Building Facebook’s visual cortex

Anmol Kalia

fb.me/fbcortex
Instagram explore ranking
Compact representations as sparse features
Violating Content Classifiers
Classes as features in multi-modal fusion
$O(\text{millions})$

Videos uploaded to our platforms daily
O(billions)

Images uploaded to our platforms daily
Consideration 1: Data is resource intensive

Annotating datasets is resource intensive - ImageNet-1K-ILSVRC2012 today:

1.43M Images in dataset [1]
* 3 Multi-review with 3 annotators
* $0.12 Price per-image - multi-label [3]

~ $0.5M

Consideration 2: Dynamic demand

Classifier for “mask” or “ice bucket challenge”?
Consideration 3: Continuous improvement
Consideration 4: Efficiency

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Model Size (# params)</th>
<th>Forward pass latency (ms on Intel Skylake, 18 core, 64 GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>25M</td>
<td>~ 70 ms</td>
</tr>
<tr>
<td>ResNeXt-101-32x4-48</td>
<td>43 - 829M</td>
<td>~ 150 ms</td>
</tr>
<tr>
<td>Faster-RCNN-Shuffle</td>
<td>6M</td>
<td>~ 600 ms</td>
</tr>
<tr>
<td>ResNeXt-3D-101</td>
<td>21M</td>
<td>~ 4 sec</td>
</tr>
</tbody>
</table>

1 billion images run on 1 machine sequentially through ResNeXt-101:

1736 days to process ~1 day of photos
Data collection resources

Efficiency

Continuous improvement

Dynamic demand
Multi-objective optimization problem

Model size

Speed - training and inference

Accuracy of target task

Energy
Idea 1: Shared backbone

- Policy violating
- Not policy violating
- MLP + Softmax
- Image backbone A
- Image backbone B
- Image backbone C

2048 dim embedding
256 bit quantized hash
kNN indices
Idea 1: Shared backbone

Custom classes

MLP + Softmax

2048 dim embedding

Residual Net

kNN indices

256 bit quantized hash [1]
Idea 2: Focus on efficient backbone architectures

Evolution of Video backbone architecture

- Use 3D conv to model appearance & motion together
- Use Residual network as backbone
- Factorize 3D conv into spatial & temporal components
- Factorize 3D conv into channel and spatiotemporal interactions
Idea 3: Develop efficiency techniques - Octave Convolution

- Low frequency (Global structure)
- High frequency (Fine details - edges)
- Low frequency (Global structure)

Idea 3: Invest in efficiency techniques - Octave Convolution

Idea: Store and process feature maps that vary spatially slower at a lower spatial resolution reducing both memory and computation. Drop-in Convolution operator giving \textbf{40\% drop in GFLOPs and 50\% drop in latency for ResNet-50}

Continuous improvement

Data collection resources

Efficiency

Dynamic demand
Importance of backbone

- Custom classes
  - MLP + Softmax
  - 2048 dim embedding
  - kNN indices
  - 256 bit quantized hash [1]

- Cat, Dog, Baby, Towel, Bear
- Fine-tuned
- Residual Net
  - N
  - N-1
  - 2
  - 1

MLP + Softmax
Reducing supervision for pre-trained network = less resource intensive annotation

- ImageNet pre-trained
- Kinetics pre-trained

Fully supervised
("clean labels")
Reducing supervision for pre-trained network = less resource intensive annotation

- Fully supervised ("clean labels")
- Weakly supervised ("noisy labels")
- Semi supervised ("large unlabelled + small labeled")
- Self supervised ("no labels")

ImageNet pre-trained
Kinetics pre-trained
Exploring the Limits of Weakly Supervised Pretraining

Dhruv Mahajan  Ross Girshick  Vignesh Ramanathan  Kaiming He  Manohar Paluri  Yixuan Li  Ashwin Bharambe  Laurens van der Maaten

Facebook
Weakly supervised - insights

- Fully ("labeled")
- Weakly ("noisy")
- Semi ("large unlabeled, small labeled")
- Self ("no labels")

Non-visual

#love

Wrong label

#persiancat #cat

Missing labels

#cat
Goal: Pre-train to predict hashtags and then evaluate transfer to image classification (ImageNet-1K)

| Dataset                  | 3.5B public Instagram images, 17K hashtags  
|                         | - Previous largest dataset: JFT-300M       |
| Pre-processing          | Replicate images from low frequency tags   
|                         | De-dup labels based on WordNet synset hierarchy |
| Loss                    | Treat as multi-label, cross-entropy between softmax and vector of k non-zero entries each set to 1/k corresponding each hashtag |
| Training                | 336 GPUs (42 machines) = 22 days to train  |
| Architecture            | ResNeXt-101-32x{4, 8, 16}                  |
Performance of target task (logistic regression on FC) increases with pre-training image set size
Transfer learning performance bottlenecked by model capacity

Source task:
- ImageNet (target = source)
- Instagram (940M, 1.5k tags)
Reducing supervision for pre-trained network = less resource intensive annotation

- Fully supervised ("clean labels")
- Weakly supervised ("noisy labels")
- Semi supervised ("large unlabelled + small labeled")
- Self supervised ("no labels")
- **Consideration 1:** Hashtag data not accessible to everyone

- **Consideration 2:** Does not leverage the large amount of unlabelled data
  - Mapping it to Instagram, account for 89% of media without hashtags

- **Consideration 3:** Capacity of the models hard to deploy
  - ResNeXt-101-32x48 has 33x more parameters than ResNet-50
Billion-scale semi-supervised learning for image classification

I. Zeki Yalniz  Hervé Jégou  Kan Chen  Manohar Paluri  Dhruv Mahajan

Facebook AI
Semi supervised - setup

Labeled data (ImageNet-1K)

Fully ("labeled")
Weakly ("noisy")
Semi ("large unlabeled, small labeled")
Self ("no labels")

Larger capacity model (Teacher, ResNeXt-101)

Target model (Student, ResNet-50)

Unlabeled data

Pick P concepts per image and then rank top-K images per concept

Pre-trained

Fine-tune

Top scoring examples
Value of unlabelled data - accuracy on ImageNet-1K of ResNet-50 (student)
<table>
<thead>
<tr>
<th>Method</th>
<th># params</th>
<th>Fully-supervised</th>
<th>Semi-supervised</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50</td>
<td>25M</td>
<td>70.6%</td>
<td>79.1%</td>
</tr>
<tr>
<td>ResNeXt-50-32x4</td>
<td>25M</td>
<td>77.6%</td>
<td>79.9%</td>
</tr>
<tr>
<td>ResNeXt-101-32x8</td>
<td>88M</td>
<td>79.1%</td>
<td>81.2%</td>
</tr>
<tr>
<td>ResNeXt-101-32x48</td>
<td>829M</td>
<td>79.8%</td>
<td>-</td>
</tr>
</tbody>
</table>

Compute saved:
- Fully-supervised (RX101-32x8) ~ Semi-supervised (R50) - 4x less params
- Fully-supervised (RX101-32x48) ~ Semi-supervised (RX101-32x4) - 33x less params
Reducing supervision for pre-trained network = less costly annotation

- Fully supervised ("clean labels")
- Weakly supervised ("noisy labels")
- Semi supervised ("large unlabelled + small labeled")
- Self supervised ("no labels")
Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles

Mehdi Noroozi and Paolo Favaro

Institute for Informatik
University of Bern
{noroozi, paolo.favaro}@inf.unibe.ch
Self-supervised setup

Fully (“labeled”)  Weakly (“noisy”)  Semi (“large unlabeled, small labeled”)  Self (“no labels”)

Scaling and Benchmarking Self-Supervised Visual Representation Learning

Priya Goyal  Dhruv Mahajan  Abhinav Gupta*  Ishan Misra*
Facebook AI Research
Scaling pre-text task data & model capacity
<table>
<thead>
<tr>
<th>Method</th>
<th>Top-1 accuracy of R-50 on ImageNet-1K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fully supervised</td>
<td>76.4%</td>
</tr>
<tr>
<td>Weakly supervised</td>
<td>78.2%</td>
</tr>
<tr>
<td>Semi supervised</td>
<td>79.1%</td>
</tr>
<tr>
<td>Self supervised (Jigsaw)</td>
<td>45.4%</td>
</tr>
</tbody>
</table>

Best ResNet-50 ImageNet-1K Top-1 accuracy across all methods

Production state

- Fully supervised ("clean labels")
- Weakly supervised ("noisy labels")
- Semi supervised ("large unlabelled + small labeled")
- Self supervised ("no labels for pretext tasks")

Images

Video*
Continuous improvement

Data collection resources

Efficiency

Dynamic demand
Making upgrades easy

Contract on compatibility support
Predictable push with N-1 backwards compatibility

Embedding backwards compatibility via distillation (L2 loss on embedding) [1]

Concepts always added ($\text{concepts2} \supseteq \text{concepts1}$) + impersonating old calibrated score by lookup table

Data collection resources
Continuous improvement
Efficiency
Dynamic demand
Fast classifiers for concepts

**2048 dim embedding**

- Cat
- Dog
- Pillow
- Towel
- Bear

**MLP + Softmax**

**Custom classes**

**Fine-tuned**

**N-1**

**N**

**Residual Net**

**256 bit quantized hash**

**O (days)**

**kNN indices**

**O (hours)**
Fast classifiers for concepts

Need platform to quickly train linear classifiers:
- Reproducible & automated
- Monitored
- $O$(hours) to bring online
Manifold sampling using FAISS [1] indices

Active Labeling
Annotation Platform

Data Management & privacy
Feature extraction

Training workflows
Evaluation
Inference

Concept drift & model unit tests

Tests

O (hours)

O (minutes)

Data collection resources
Reduce supervision

Continuous improvement
Compatibility contracts for fast upgrades

Efficiency
Attack from various angles - operators, architectures

Dynamic demand
O(hours) workflow for trending concepts
A few pointers...

Costly data
- Image weakly supervised
  - **Pre-trained model:** [https://github.com/facebookresearch/WSL-Images](https://github.com/facebookresearch/WSL-Images)
- Video weakly supervised
  - **Pre-trained model:** [https://github.com/facebookresearch/VMZ](https://github.com/facebookresearch/VMZ)
- Image semi-supervised
  - **Pre-trained model:** Coming soon!
- Image self-supervised
  - **Benchmark:** [https://github.com/facebookresearch/fair_self_supervision_benchmark](https://github.com/facebookresearch/fair_self_supervision_benchmark)

Efficiency
- Architecture Search
- Video architecture evolution
- Octave Convolution
  - **Code:** [https://github.com/facebookresearch/OctConv](https://github.com/facebookresearch/OctConv)
- Optimized kernels
  - **FBGEMM (server):** [https://github.com/pytorch/FBGEMM](https://github.com/pytorch/FBGEMM)
  - **QNNPACK (mobile):** [https://github.com/facebookresearch/QNNPACK](https://github.com/facebookresearch/QNNPACK)
- Catalyzer hash
  - **Code:** [https://github.com/facebookresearch/spreadingvectors](https://github.com/facebookresearch/spreadingvectors)

Dynamic Demand
- **FAISS**
  - **Code:** [https://github.com/facebookresearch/faiss](https://github.com/facebookresearch/faiss)
Thank you!