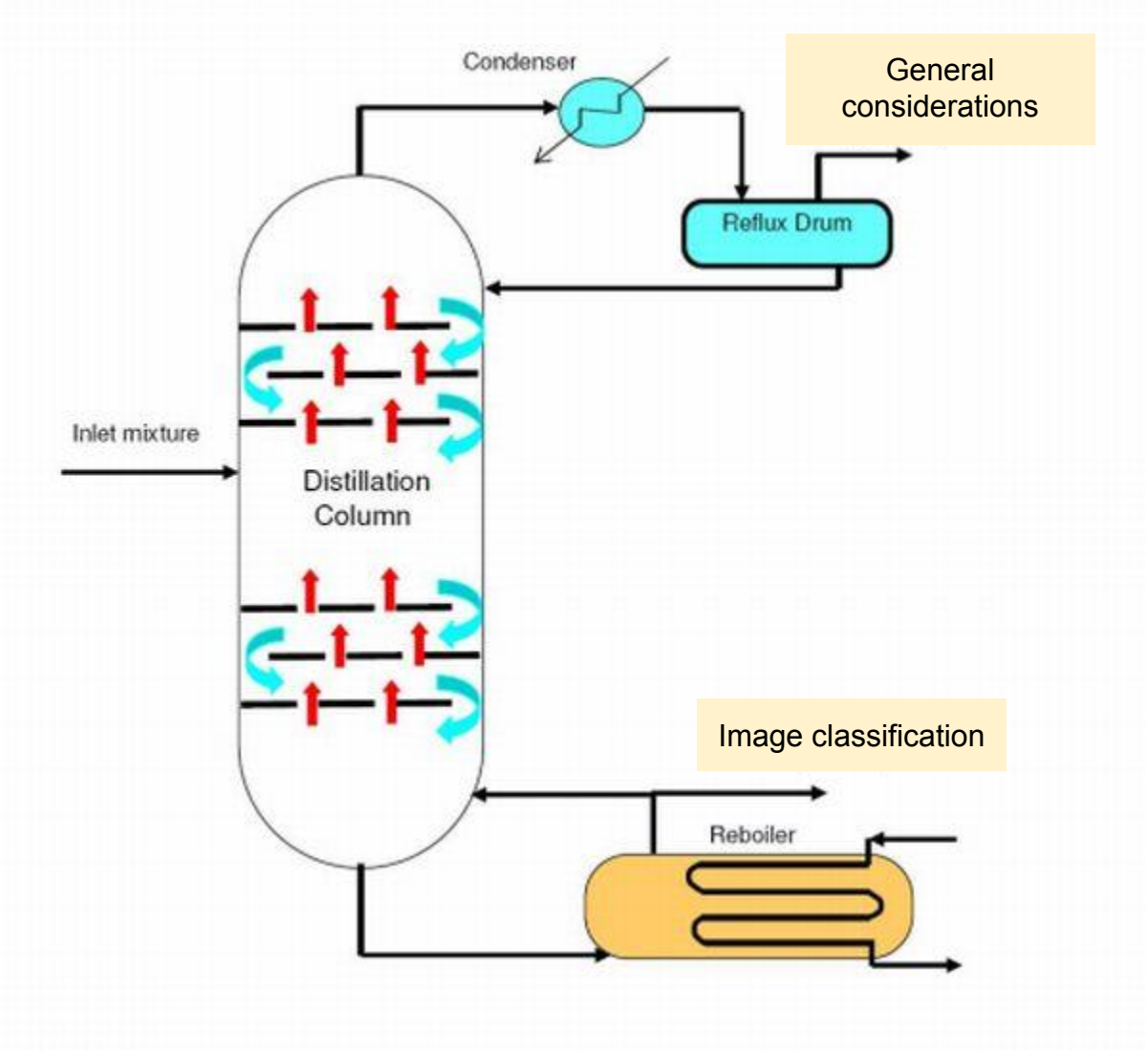
COLA 2016 - Ferrara, February 26, 2016

Squeezing Deep Learning onto a Phone

Carlo Fantozzi
Fabio Maiani
University of Padova

Byproduct


Condenser
General considerations
Reflux Drum
Inlet mixture
Distillation Column
Image classification
Reboiler

Google Research Blog

The latest news from Research at Google

How Google Translate squeezes deep learning onto a phone

Wednesday, July 29, 2015

Posted by Otavio Good, Software Engineer, Google Translate

Today we announced that the Google Translate app now does real-time visual translation of 20 more languages. So the next time you're in Prague and can't read a menu, we've got your back. But how are we able to recognize these new languages?

In short: deep neural nets. When the Word Lens team joined Google, we were excited for the opportunity to work with some of the leading researchers in deep learning. Neural nets have gotten a lot of attention in the last few years because they've set all kinds of records in image recognition. Five years ago, if you gave a computer an image of a cat or a dog, it had trouble telling which was which. Thanks to convolutional neural networks, not only can computers tell the difference between cats and dogs, they can even recognize different breeds of dogs. Yes, they're good for more than just trippy art—if you're translating a foreign menu or sign with the latest version of Google's Translate app, you're now using a deep neural net. And the amazing part is it can all work on your phone, without an Internet connection. Here's how.

Deep Neural Networks (DNNs)

Resurgence of neural networks in recent years

Breakthrough results in computer vision, language processing, games, ...

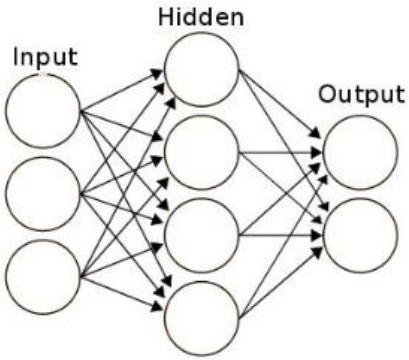
New: **several hidden layers (hence the word “deep”) with novel elements**

Advances in training algorithms to fight overfitting

Use of resources (CPU, memory, energy): training vs. inference

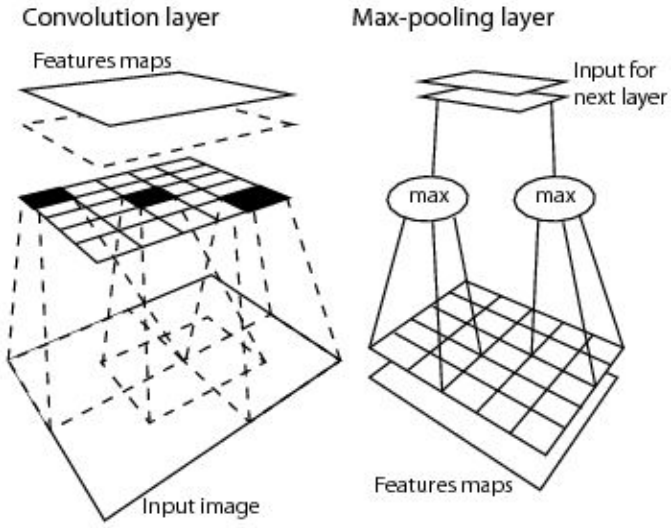
Several software tools: Caffe, DistBelief, TensorFlow, ...

DNNs: Layers



- Fully connected [memory consuming]
- Convolutional [computationally expensive]

  Variant: convolutional + MLP (used in NIN)
- Pooling (max, average)
- Response normalization
- Softmax



Standard design in image classification: convolutional layers, then FC layers

Trend: increase the number of layers, hence complexity and resource consumption

DNNs on Mobile Devices

Software approach (Stanford): quantize the network, compress data (http://arxiv.org/abs/1510.00149, 2016)

Hardware approach (MIT+NVIDIA): design ad-hoc hardware for DNNs (http://dspace.mit.edu/handle/1721.1/101151, ISSCC 2016)

Under which circumstances are such approaches necessary?
Is it a **"poor man's approach" already viable in some cases?**

Inference:

- What is the actual resource utilization (CPU, memory)?
- How much slower w.r.t. a PC?
- What is the actual energy consumption?

Commodity hardware

ImageNet

From http://dx.doi.org/10.1109/CVPR.2015.7298594

Figure 1: Two distinct classes from the 1000 classes of the ILSVRC 2014 classification challenge. Domain knowledge is required to distinguish between these classes.

Large database of tagged images

14,197,122 images in 21,841 classes

Reference dataset in the image classification community

ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

AlexNet

5 convolutional layers, 3 FC layers, plus maxpools & normalization layers
60 million parameters

“removing any convolutional layer [...] resulted in inferior performance”

“In the end, the network’s size is limited mainly by the amount of memory available on current GPUs and by the amount of training time that we are willing to tolerate”

Winner of ILSVRC-2012

Network In Network (NIN)

3 convolutional + MLP layers, 1 global average pooling layer

MLP as a “potent” nonlinear function approximator

GoogLeNet

“Logical culmination of NIN”

Designed by keeping mobile/embedded computing in mind: computational budget of 1.5 billion multiply-adds at inference time

12x fewer parameters than AlexNet while being more accurate

27 layers including pooling layers

Concat layer

Training: “extra” network for faster back-propagation in the main, deep network

Winner of ILSVRC-2014 (relying on ensemble prediction)

Caffe

Deep learning framework designed by Berkeley VLC; open source

State-of the-art NN building blocks
Models and optimization defined by configuration without hard-coding

Expressive architecture encourages application and innovation

Fast; optimized for GPUs

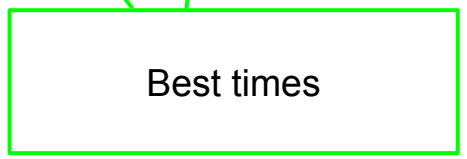
Active community

Several prominent DNNs (incl. AlexNet, NIN, GoogLeNet) available as Caffe models

Memory and CPU: PC

Model size (MB)	227	26	41

Operation	AlexNet	NIN	GoogLeNet
Initialization time (s)	1.771	0.239	0.158
Inference time (s)	0.172	0.132	0.585



Main PC characteristics

CPU: quad-core Intel Core i5-2550K, 3.8 GHz

RAM: 16 GB (4x4) DDR3 1333 MHz

Memory and CPU: Mobile

Model size (MB)	227	26	41
Operation	**AlexNet**	**NIN**	**GoogLeNet**
Initialization time (s)	16.738	2.067	1.986
Inference time (s)	1.091	1.079	2.198

The size of AlexNet's model heavily affect tinitialization

Best times

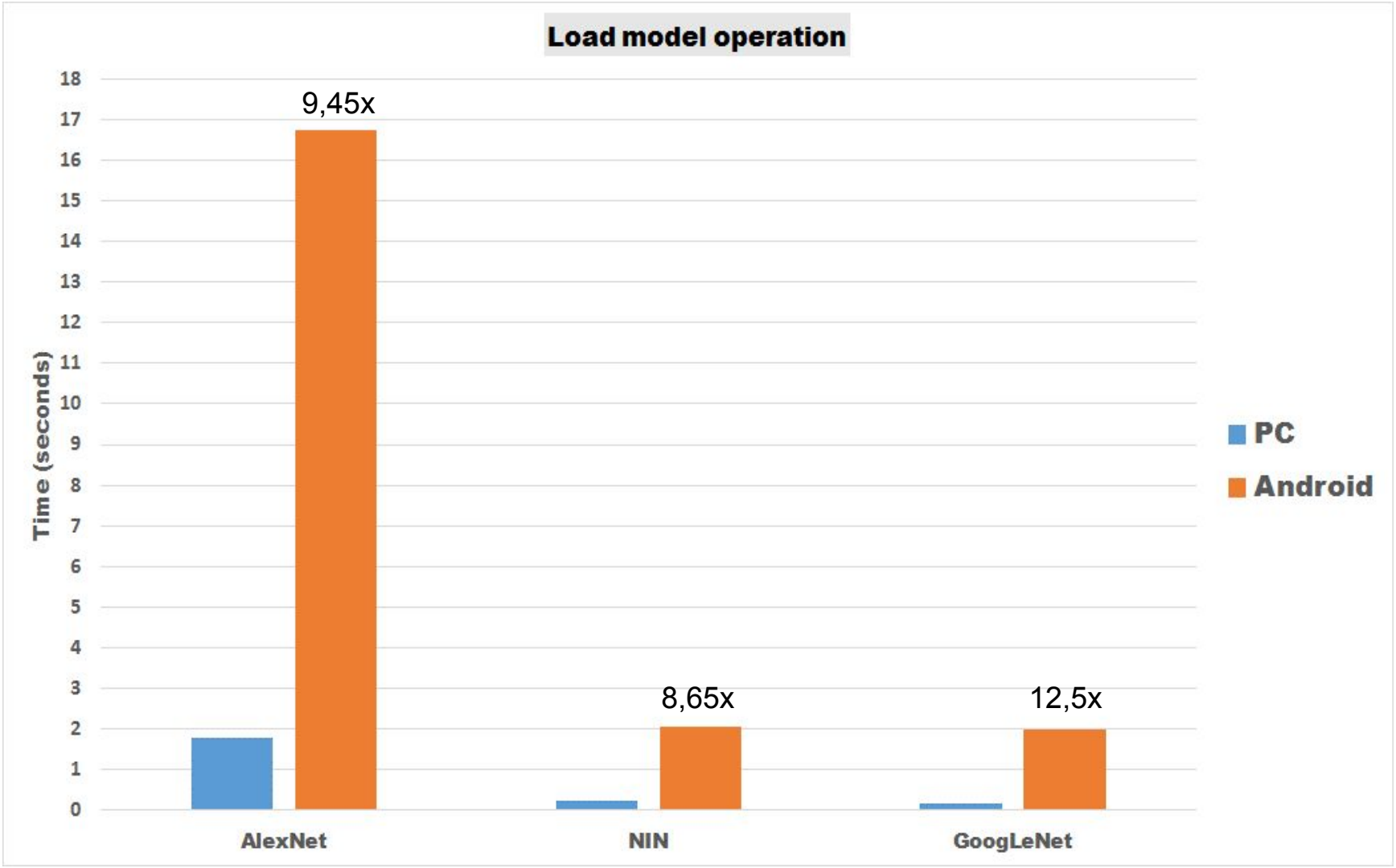
Mobile device (Nexus 7 2013)

CPU: 1.51 GHz quad-core Krait 300

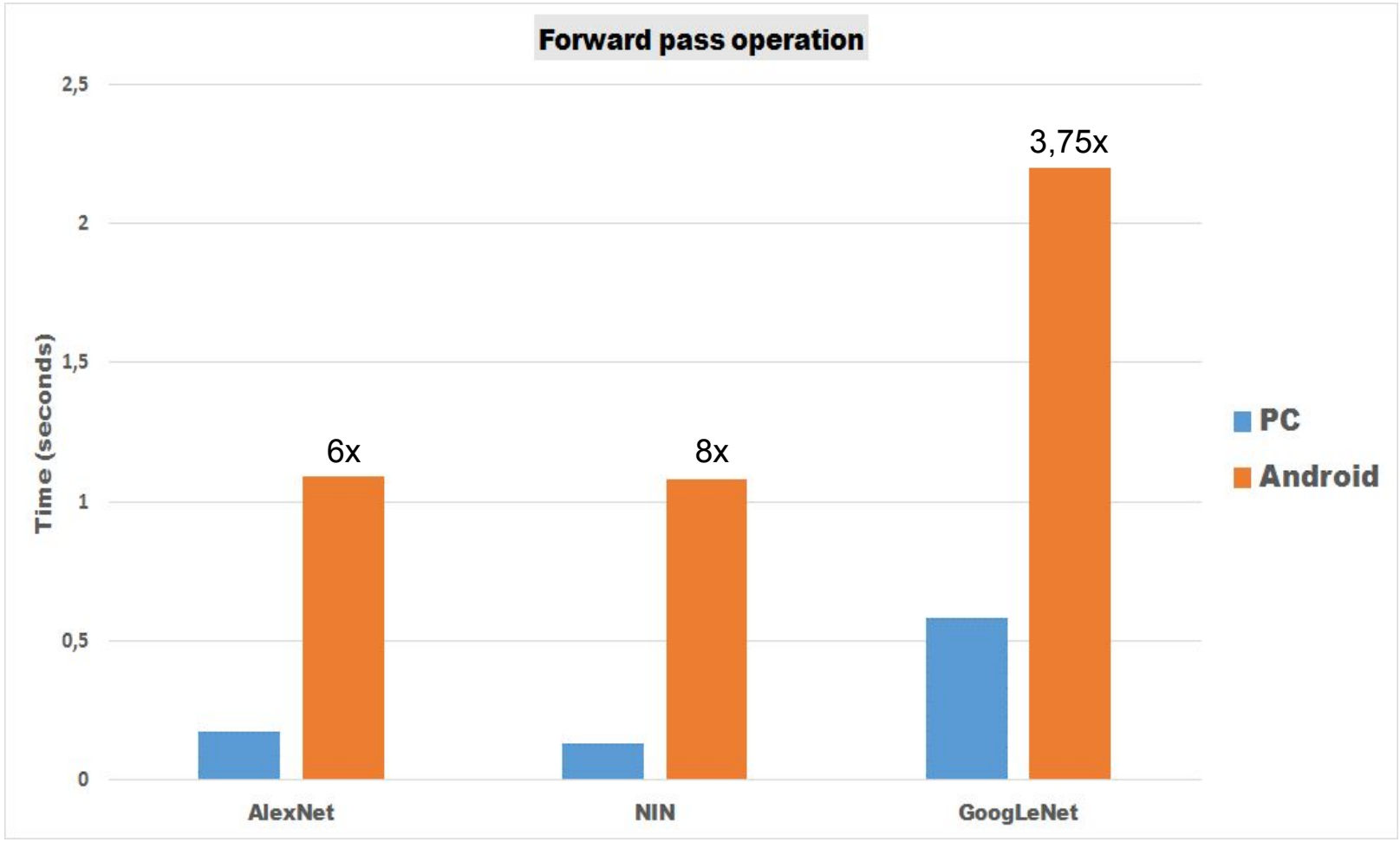
RAM: 2 GB DDR3L 1600 MHz

Battery: 3,950 mAh (i.e., 15 Wh, 54 KJ)

CPU: PC vs. Mobile, Initialization



CPU: PC vs Mobile, Inference



Comments on Memory and CPU

"Order of times" is the same on PC and mobile:

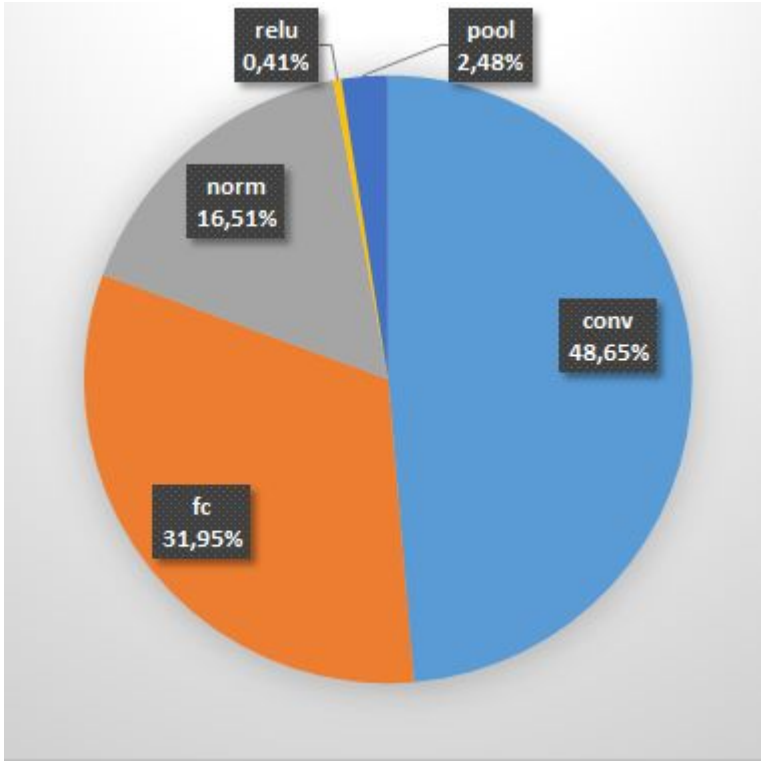
- initialization time: GoogLeNet < NIN << AlexNet
- inference time: NIN < AlexNet < GoogLeNet

All in all, NIN is the fastest network; not the most accurate, however
(Remember that you may need to load the model several times on mobile)

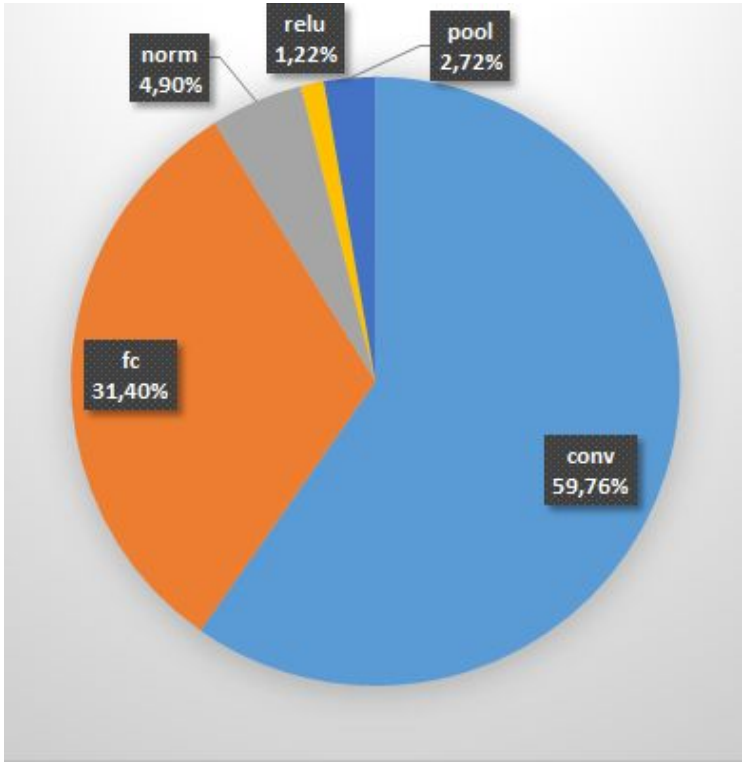
NIN is also the network that requires less memory

Breakdown of inference time on PC and mobile?

Inference Time: AlexNet



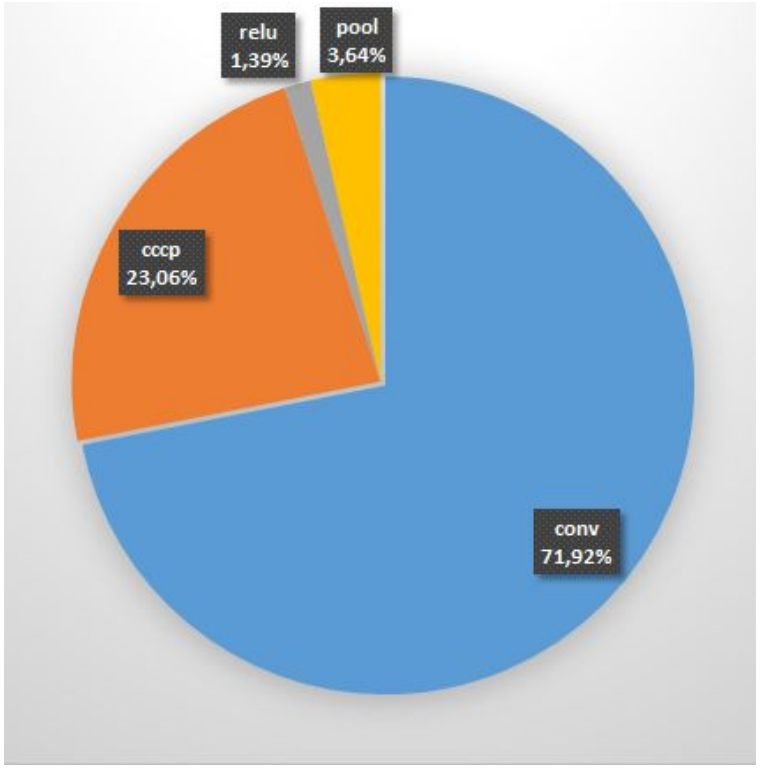
PC



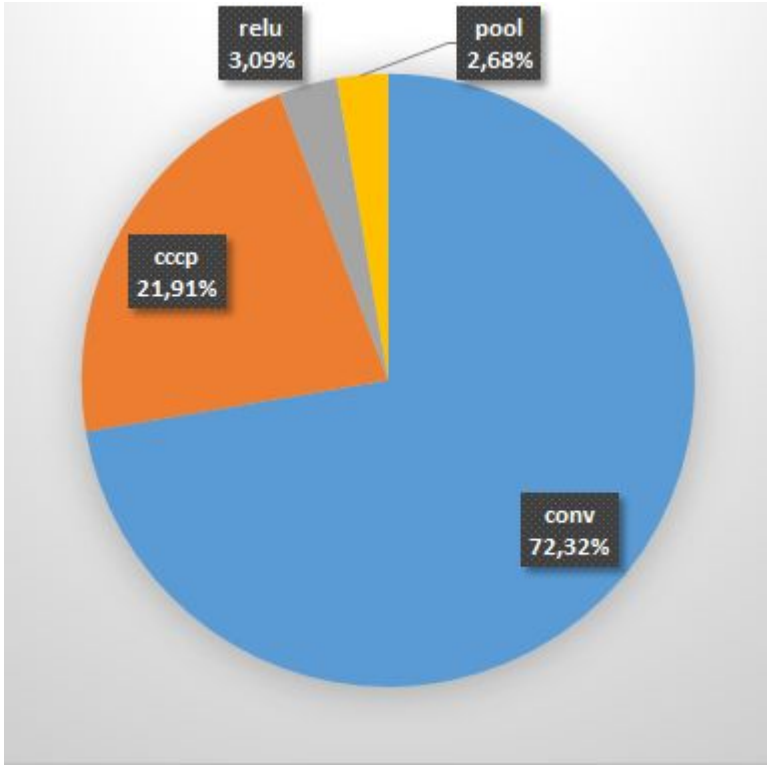
Android Device

- Each figure is the sum of all layer's time percentage of one specific type of layer, observed during a single forward pass
- Note the difference between the normalization layers percentage

Inference Time: NIN



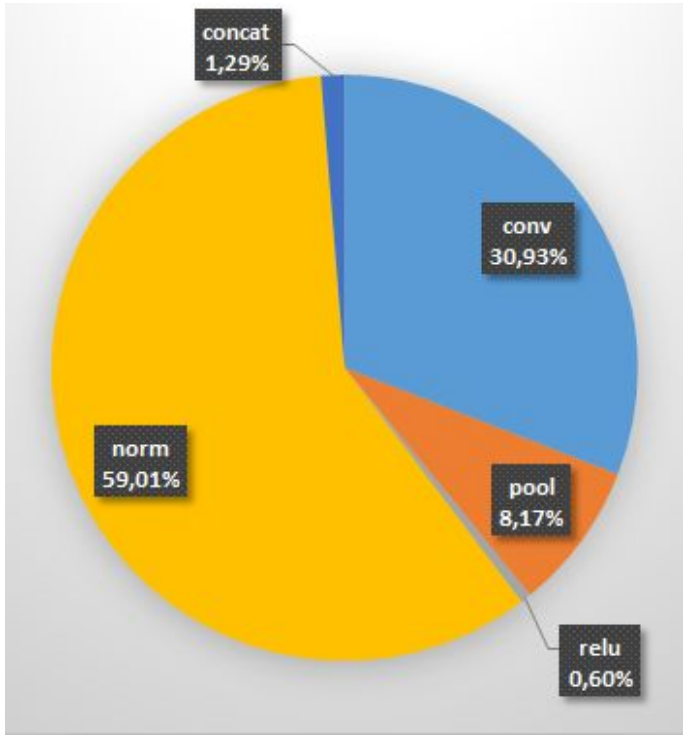
PC



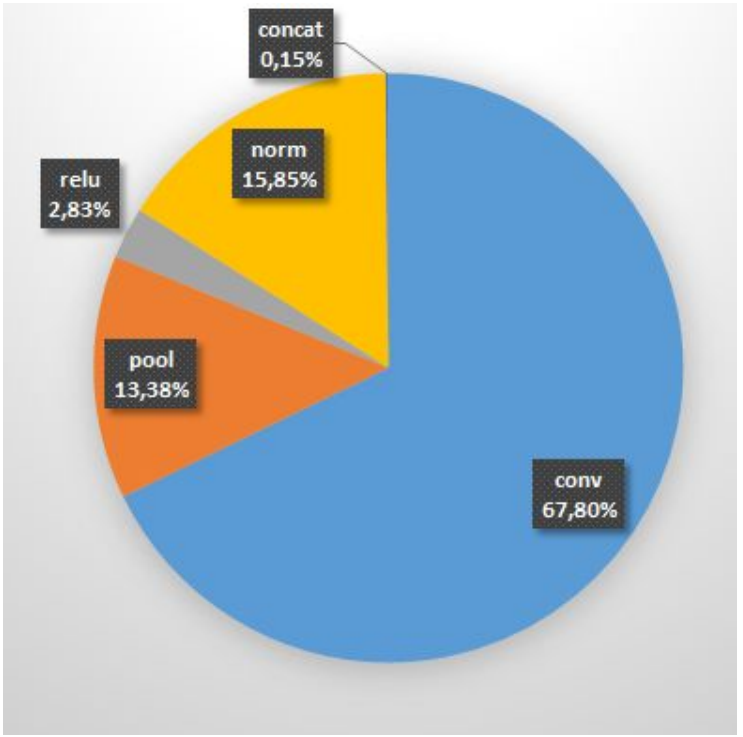
Android Device

- No normalization here, so we have very similar data

Inference Time: GoogLeNet



PC



Android Device

- Still some difference because of normalization
- Very high contribution of normalization on PC

Comments on Inference Time

General Remarks

The most expensive layers are the convolutional ones (up to ~70%), as expected

Not the “over 90%” figures sometimes found in the open literature

AlexNet

The single most expensive layer is fully connected

Less time spent for normalization on mobile, but similar behavior

NIN

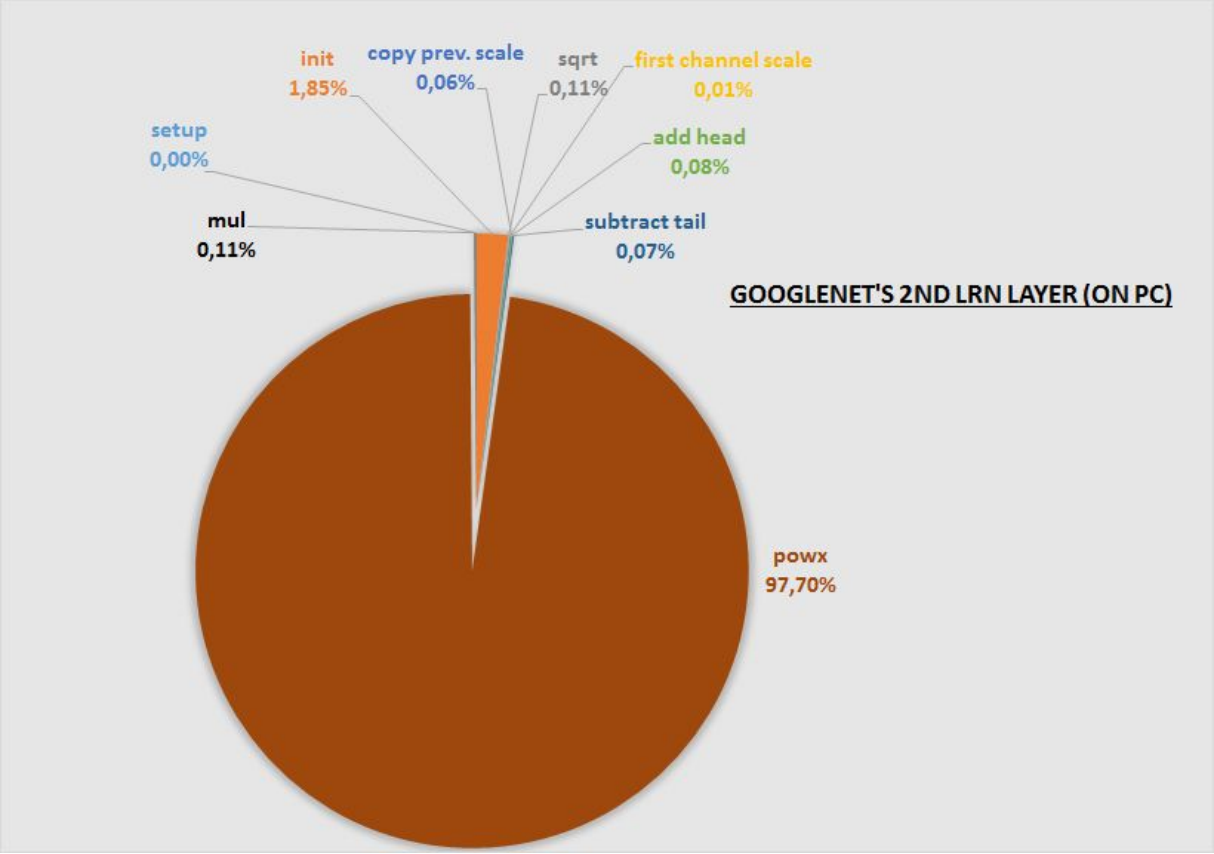
`cccp` (cascadable cross-channel parametric) layers very significant: 22÷23%

GoogLeNet

Suspicious difference on `norm` (~59% on PC, ~16% on mobile)

Local Response Normalization (LRN)

- Normalizes over local input regions
- Can sometimes improve generalization and consequently the overall results
- Not strictly necessary (ReLU layer do not require input normalization)



Energy: Mobile

Non-intrusive measurements via the USB port, in a real-world setting

Sources of interference minimized, of course (airplane mode, no apps running, …)

	AlexNet	NIN	GoogLeNet
Energy for 1 inference (J)	4.05	3.65	5.63

Comparison stone: the energy for uploading a 1.1MB picture via mobile was estimated to be 40 J with 4G, 23 J with Wi-Fi (http://dx.doi.org/10.1109/CCGrid.2014.68)

Comments on Energy

GoogLeNet -- which has the highest inference time -- also sports the highest energy consumption. AlexNet and NIN have similar consumption

Energy related to inference time, as expected

The fact that NIN has no FC layers, with a huge amount of parameters, is probably the cause of why NIN consume less energy than AlexNet, even if NIN is a little deeper

With a fully-charged Nexus 7 battery, we can estimate to perform

- ~13,300 inferences with AlexNet
- ~14,800 inferences with NIN
- ~ 9,600 inferences with GoogLeNet

Conclusions

"Poor man's approach" viable in some scenarios

- Main memory, Storage
- CPU (compute time), Energy

Compute time gap between mobile and PC: limited, even with obsolescent hardware

Client-side computing: will replace server-side computing? I think so

Competitive even from an energy standpoint? Further investigation needed

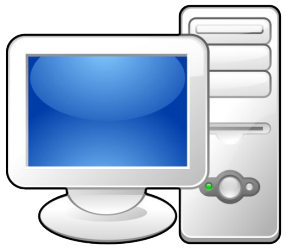
Investigate relations between the energy consumed by a layer
and the contribution of the layer to classification performance

COLA 2016

Thank You

Ferrara, February 26, 2016

DNNs on Android



when you have a good model

1. Create your own dataset
2. Choose a framework for CNN (we select Caffe)
3. Select a specific network and train it on your dataset
4. Keep only good final models (high accuracy on validation set)

1. Import the needed libraries
2. Import the network's files necessary for predictions (model included)
3. Test your network on Android (take a photo or import an image)

Energy: Mobile Setup

To measure the energy spent with a single inference of each network we used a normal USB charge cable, with a power supply

- add a resistor R in series across the supply voltage (give from the power supply) to measure the electric potential difference on it
- R must not be too high (at most 1÷2% of the circuit overall power must be dissipated on it)
- our case: R=0.12 Ohm, V=5.16 Volt
- we used an oscilloscope Agilent DSO3102A to measure the electric potential difference

About the Android device:

- Airplane mode
- Display is ON
- All other apps are closed during the experiment