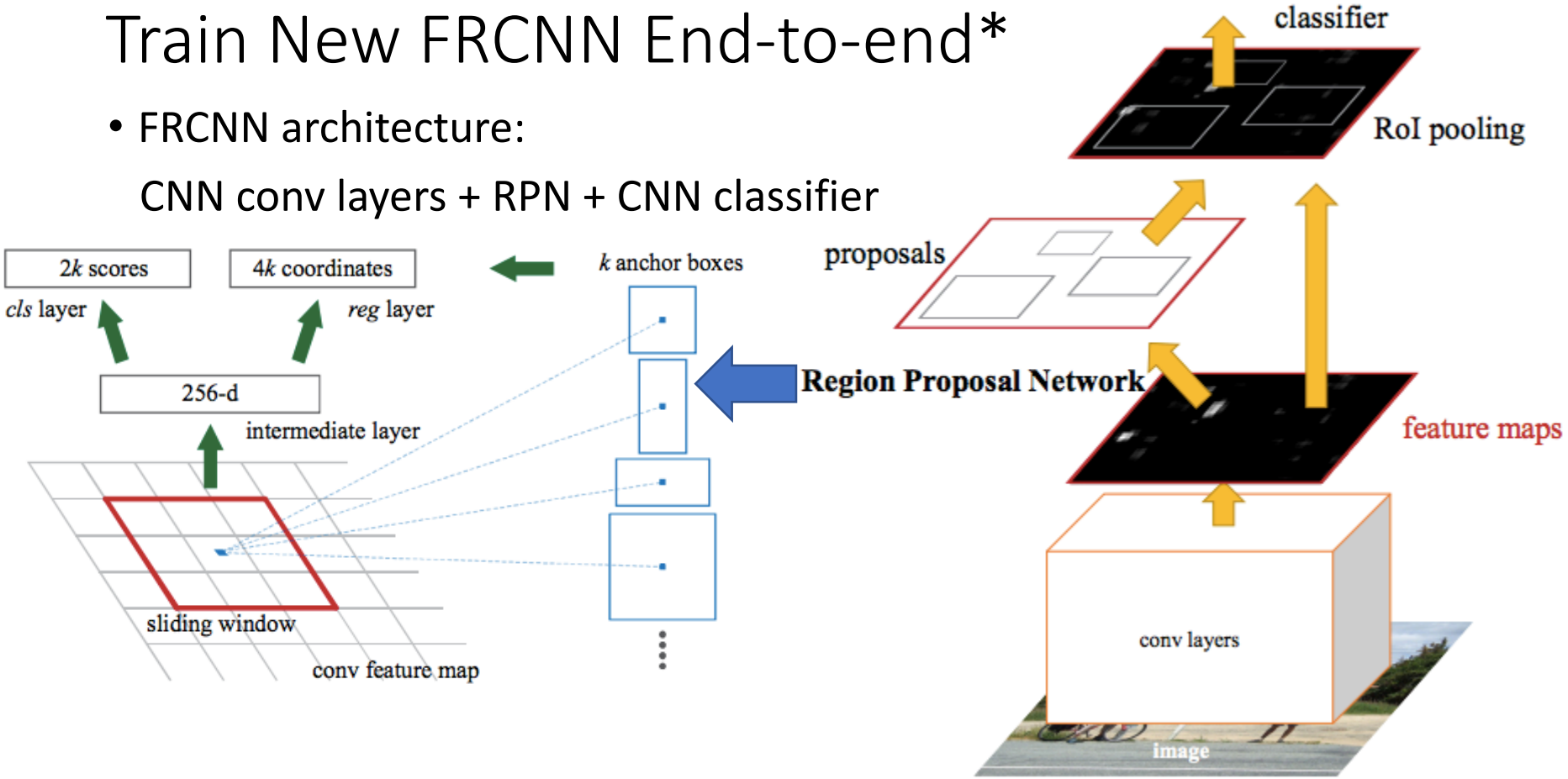
1. Pre-trained model weights → network, run train img through → bottleneck features
2. Train self-def top model (2 classes) → top model weights
3. → fine-tuned model weights

6000 images of each class (12000 in total)
Training/validation 3:1
Step 2 and 3 each 100 epochs
Accuracy > 0.95, loss < 0.01

Train New FRCNN End-to-end*

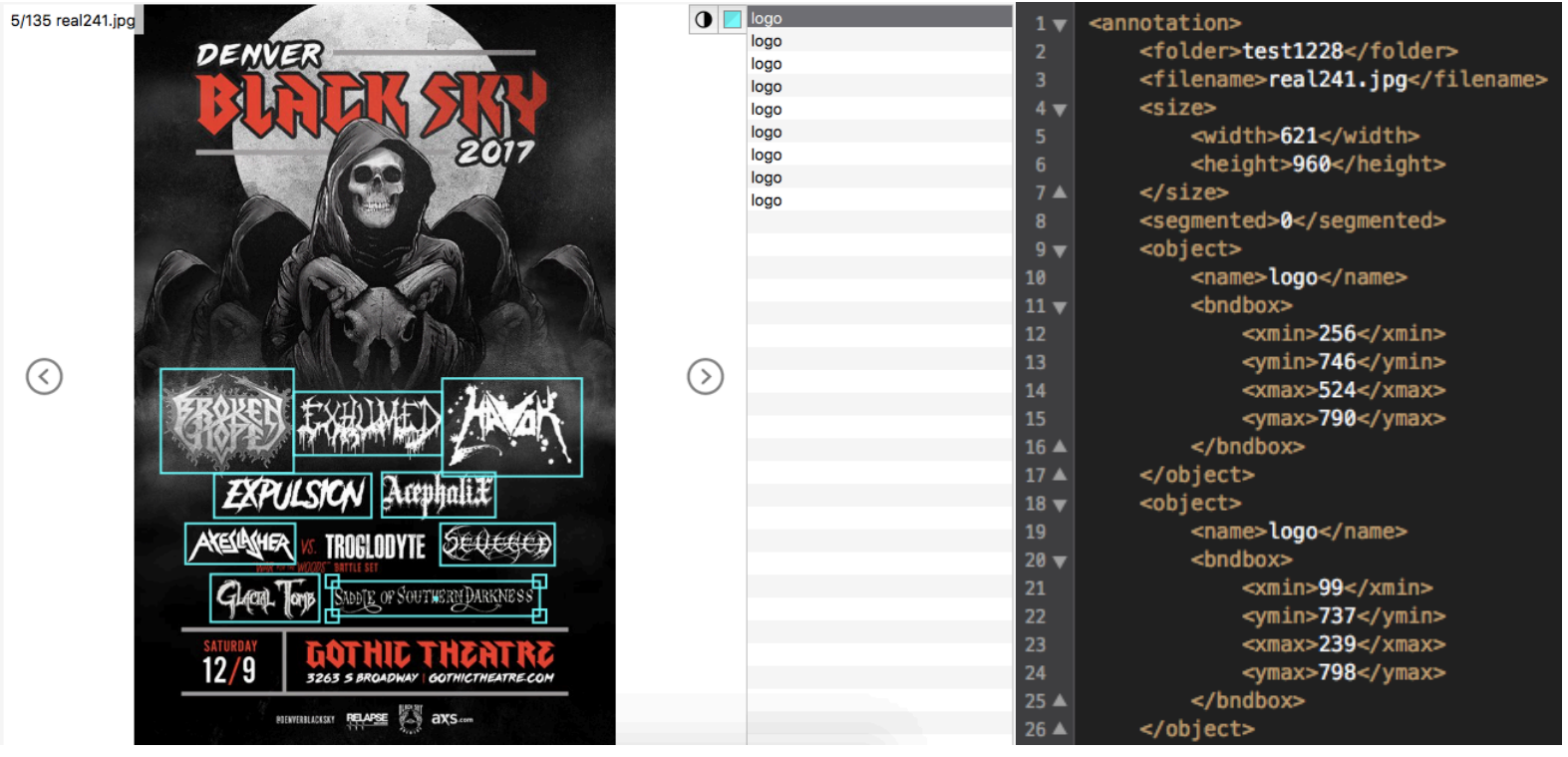
- FRCNN architecture:

  CNN conv layers + RPN + CNN classifier



Train New FRCNN End-to-end*

- Train img + bbox info → network → RoI info + obj class



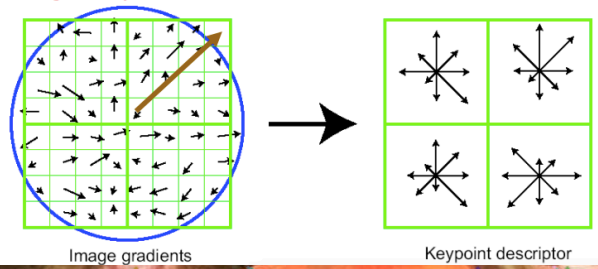
Train New FRCNN End-to-end*

- → FRCNN model weights

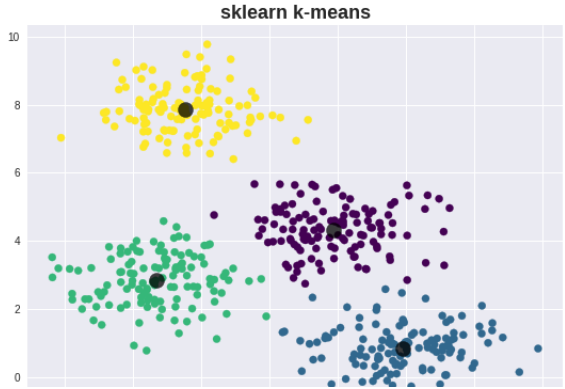
```
Epoch 298/350
 999/1000 [============================>.] - ETA: 1s - rpn_cls: 0.2556 - rpn_regr: 0.0404 - detector_cls: 0.0440 - detector_regr: 0.0059Average number of over
lapping bounding boxes from RPN = 25.261 for 1000 previous iterations
1000/1000 [==============================] - 1522s - rpn_cls: 0.2557 - rpn_regr: 0.0404 - detector_cls: 0.0440 - detector_regr: 0.0059
Mean number of bounding boxes from RPN overlapping ground truth boxes: 25.239
Classifier accuracy for bounding boxes from RPN: 0.98315625
Loss RPN classifier: 0.281904333636
Loss RPN regression: 0.0405812064885
Loss Detector classifier: 0.0440586159273
Loss Detector regression: 0.00606954138941
Elapsed time: 1522.04713583
Total loss decreased from 0.384222835034 to 0.372613697442, saving weights
Best weights so far saved to ./model6.hdf5. best_loss = 0.372613697442
[[  5.40990470e-08   3.61807202e-03   9.12488922e-02   7.39599857e-03
    9.68750000e-01]
 [  1.04356118e-07   4.39091511e-02   5.59981167e-02   2.77232495e-03
    9.68750000e-01]
 [  2.52584994e-01   1.00336432e-01   3.02801607e-03   6.93191495e-03
    1.00000000e+00]
 ...,
 [  5.90860658e-08   2.74258386e-03   1.20253884e-03   4.21987521e-03
    1.00000000e+00]
 [  6.19410301e-08   1.87421683e-02   1.69737056e-01   3.37358974e-02
    9.68750000e-01]
 [  5.59893074e-08   6.67789802e-02   1.09739833e-01   2.01013666e-02
    9.68750000e-01]]
```

1312 train img
500 epochs in total

3. Feature Extraction*



- Keypoint level: SIFT descriptors, 10^2 ~ 10^3 per img
- k-means cluster SIFT from all imgs
- Img level: VLAD descriptor, 1 per img
- → ground truth VLAD descriptors



For each centroid c_i, a sub vector v_i is obtained by accumulating the residual vector between the centroid and local features quantized to this centroid:

$$v_i = \sum_{x:q(x)=c_i} x - c_i \qquad (2)$$

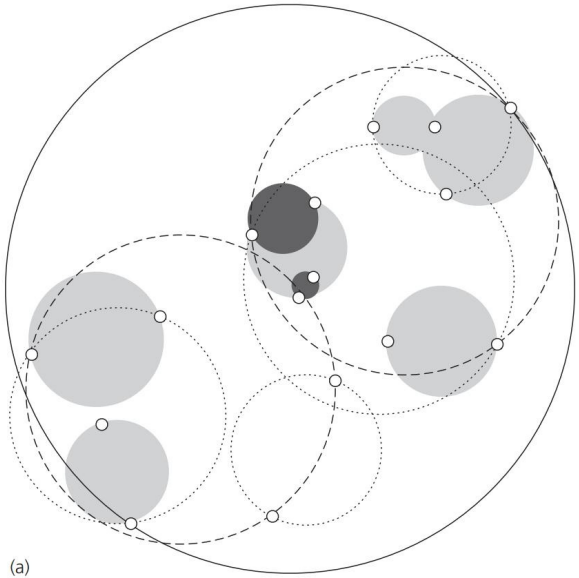
Initial Experiments on Merged Features

- Purpose for initial test is only for illustration and finding potential better-performance feature combinations.

No.	Merged Features	Successful Rate %
i	VLAD	100
ii	VLAD + Object-level FRCNN features	22
iii	Object-level FRCNN features	22
iv	FRCNN column features	22
v	Object-level + FRCNN column features	22
vi	VLAD + Object-level features of Pre-trained CNN	56
vii	VLAD + Column features of Pre-trained CNN	67
viii	Object-level features of Pre-trained CNN	56
ix	Column features of Pre-trained CNN	67
x	Object-level + Column features of Pre-trained CNN	33
xi	VLAD + Color Histogram	0
xii	Color Histogram	0
xiii	VLAD with PCA (6400 components)	<30 (didn't keep record)
xiv	VLAD with PCA (6400 components) + Object-level features of Pre-trained CNN	<30 (didn't keep record)
xv	VLAD with PCA (12800 components)	<30 (didn't keep record)

4. Instance Matching

- VLAD descriptors → construct ball trees
- “comprehensive search for 3D space”
- Search for 10 most similar imgs, return IDs

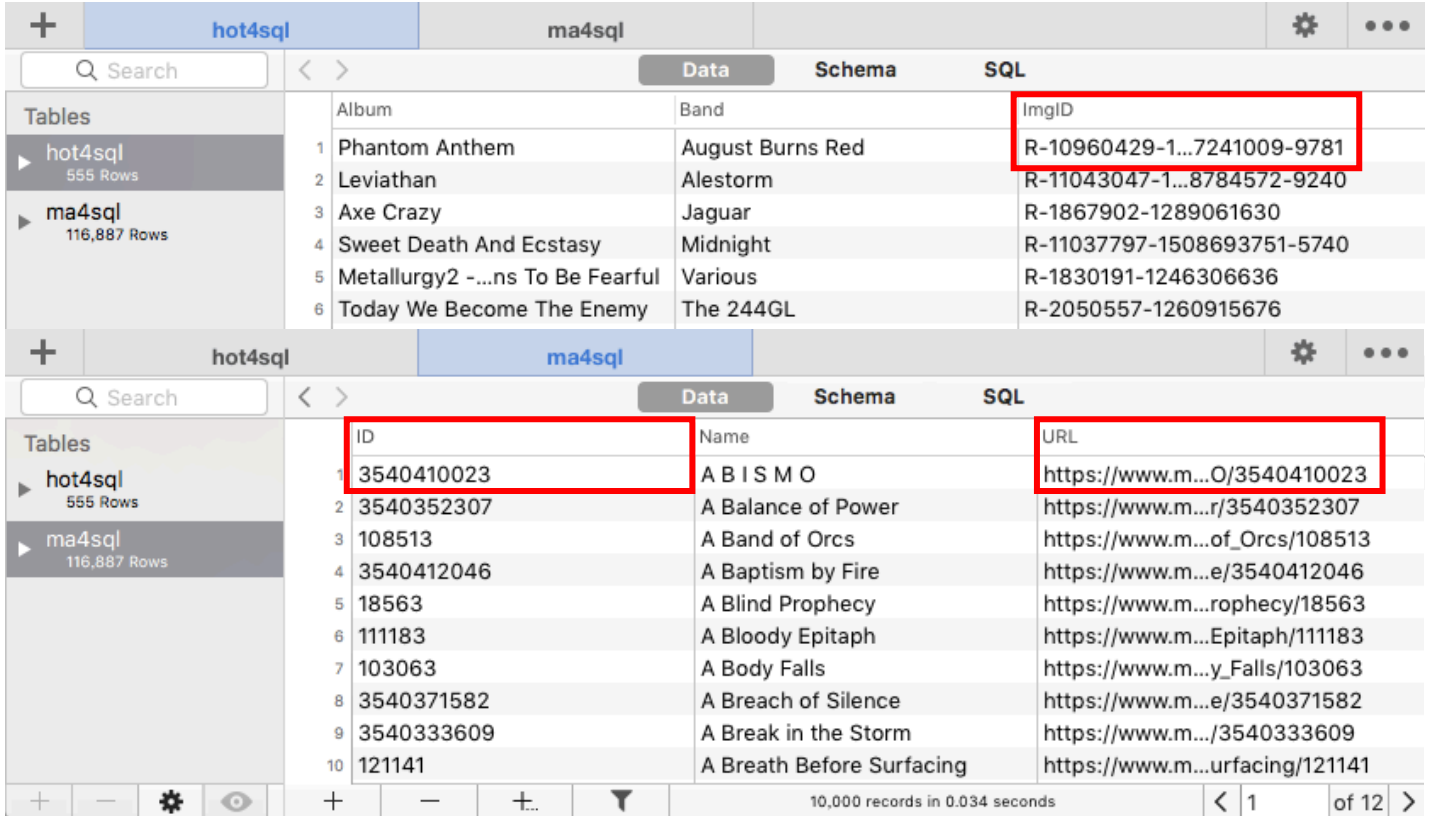


5. Information Retrieval

- Info from 2 online databases → 2 tables
- Objective: select URLs where ID = ID retrieved in last step

Album

Logo/bandinfo



In conclusion, query has 5 steps:

Target	Content	Input	Output/Mid product
1 Query img	Obj rec in FRCNN	Query img	Cropped RoIs (O)
1 Crop	VLAD extraction*	Crop	SIFT(VLAD) descriptor of crop (M)
1 Crop	Instance matching	SIFT(VLAD) of crop	Matched img IDs (M)
1 Crop	Info retrieval	Matched img IDs	(temp) Possible band URLs (M)
1 Query img	Info retrieval	URLs of each crop	All possible band URLs (O)

Initial End-to-end Test

- Precision of object recognition and overall accuracy were calculated wrong in the mid-point presentation, actual values are lower.

Item	Remark	Number
Album ground truth image set		500
Logo ground truth image set		500
Test images		155
All crops	Crops = RoIs written to file	426
Correct crops	Crops containing album or logo	316
Matched crops	Correct band information retrieved	212
Recall of object recognition	Recall = correct RoIs/correct RoIs+RoIs haven't been recognized (multiple crops on one object is counted once)	159/192 = 83.68
Precision of object recognition	Precision = correct crops/(correct+wrong) crops	170/426 = 39.91
Overall accuracy		33.40

osition

better

Sem 2: Optimization

1. Network Optimization

- Train a new FRCNN -> Finetune pretrained ImageNet FRCNN
- Add cross validation
- Add data augmentation
- Human eval -> comprehensive mAP calculation
- Generalization ability: balance between underfitting/overfitting

1. Network Optimization

- Train a new FRCNN -> Finetune pretrained ImageNet FRCNN
  - FRCNN = Resnet50 conv layers + RPN + Resnet50 classifier
  - Sem 1 used self-trained weights for conv, RPN/classifier trained from zero
  - Sem 2 used self-trained weights for conv/classifier, pretrained weights for RPN, then finetuned
- Add cross validation
- Add data augmentation
- Human eval -> comprehensive mAP calculation
- Generalization ability: balance between underfitting/overfitting

1. Network Optimization

- Train a new FRCNN -> Finetune pretrained ImageNet FRCNN
- Add cross validation
  - Matriculated with the source code, found out the cross validation section was dysfunctional, modified
- Add data augmentation
- Human eval -> comprehensive mAP calculation
- Generalization ability: balance between underfitting/overfitting
  - Diagnostic stats/best weights saved from mAP calculation whenever loss decreases for ploting training loss vs. test loss

1. Network Optimization

- Train a new FRCNN -> Finetune pretrained ImageNet FRCNN
- Add cross validation
- Add data augmentation
  - Introduced noise
- Human eval -> comprehensive mAP calculation
- Generalization ability: balance between underfitting/overfitting
  - Diagnostic stats/best weights saved from mAP calculation whenever loss decreases for ploting training loss vs. test loss

1. Network Optimization

- Train a new FRCNN -> Finetune pretrained ImageNet FRCNN
- Add cross validation
- Add data augmentation
- Human eval -> comprehensive mAP calculation
  - Sem 1: used OpenCV imshow to check the RoIs, manually counted, bound to error
  - Sem 2: wrote script for auto calculation
- Generalization ability: balance between underfitting/overfitting

1. Network Optimization

- Train a new FRCNN -> Finetune pretrained ImageNet FRCNN
- Add cross validation
- Add data augmentation
- Human eval -> comprehensive mAP calculation
- Generalization ability: balance between underfitting/overfitting
  - Diagnostic stats/best weights saved from mAP calculation whenever loss decreases for ploting training loss vs. test loss

Trials after Adjusting Network Configurati

170/426 = 39.91

33.40

Trials	AP album	AP logo	mAP	loss	Rank
finetune + cross val + final mAP	0.875773854135	0.685576560585	0.78067520736	1.34752395183	2
finetune + aug + final mAP	0.75902475123	0.696653130193	0.727838940712	2.66465598715	6
finetune + cross val + aug + final mAP	0.871920340588	0.667815494874	0.769867917731	1.42401295861	4
finetune + cross val + per optimal mAP + gen/plt (best)	0.893460460836	0.727167063276	0.810313762056	1.2770022666	1
finetune + cross val + per optimal mAP + gen/plt (2nd best)	0.814064204169	0.733884851467	0.773974527818	1.37275927728	3
finetune + cross val + aug + per optimal mAP + gen/plt (best)	0.846656380608	0.645819069931	0.74623772527	1.55319615773	5
finetune + cross val + aug + per optimal mAP + gen/plt (2nd best)	0.727294211281	0.688764544586	0.708029377933	2.89686328396	7

(Opt) Hyperparameter Optimization

- Python lib: hyperopt, idea: random search
- Hyperspace, Fmin
- Implementation: training 20 epochs to save time, output config in JSON, retraining
- Choice of hyperparameters: lr, decay, kernel size, anchor box scale
- Problem and insight

- Conclusion: after retraining, best model with 0.81 accuracy

Demo: Re-eval of FRCNN

2. Problem on Feature Fusion

- Inappropriate input img size (224,224) -> (600,?)/(?,600) keep the ratio

Step 1, branch 1	SIFT	a. Describe SIFT features on all training images b. Kmeans clustering on all SIFT descriptors from all training images to build visual dictionary c. Calculate VLAD vector for each image based on visual dictionary
Step 1, branch 2	CNN	d. Extract conv5 feature as object level feature e. Extract conv4 feature as column feature • Misunderstanding of column feature/FC feature
Step 2	Concatenation of CNN feature and VLAD descriptor	
Step 3	Index ball tree for searching	

Trials after Changing Input Img Size

	Album		Logo	
	Dimension	Accuracy	Dimension	Accuracy
SIFT(VLAD)	65536->65536	0.8877551	65536->65536	0.39215686
CNN object level feature (original weights)	2048->8192	0.41326531	2048->28672	0.33529412
CNN object level feature (model 1)	2048->8192	Fail	2048->28672	Fail
CNN object level feature (model 2)	2048->8192	Fail	2048->28672	Fail
SIFT(VLAD) + CNN object level feature (original weights)	67584->73728	0.41326531	67584->94208	0.33529412
SIFT(VLAD) + CNN object level feature (model 1)	67584->73728	Fail	67584->94208	No trial
SIFT(VLAD) + CNN object level feature (model 2)	67584->73728	Fail	67584->94208	No trial

- Higher overall than Sem 1.
- SIFT on album could be considered as a successful case.

Trials with FC Features

	Accuracy
FC	0.18823529
SIFT(VLAD) + FC	0.12941176
SIFT(VLAD) with PCA + FC	0.22745098

- Sub-ideal result

New Mechanism: VLAD on CNN conv5

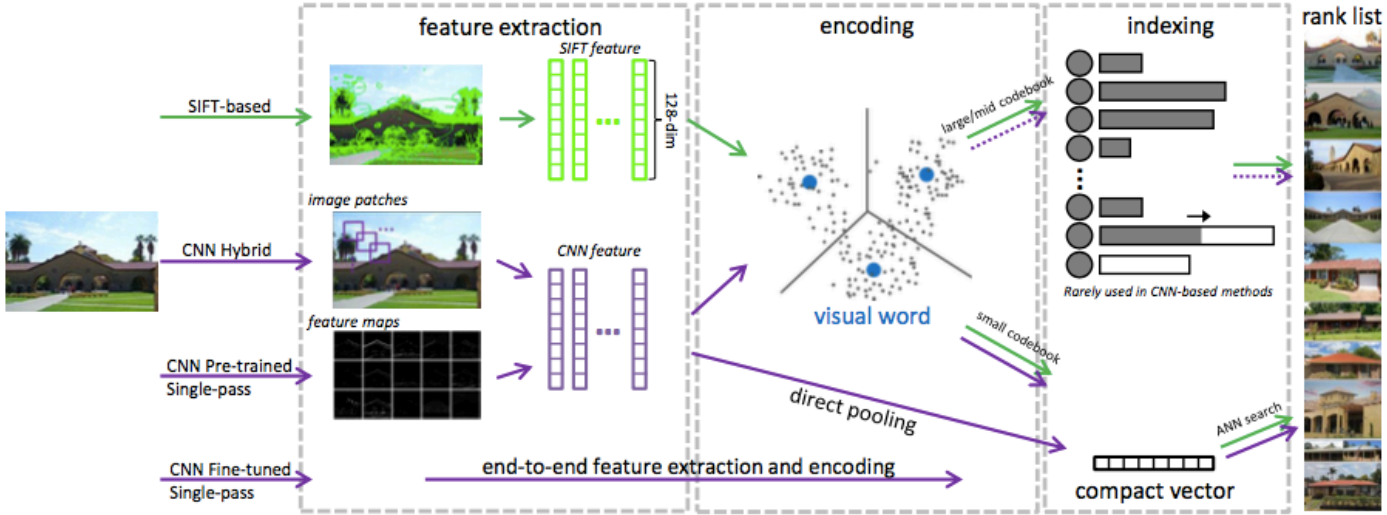


Fig. 2: A general pipeline of SIFT- and CNN-based retrieval models. Features are computed from hand-crafted detectors for SIFT, and densely applied filters or image patches for CNN. In both methods, under small codebooks, encoding/pooling is employed to produce compact vectors. In SIFT-based methods, the inverted index is necessary under large/medium-sized codebooks. The CNN features can also be computed in an end-to-end way using fine-tuned CNN models.

Trials with VLAD on CNN conv5

	Accuracy
CONV5(VLAD) on hard images	0.33529412
SIFT(VLAD) + CONV5(VLAD) on hard images	0.44313725
SIFT(VLAD) + CONV5(VLAD) on FRCNN RoIs	0.64705882

- At least… it's better.

(Opt) New Mechanism: Canonical Correlation Analysis

- Sub-ideal results.

Demo: Re-eval of Logo Matching

- Conclusion:
- SIFT(VLAD) + CNN conv5(VLAD) for logos, balancing the time consumption and accuracy.
- SIFT(VLAD) for albums.
- Best mAP ≈0.76.

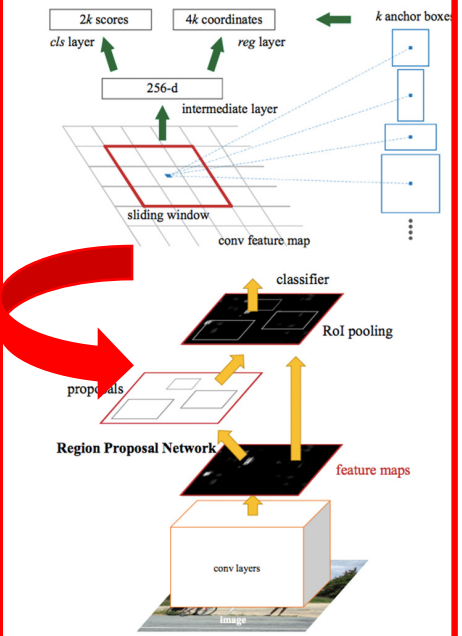
- **End-to-end accuracy ≈0.65**, reasonable considering FRCNN and CNN each have an accuracy around 0.80.

Finalized Server Pipeline

1. Object Recognition – FRCNN

Detection – RPN

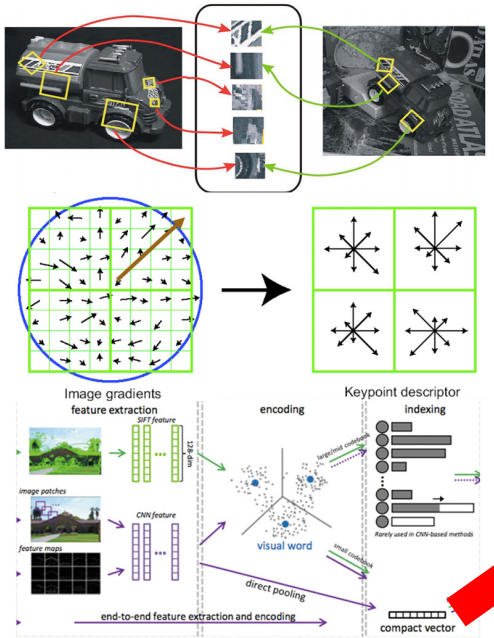
Classification – CNN classifier



2. Feature Extraction

SIFT on albums

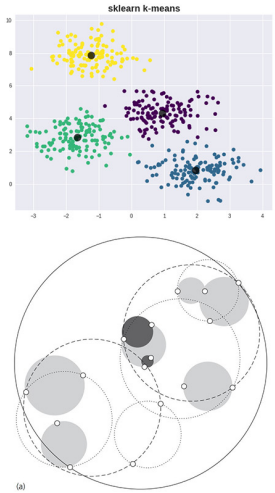
SIFT + conv5 on logos



3. Instance Matching

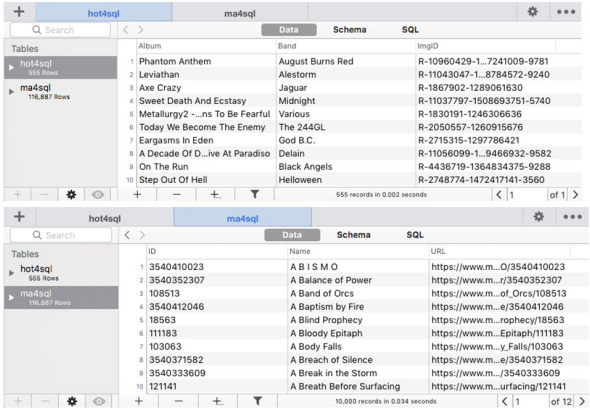
Words – Kmeans

Indexing – Ball Tree

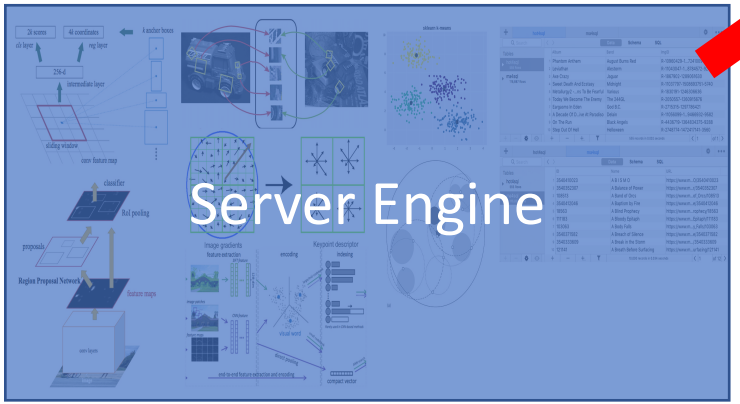


4. Information Retrieval

Database



Sem 2: Mobile Development
(Opt) MVC Design

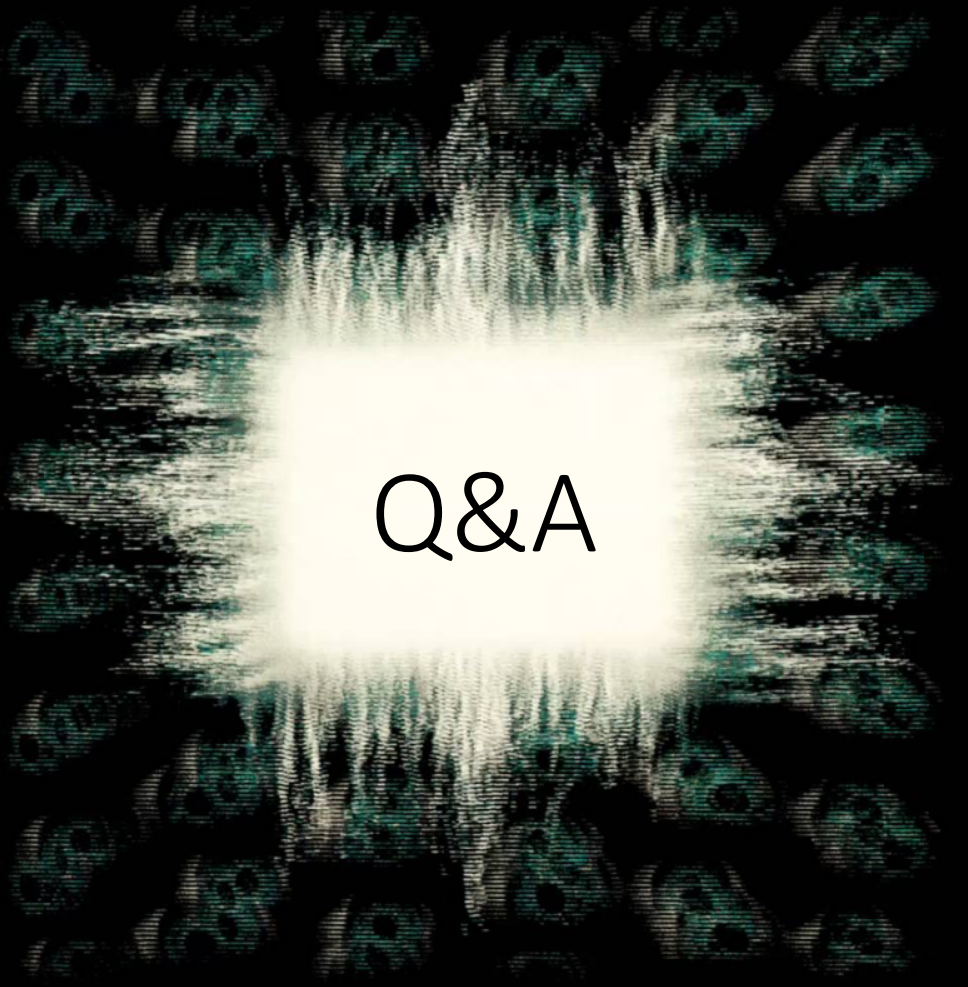


Retrieved Information

JSON

Network

Demo: the APP


Q&A