

Getting more for less out of your CNNs

Andrew D. Bagdanov

Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Firenze
andrew.bagdanov AT unifi.it



ICATCES 2019 – Alanya, Turkey – 26 April 2019

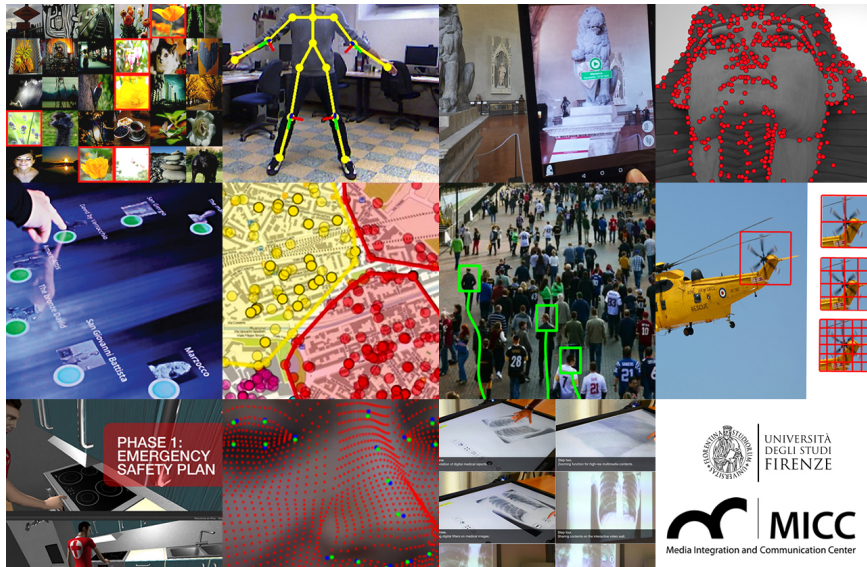
- 1 A Brief Introduction
- 2 Context: The New State-of-the-Art in Computer Vision
- 3 Self-supervised Learning
- 4 Domain-adaptive Deep Network Compression
- 5 Discussion

A Brief Introduction

Where in the world is Andy Bagdanov?





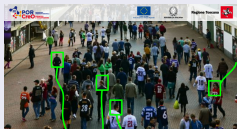
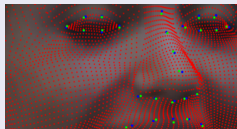
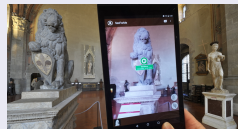


UNIVERSITÀ
DEGLI STUDI
FIRENZE

MICC
Media Integration and Communication Center

Main Research Directions

- **Computer Vision**: deep learning, object detection and recognition, person re-identification, face recognition, person counting and smart mobility.
- **3D Imaging and Vision**: 3D face and body capture, features for 3D recognition and retrieval, 3D super-resolution.
- **Multimedia**: multimedia for cultural heritage, **social museums and smart tourism**, image and video annotation and retrieval, knowledge modeling for multimedia.
- **Smart Environments**: visitor profiling and *ad hoc* content recommendation, **HCI for Cultural Heritage**.
- **Internet Applications**: social networks, profiling and content adaptation, visualization, interaction metaphors, image forensics and security.

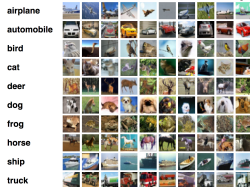
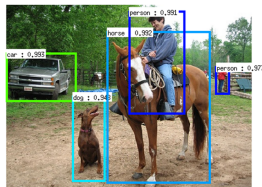
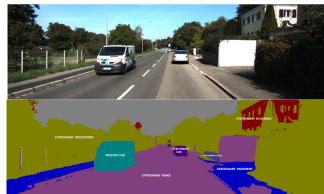


Context: The New State-of-the-Art in Computer Vision

- The computer vision field is at a **watershed** moment in its history – one that I don't believe we have ever seen the likes of.
- This is largely due to the advent of **Convolutional Neural Networks (CNNs)** as the model of choice for many vision problems.
- Since 2012, the community has been rapidly expanding and rapidly trying to come to terms with the **radical shift in thinking**.
- This talk will focus on two aspects of CNNs applied to specific types of visual recognition problems.
- To appreciate the motivations for these problems, it is useful to take a wide-angle view of state-of-the-art models for visual recognition.

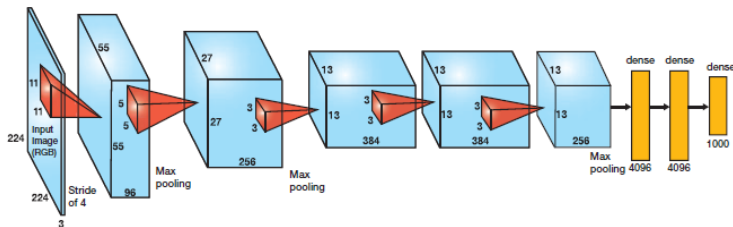
Context: visual recognition

- **Visual object recognition** is the process of mapping images to distributions over **object class labels** or **semantic properties**.
- As a category of problems, visual recognition includes:
 - **Object recognition**: given an image with one or more instances of known classes, determine **which classes are present**.
 - **Object detection**: given an image with one or more instances of known classes, determine which classes are present and **where** in the image.
 - **Object segmentation**: given an image with one or more instances of known classes, **label each pixel with a class name**.
 - **Visual regression**: given an image, estimate one or more **continuous, semantic** properties.



Context: the anatomy of a CNN

- The computer vision field is at a **watershed** moment in its history – one that I don't believe we have ever seen the likes of.
- **Convolutional Neural Networks (CNNs)** are state-of-the-art in visual recognition.
- They took our community by storm in 2012 as an **end-to-end trainable model**, largely due to:
 - The **availability** of **enormous** amounts of **annotated** data.
 - Modern **GPUs**, which make convolutions super fast.
 - **Decades** of persistent theoretical development.
- This talk will focus on two aspects of CNNs applied to specific types of visual recognition problems.



[Krizhevsky, NIPS+ECCV 2012]

Not all sunshine and lollipops

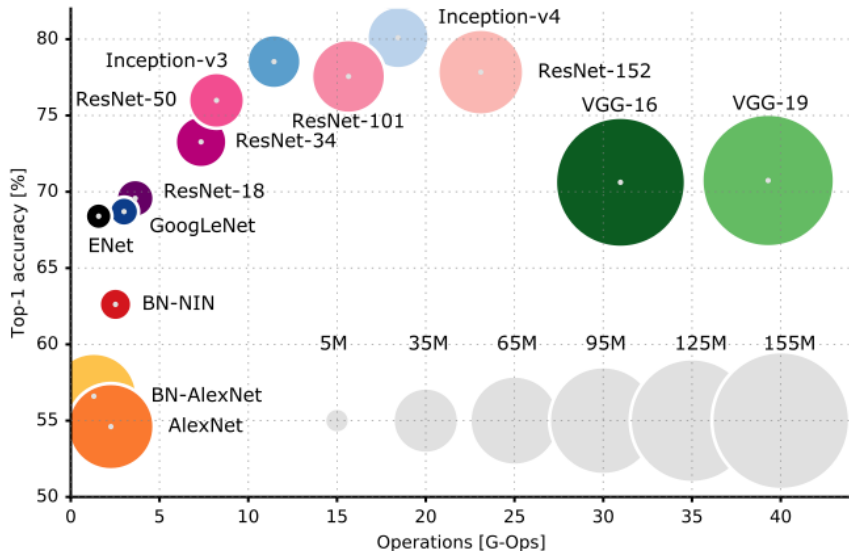
- Despite their widespread **adoption** and **success**, effective application of CNNs to real-world problems is not straightforward.
- Two common problems that arise are due to:
 - The need for **massive amounts of annotated data**. Training modern CNNs require **millions** of annotated image examples.
 - The **large GPU memory footprint**. Modern CNN architectures have **hundreds of millions of parameters**, occupying **gigabytes** of **GPU memory**.

Today's talk

- These problems are related, and I will talk about two ways to address them:
 - **Self-supervised learning**: derive training signal **for free** from unlabeled data via **proxy tasks** related to the primary one.
 - **Domain adaptive network compression**: when you know what domain you need to apply the network to, **compress it** using that knowledge.

Self-supervised Learning

CNNs are really big



[Canziani 2017]

CNNs are really big

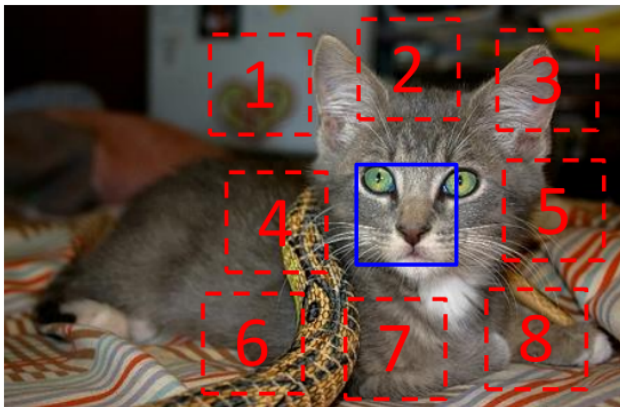
- One of the first observations one can make about CNNs is that they have a **HUGE** number of parameters.
- Even modestly-sized, state-of-the-art networks have on the order of *150 million trainable parameters*.
- Fitting such models of course requires **massive** amounts of **labeled training data**.



[Russakovsky 2015]

- Such data can be **enormously expensive** to collect.
- For basic image recognition problems (e.g. **cats** versus **dogs**), labeled data is relatively easy to **crowdsource**.
- For other problems, the annotation task is significantly more tedious and requires careful **supervision** and **annotator corroboration**.
- This, of course, translates into **higher annotation costs**.
- **Self-supervision** offers the prospect of synthesizing training signal **for free** and has been applied to representation learning for object recognition:
 - **Learning to Count**: learning to count unsupervised visual primitives in a consistent way [Noroozi 2017].
 - **Patch Colorization**: learn representations via colorization of grayscale patches in images [Larsson 2017].
 - **Local Context Prediction**: use prediction of local patch context as a proxy task for representation learning [Doersch 2016].

Example: Local Context Prediction



$$X = (\text{crops of boxes 4 and 5}); Y = 3$$

[Doersch 2017]

- **Image Quality Assessment (IQA)** is the task of estimating an **objective** image quality measurement for an input image.
- This measure should reflect the judgment of **human experts**.
- It has a broad range of practical applications, and requires a **huge** number of human judgments *per-image* to establish the “objective” baseline.
- One IQA dataset averages the IQA estimates of **100** human experts.
- Consequently, available datasets tend to be **tiny** (hundreds of images).



score: 85 $\in [0,100]$

- Another problem for which annotation is **tedious** and **expensive** is **crowd counting**.
- Given an image, the task is to estimate the **number of persons** present in the scene.
- Given the difficulty of annotation, most crowd counting datasets have **at most** around 1000 labeled images.

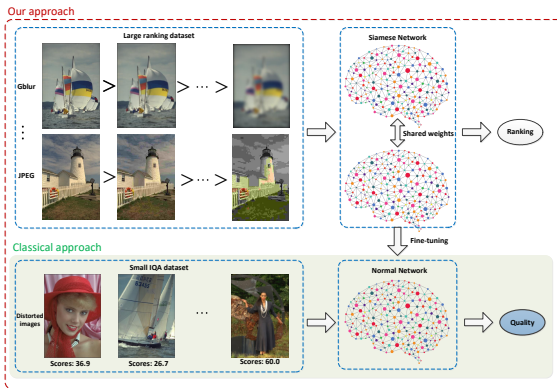


Our observation

- For many **niche** problems, we can define much more specific and useful proxy tasks for self-supervised learning.
- These domain-specific proxy tasks allow us to address the lack-of-data problem in application areas where it is **much more acute**.

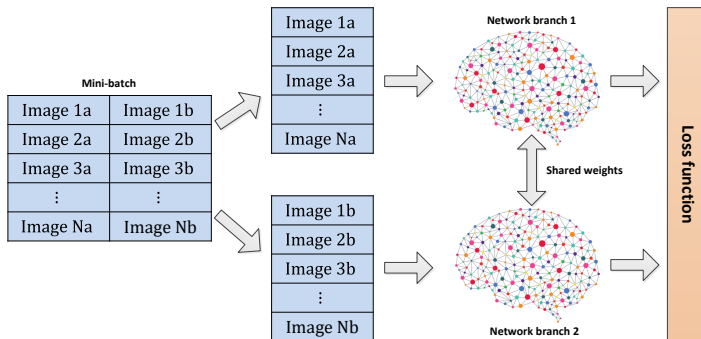
Our approach: learning from rankings

- What we propose is to create a **proxy ranking task** for which we can **automatically** generate training signal.
- After **pre-training** a CNN using this proxy task, we then **transfer** the weights to a CNN that estimates **absolute** IQA on images.
- This new CNN is **fine-tuned** using limited, labeled IQA data.



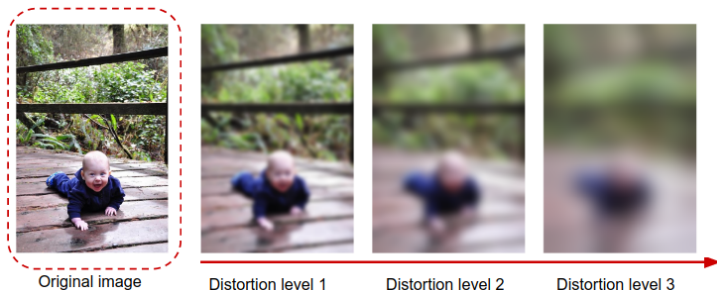
Our approach: ranking image quality

- Our network is conceptually a **Siamese** network.
- Its inputs are **pairs** of images, each are passed through a **branch** of the network.
- These branches **share all weights** (and so are really the **same**).
- The loss is computed on the **outputs** resulting from the forward pass of the images.
- In our case: if we know that the image quality of image *a* is **higher** than that of image *b*, we only require the network outputs to reflect this.



Our approach: image pair generation

- For IQA, it is easy to **synthetically** generate training pairs.
- Given a high-quality reference images, we **generate** distorted pairs using known image degradations
- There are about 30 **standard image distortions**: Gaussian blur, JPEG compression, Gaussian noise, fast fading, etc.
- We don't know the **absolute** image quality of a distorted images, but we do know the image quality **ranking**.



Our approach: the ranking loss



- We use $\hat{c}(I)$ to denote the **output of the network** on image I .
- Assuming the quality of I_i is **higher** than that of I_j when $i < j$, we define a **ranking hinge loss** with margin ε :

$$L_{rank} = \max(0, \hat{c}(I_j) - \hat{c}(I_i) + \varepsilon)$$

- The gradient is of this loss with respect to network parameters θ is:

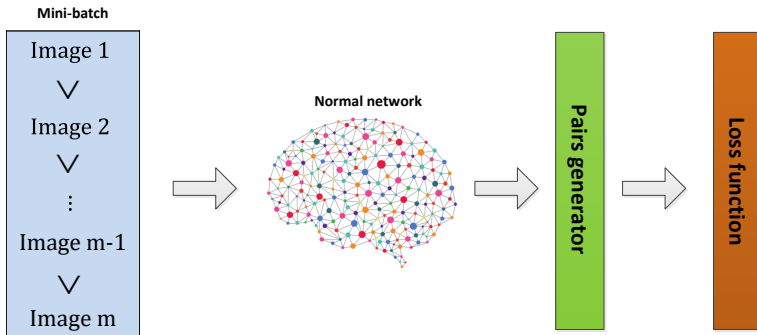
$$\nabla_{\theta} L_r = \begin{cases} 0 & \text{if } \hat{c}(I_j) - \hat{c}(I_i) + \varepsilon \leq 0 \\ \nabla_{\theta} \hat{c}(I_j) - \nabla_{\theta} \hat{c}(I_i) & \text{otherwise} \end{cases}$$

- As long as we **carefully** construct our minibatches so that we know the relative quality ranking, we can calculate gradients and backprop away.
- After training to **rank**, we fine-tune on a **small labeled training set** using $(c(I_i))$ is the **groundtruth** IQA for image i):

$$L_{iqa} = \frac{1}{M} \sum_{i=1}^M (c(I_i) - \hat{c}(I_i))^2$$

Our approach: fast Siamese backprop

- Another contribution of this work is a technique for **significantly** speeding up Siamese network training.
- It avoids redundant forward passes of images through the network by generating pairs at the **end** rather than at the beginning.
- This also makes applying (and learning over) **multiple** distortions much faster.



Datasets

- The **LIVE dataset**: 808 images generated from 29 original images by distorting with Gaussian blur (GB), Gaussian noise (GN), JPEG compression (JPEG), JPEG2000 compression (JP2K) and fast fading (FF).
- The **TID2013 dataset**: 25 reference images with 3000 distorted images from 24 different distortion types at 5 degradation levels. Distortion types include a range of noise, compression, and transmission artifacts.

Training

- We use the **VGG16** network pretrained on ImageNet as the **backbone**.
- We randomly sample sub-images from the original high resolution images before synthetically distorting.
- We use **Stochastic Gradient Descent** for both training phases.

Experimental results: us versus them

- We outperform the state-of-the-art on the TID2013 dataset:

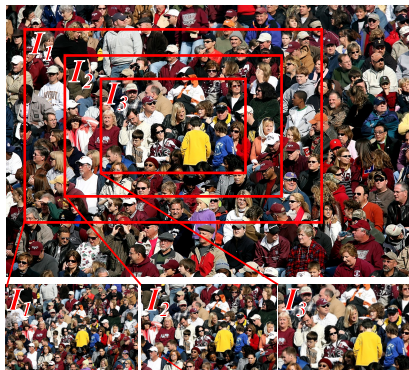
Method	#01	#02	#03	#04	#05	#06	#07	#08	#09	#10	#11	#12	#13
BLIINDS-II [23]	0.714	0.728	0.825	0.358	0.852	0.664	0.780	0.852	0.754	0.808	0.862	0.251	0.755
BRISQUE [19]	0.630	0.424	0.727	0.321	0.775	0.669	0.592	0.845	0.553	0.742	0.799	0.301	0.672
CORNIA-10K [40]	0.341	-0.196	0.689	0.184	0.607	-0.014	0.673	0.896	0.787	0.875	0.911	0.310	0.625
HOSA [36]	0.853	0.625	0.782	0.368	0.905	0.775	0.810	0.892	0.870	0.893	0.932	0.747	0.701
Baseline	0.605	0.468	0.761	0.232	0.704	0.590	0.559	0.782	0.639	0.772	0.817	0.571	0.725
RankIQA	0.891	0.799	0.911	0.644	0.873	0.869	0.910	0.835	0.894	0.902	0.923	0.579	0.431
RankIQA+FT	0.667	0.620	0.821	0.365	0.760	0.736	0.783	0.809	0.767	0.866	0.878	0.704	0.810
Method	#14	#15	#16	#17	#18	#19	#20	#21	#22	#23	#24	ALL	
BLIINDS-II [23]	0.081	0.371	0.159	-0.082	0.109	0.699	0.222	0.451	0.815	0.568	0.856	0.550	
BRISQUE [19]	0.175	0.184	0.155	0.125	0.032	0.560	0.282	0.680	0.804	0.715	0.800	0.562	
CORNIA-10K [40]	0.161	0.096	0.008	0.423	-0.055	0.259	0.606	0.555	0.592	0.759	0.903	0.651	
HOSA [36]	0.199	0.327	0.233	0.294	0.119	0.782	0.532	0.835	0.855	0.801	0.905	0.728	
Baseline	0.187	0.308	0.062	0.546	0.383	0.500	0.420	0.700	0.611	0.573	0.742	0.612	
RankIQA	0.463	0.693	0.321	0.657	0.622	0.845	0.609	0.891	0.788	0.727	0.768	0.623	
RankIQA+FT	0.512	0.622	0.268	0.613	0.662	0.619	0.644	0.800	0.779	0.629	0.859	0.780	

- We even outperform **full-reference** IQA on standard benchmarks:

	SROCC	JP2K	JPEG	GN	BLUR	FF	ALL
FR-IQA	PSNR	0.87	0.885	0.942	0.763	0.874	0.866
	SSIM [35]	0.939	0.946	0.964	0.907	0.941	0.913
	FSIM [17]	0.97	0.981	0.967	0.972	0.949	0.964
	DCNN [17]	–	–	–	–	–	0.975
	DIVINE [21]	0.913	0.91	0.984	0.921	0.863	0.916
NR-IQA	BLIINDS-II [23]	0.929	0.942	0.969	0.923	0.889	0.931
	BRISQUE [19]	0.914	0.965	0.979	0.951	0.887	0.94
	CORNIA [40]	0.943	0.955	0.976	0.969	0.906	0.942
	CNN [11]	0.952	0.977	0.978	0.962	0.908	0.956
	SOM [43]	0.947	0.952	0.984	0.976	0.937	0.964
	DNN [2]	–	–	–	–	–	0.960
	RankIQA+FT	0.970	0.978	0.991	0.988	0.954	0.981

Our approach: crowd counting

- We have extended this approach to the **crowd counting** problem.
- It uses the observation that **sub-images** must contain the same number or fewer persons than the **super-image**.
- Key to this approach is a **multi-task** training objective.

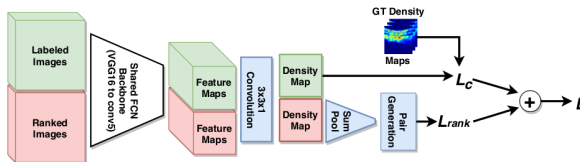
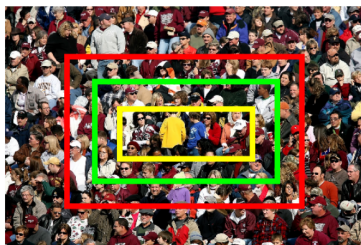
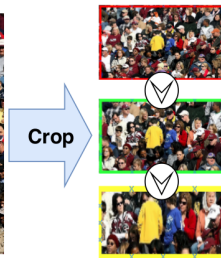


$$C(I_1) \geq C(I_2) \geq C(I_3)$$

[X. Liu, J. van de Weijer, A. D. Bagdanov, "Self-Supervised Crowd Counting in Images by Learning to Rank." CVPR 2018.]

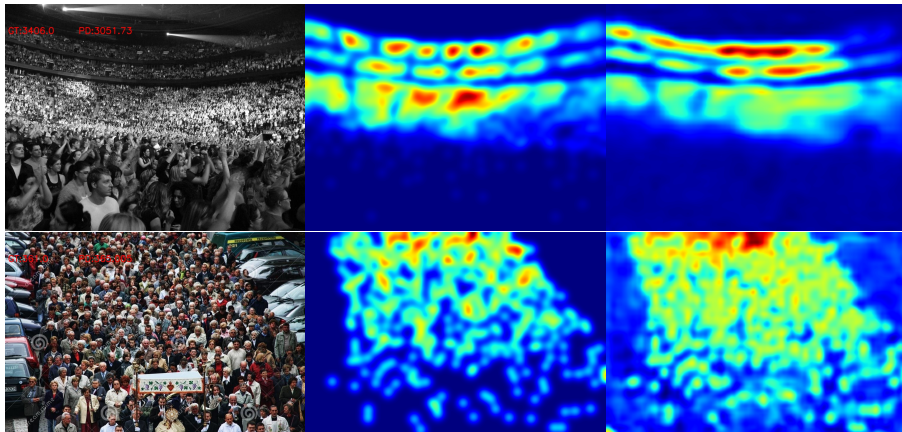
Our approach: crowd counting

- Key to applying this approach to crowd counting was extending the architecture to a **multi-task loss**.
- With this we can **simultaneously train** the self-supervised and fully-supervised tasks.
- This helps guarantee that the network is counting **useful** features, instead of just random correlations (loosely speaking).



Our approach: crowd counting

- The result is a **high-quality density map** that can be **integrated** to estimate counts:



- An ablation study demonstrates the effectiveness of our approach:

Method	Split 1	Split 2	Split 3	Split 4	Split 5	Ave MAE
Basic CNN	701.41	394.52	497.57	263.56	415.23	454.45
+ Pre-trained model	570.01	350.63	334.89	184.79	202.41	328.54
+ multi-scale	532.85	307.43	266.75	216.96	216.35	308.06
Ranking+FT	552.68	375.38	241.28	211.66	247.70	325.73
Multi-task (Random)	462.71	345.31	218.71	226.44	210.19	292.67
Multi-task (Hard)	460.35	343.91	208.23	221.75	205.57	287.96
Multi-task (Ours)	443.68	340.31	196.76	218.48	199.54	279.60

- And our results are comparable to the state-of-the-art:

Method	Part A		Part B	
	MAE	MSE	MAE	MSE
Cross-scene [34]	181.8	277.7	32.0	49.8
MCNN [10]	110.2	173.2	26.4	41.3
Switching-CNN [11]	90.4	135.0	21.6	33.4
CP-CNN [33]	73.6	106.4	20.1	30.1
ACSCLP [38]	75.7	102.7	17.2	27.4
CSRNet [36]	68.2	115.0	10.6	16.0
ic-CNN [41]	68.5	116.2	10.7	16.0
Ours: Multi-task (Query-by-example)	72.0	106.6	14.4	23.8
Ours: Multi-task (Keyword)	73.6	112.0	13.7	21.4

The good

- Self-supervision is an effective way to mitigate the **lack of labeled data** problem.
- **Rankings** (via a ranking hinge loss) can be used to generate training signal.

The bad and the ugly

- Many self-supervised learning approaches still require **massive** amounts of (unlabeled) data.
- Some approaches (like ours for counting) require supervision in **parallel** with self-supervision – making them **semi-supervised** in fact.

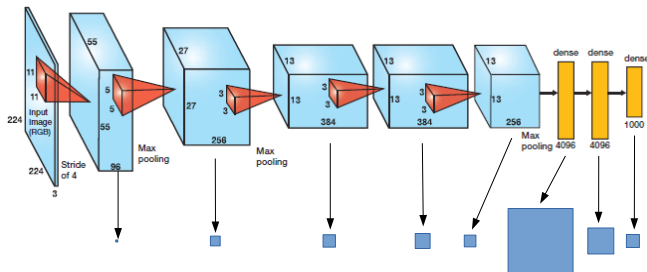
Final comments

- I like to think of self-supervised learning as a sort of **end-to-end pre-training** (as opposed to **layer-wise** pre-training).
- With a semantically rich **proxy task**, we can learn useful **representation** for hard problems.
- **Learning from Rankings** is a general framework that can be used to derive such proxies.

Domain-adaptive Deep Network Compression

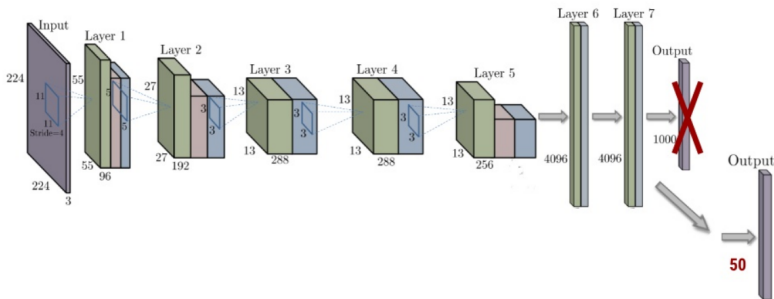
Some problems

- A lot of the training **investment** is really in the **representational learning** in the early, convolutional layers of the network.
- The dense layers require a **semantically rich** image representations in order to solve (relatively) simple classification problems.
- To appreciate the complexity **bottleneck**, it is useful to visualize the **relative** numbers of parameters per-layer in a typical CNN.
- Things get **dicey** when the network becomes **dense**: about 75% of the trainable parameters are in just the fully-connected layers of AlexNet.
- Not only does this render such networks difficult (and slow) to train, but also impossible to **deploy** in many scenarios.



CNNs in practice (fine-tuning)

- A common technique, in fact, has become to: use a model **pre-trained** on ImageNet, and **fine-tune** it for a specific task.
- **Key point:** fine-tuning a network usually requires **significantly** less labeled data.
- This is a type of **domain adaptation**: we transfer knowledge from one domain to another.
- **Question:** does the new problem require the same model complexity? Can we not **eliminate** away some of the redundancy in the CNN, rendering it more compact?

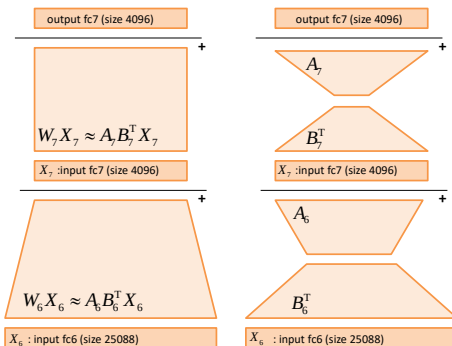


A standard approach (SVD)

- A standard approach to addressing this problem is via **compression**.
- There are a host of techniques, but a simple and effective one is to use **truncated Singular Value Decomposition**.
- That is, to **approximate** weight matrices in dense layers using a low-rank factorization: $W = USV^T$.
- The layer weights W can be approximated by keeping the k most significant singular vectors, $\hat{W} = \hat{U}\hat{S}\hat{V}^T$ where $\hat{U} \in \mathbb{R}^{m \times k}$, $\hat{S} \in \mathbb{R}^{k \times k}$, $\hat{V} \in \mathbb{R}^{n \times k}$.
- Compression happens by **replacing** the original layer by two new ones: the first with weights $\hat{S}\hat{V}^T$ and the second with weights $\hat{U}$.
- It is crucial that these two new layers contain **fewer** parameters than the original network (i.e. that $nm > (n + m)k$).

Our approach: domain adaptive decomposition

- A limitation of the SVD compression approach is that it minimizes the **Frobenius norm** of the difference between W and USV^T .
- It does not preserve the **statistics** of forward **activations** in the network.
- **Our approach**: guarantee that our **compressed** W preserves (in an average-over-training-set sense) the activations of the original W .



[Masana, Weijer, Herranz, Bagdanov, Alvarez, "Domain-adaptive deep network compression." In: *Proceedings of ICCV*, 2017.]

- Our approach also works by **decomposing** weights W into two layers such that:

$$W \approx \hat{W} = AB^T,$$

- In this decomposition $\hat{W} \in R^{m \times n}$, $A \in R^{m \times k}$ and $B \in R^{n \times k}$ – again k chosen so that $m \times n > (m + n) \times k$.
- We want the decomposition which minimizes (X is training data from the **new domain**):

$$\min_{A,B} \|Y - \hat{Y}\|_F = \min_{A,B} \|WX - AB^T X\|_F,$$

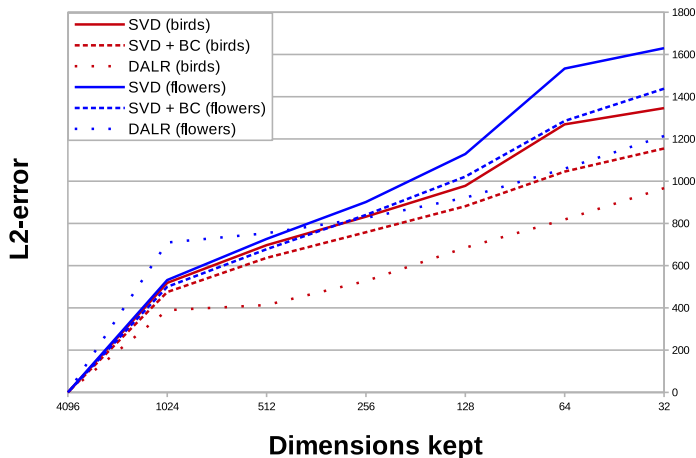
- By writing $C = AB^T$ and $Z = WX$, and adding a ridge penalty to guarantee good behavior, we have a **rank-constrained regression problem** which can be written as:

$$\begin{aligned} \arg \min_C & \|Z - CX\|_F^2 + \lambda \|C\|_F^2 \\ \text{s.t.} \quad & \text{rank}(C) \leq k, \end{aligned}$$

- [Mukherjee 2013] showed there is a closed-form solution to this problem, and we obtain new weights A and B by SVD on C .

Experiments: error analysis

- In fact, we immediately see that Frobenius norm preservation is not equivalent (this was already clear, with a little thought).
- It becomes **acutely** apparent at high compression rates.



- Compressing only fc6 (the **big** layer) on CUB-200 (a dataset of 200 bird species, **original accuracy 55.73%**):

dim kept	32	64	128	256	512	1024
params	0.91%	1.82%	3.64%	7.27%	14.54%	29.08%
SVD	16.83	36.47	51.74	54.47	54.85	55.25
SVD + BC	27.91	46.00	53.50	54.83	55.21	55.44
Pruning (mean)	4.30	8.06	12.57	25.82	37.25	50.41
Pruning (max)	4.06	7.01	15.26	25.27	36.69	48.95
DALR	48.81	54.51	55.78	55.85	55.71	55.82

- Combined** compression of fc6 and fc7:

	Orig. Acc	Compr. Acc	fc6 red.	fc7 red.	total red.
Birds	55.73	55.97	4.66%	19.97%	6.81%
Flowers	78.84	77.62	10.45%	41.99%	14.88%
Stanford	68.73	69.43	25.59%	55.96%	29.85%

The good

- For typical settings where one would accept a **loss of accuracy** of around 1%, truncated SVD must retain between 4x and 8x the parameters compared to DALR *to maintain the same level of performance*.
- Does not require **radical architectural modifications** to the network.

The bad and the ugly

- Compression is **layerwise greedy**: correlations between layers are not exploited.

Final comments

- This work that is primarily an application of classical linear **optimization**.
- Given the vast literature on **low-rank matrix factorization**, I doubt DALR is the end-all-be-all of objective formulations.

Discussion

- In this talk I have given a high-entropy overview of two techniques that can be used to get **more** for **less** out of your CNNs.
- Self-supervised Learning by Learning to Rank:
 - F0o
 - Bar
- Domain Adaptive Low-rank Decomposition (DALR):
 - Foo
 - Bar

Current directions

- Some hot topics on my horizon (or already on the field):
 - **Lifelong learning**: CNNs don't learn like humans – when they learn **new** tasks, they easily **forget** old ones. Can we realize **sequential**, **lifelong** task learning? Maybe second-order techniques are a way forward.
 - **Zero-shot learning**: CNNs are notorious for not generalizing knowledge. Can we learn representations that can be applied to **new problems** without requiring **any** labeled examples in the new domain?

Thanks

- The work presented today obviously represents the effort of **many collaborators**, from colleagues to students to postdocs.
- **Florence**: Riccardo Del Chiaro, My Kieu.
- **Barcelona**: Joost van de Weijer, Xia-lei Liu, Marc Masana, Luis Herranz, Esteve Cervantes, Jose Alvarez.

Self-supervised learning:

- X. Liu, J. van de Weijer, A. D. Bagdanov, "Exploiting Unlabeled Data in CNNs by Self-supervised Learning to Rank." *Under minor revision: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.
- X. Liu, J. van de Weijer, A. D. Bagdanov, "Self-Supervised Crowd Counting in Images by Learning to Rank." *In: Proceedings of CVPR, 2018.*
- X. Liu, J. van de Weijer, A. D. Bagdanov, "Rank-IQA: Learning from Rankings for No-reference Image Quality Assessment." *In: Proceedings of ICCV, 2017.*

Network compression:

- M. Masana, J. van de Weijer, L. Herranz, A. D. Bagdanov, J. M. Alvarez, "Domain-adaptive deep network compression." *In: Proceedings of ICCV, 2017.*
- M. Masana, J. van de Weijer, A. D. Bagdanov, "On-the-fly Network Pruning for Object Detection." *In: Proceedings of ICLR, 2016.*

Zero-shot and lifelong learning:

- R. Del Chiaro, A. D. Bagdanov, A. Del Bimbo, "NoisyArt: a Dataset for Webly-supervised Artwork Recognition." *In: Proceedings of VISAPP, 2019 (to appear).*
- X. Liu, M. Masana, L. Herranz, J. van de Weijer, A. Lopez, A. D. Bagdanov, and J. M. Alvarez, "Rotate your Networks: Better Weight Consolidation and Less Catastrophic Forgetting." *In: Proceedings of ICPR, 2018.*