E2GC: Energy-efficient Group Convolution in Deep Neural Networks

Nandan Kumar Jha, Rajat Saini, Subhrajit Nag, Sparsh Mittal

{cs17mtech11010, cs17mtech11002, cs17resch11006, sparsh}@iith.ac.in

Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad

The 33rd International Conference on VLSI Design &
The 19th International Conference on Embedded Design

January 6, 2020

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Outline

- Introduction: challenges and motivation
- Previous works: finding the gap
- Proposed approach: E2GC
- Experimental results: E2GC vs. FgGC
- Predictive performance: A discussion
- Conclusion

Challenges: DNNs and Energy Consumption (training)

Higher predictive performance of DNNs comes at the cost of higher computational complexity and large model size.

Challenges: DNNs and Energy Consumption (training)

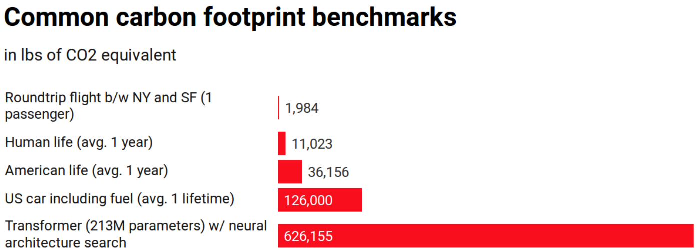
Higher predictive performance of DNNs comes at the cost of higher computational complexity and large model size.

- Deeper and wider DNNs: (# FLOPs ($\sim 10^9$), # Parameters ($\sim 10^6$))
  - ResNet-101 (7.6B, 44.5M), ResNeXt-50 (4.2B, 25M)

Challenges: DNNs and Energy Consumption (training)

Higher predictive performance of DNNs comes at the cost of higher computational complexity and large model size.

- Deeper and wider DNNs: (# FLOPs ($\sim 10^9$), # Parameters ($\sim 10^6$))
  - ResNet-101 (7.6B, 44.5M), ResNeXt-50 (4.2B, 25M)



Source: MIT Technology Review (June 6, 2019)

Challenge: High power consumption.
"Training a single model can emit as much carbon as **five** cars in their lifetimes"

Challenges: DNNs and Energy Consumption (inference)

	Deployed IOT	Wearables	Mobile Phones	Surveillance
Example Application Processor	Ti MSP430	Snapdragon Wear, Apple S3, Exynos 7 dual	Snapdragon 845, Apple A11, Exynos 9	Qualcomm QC605
Power Budget	50-1000 µW	1-2 W	3-5W	4-7W
Typical Battery	Energy Harvest / Li-Ion 5-300 mWh	Li-Ion 0.9 - 1.7 Wh	Li/Polymer-Ion 10-20 Wh	Li-Ion / Alkaline 5-40 Wh

Source: Kurt Keutzer (DeepScale AI)

Challenge: Limited battery life and tight power budget.

Challenges: DNNs and Energy Consumption (inference)

	Deployed IOT	Wearables	Mobile Phones	Surveillance
Example Application Processor	Ti MSP430	Snapdragon Wear, Apple S3, Exynos 7 dual	Snapdragon 845, Apple A11, Exynos 9	Qualcomm QC605
Power Budget	50-1000 µW	1-2 W	3-5W	4-7W
Typical Battery	Energy Harvest / Li-Ion 5-300 mWh	Li-Ion 0.9 - 1.7 Wh	Li/Polymer-Ion 10-20 Wh	Li-Ion / Alkaline 5-40 Wh

Challenge: Limited battery life and tight power budget.

Source: Kurt Keutzer (DeepScale AI)

"AI Algorithm will be measured by the amount of intelligence they provide per killowatthour. Broad economic viability requires energy-efficiency"

- Max Welling, "Intelligence per Killowatthour ", ICML invited talk, 2018

Solution?

Solution?

Compact DNNs

Compact DNN Techniques

- Pruning
- Knowledge Distillation
- Tensor Factorization
- Low rank filters (1×3 and 3×1 filters)
- Group Convolution
- Depthwise Convolution

Compact DNN Techniques

- Pruning
- Knowledge Distillation
- Tensor Factorization
- Low rank filters (1×3 and 3×1 filters)
- Group Convolution
- Depthwise Convolution

$\boxed{\textbf{Group convolution}}$

Background: Types of convolution

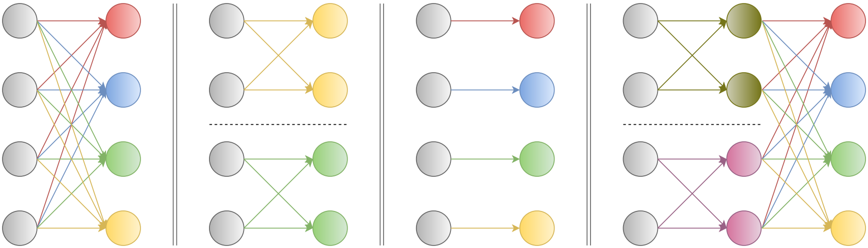


Figure: (a) Standard conv, (b) Group conv (GCconv), (c) Depthwise conv (DWConv), (d) GConv followed by pointwise conv.

Quantity (symbol)	Expression	Quantity (symbol)
# MACs (M_c)	$n \times m \times d_k^2 \times h \times w$	# input fmaps (m)
# parameters (P)	$n \times m \times d_k^2$	# output fmaps (n)
# activations (A)	$(n + m) \times h \times w$	# groups in GConv (g)
Arith. intensity (AI)	$M_c/(P + A)$	Energy per frame (EPF)
# channels in a group (G)	m/g	Group size in GConv (G)

Why group convolution?

- Enables compute and parameter **efficient** convolution
  - Only a group of ifmaps ($\frac{m}{g}$) interacts with each ofmap.
  - Parameters and FLOPs (in each group) reduces by a **factor** of g^2.

M_c	P	A (ifmaps + ofmaps)
$\frac{n \times m \times h \times w \times d_k^2}{g}$	$\frac{n \times m \times d_k^2}{g}$	$(n + m) \times h \times w$

Why group convolution?

- Enables compute and parameter **efficient** convolution
  - Only a group of ifmaps ($\frac{m}{g}$) interacts with each ofmap.
  - Parameters and FLOPs (in each group) reduces by a **factor** of g^2.

M_c	P	A (ifmaps + ofmaps)
$\frac{n \times m \times h \times w \times d_k^2}{g}$	$\frac{n \times m \times d_k^2}{g}$	$(n+m) \times h \times w$

- **Sparsity** and **Redundancy:**
  - GConv breaks the fully connected pattern between ifmaps and ofmaps and reduces the redundancy in channel extent.
  - GConv acts as coarse-grained and structured channel pruning.

Why group convolution?

- Enables compute and parameter **efficient** convolution
  - Only a group of ifmaps ($\frac{m}{g}$) interacts with each ofmap.
  - Parameters and FLOPs (in each group) reduces by a **factor** of g^2.

M_c	P	A (ifmaps + ofmaps)
$\frac{n \times m \times h \times w \times d_k^2}{g}$	$\frac{n \times m \times d_k^2}{g}$	$(n+m) \times h \times w$

- **Sparsity** and **Redundancy:**
  - GConv breaks the fully connected pattern between ifmaps and ofmaps and reduces the redundancy in channel extent.
  - GConv acts as coarse-grained and structured channel pruning.

Thus, GConv achieves better compute efficiency along with structured sparsity, which enables better generalization and better hardware acceleration.

Previous Work on GConv

Interleaved (comprised of primary and secondary) GConv

IGCNet-V1 [ICCV'17], IGCNet-V2 [CVPR'18], IGCNet-V3 [BMVC'18].

GConv with fixed g (FgGC module)

AlexNet (g=2) [NIPS'12], ResNeXt (g=32) [CVPR'17], ShuffleNet-V1 (g=3) [CVPR'18].

Depthwise convolution: A special case of GConv

MobileNet-V1/V2 [CVPR'18], ShuffleNet-V1/V2 [ECCV'18], Xception-Net [CVPR'17], ChannelNet [NeurIPS'18].

- CondenseNet [CVPR'18]: learned GConv.

Key Shortcoming

Overlooked the interplay of g with the number of computation and data reuse which results in energy-inefficient design of DNN.

Key Shortcoming

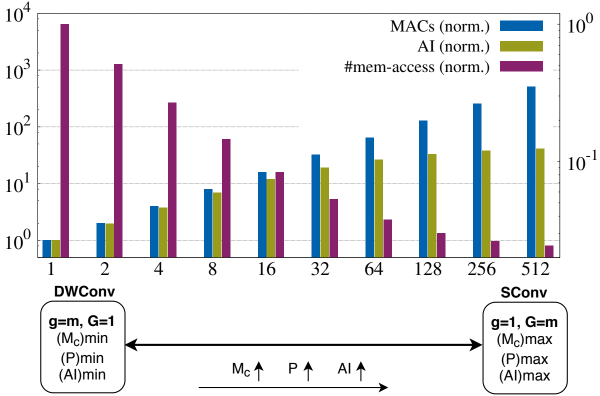
Overlooked the interplay of g with the number of computation and data reuse which results in energy-inefficient design of DNN.

M_c	AI
$\frac{n \times m \times h \times w \times d_k^2}{\mathbf{g}}$	$\frac{n \times m \times h \times w \times d_k^2}{n \times m \times d_k^2 + \mathbf{g} \times (n+m) \times h \times w}$

Key Shortcoming

Overlooked the interplay of g with the number of computation and data reuse which results in energy-inefficient design of DNN.

M_c	AI
$\dfrac{n\times m\times h\times w\times d_k^2}{\mathbf{g}}$	$\dfrac{n\times m\times h\times w\times d_k^2}{n\times m\times d_k^2+\mathbf{g}\times(n+m)\times h\times w}$



Optimal Energy Efficiency

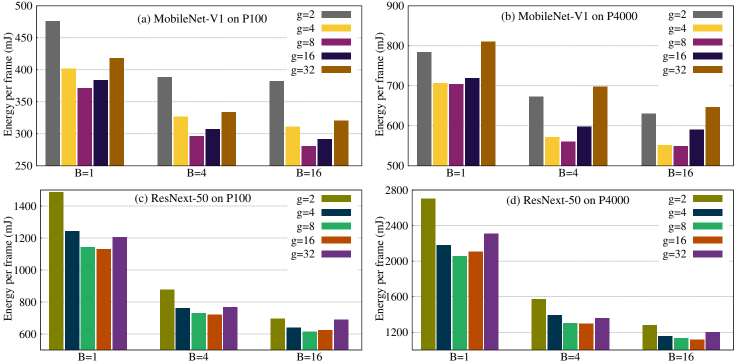


Figure: Energy per frame for MobileNet-V1 (a, b) and ResNeXt-50 (c, d).

Optimal Energy Efficiency

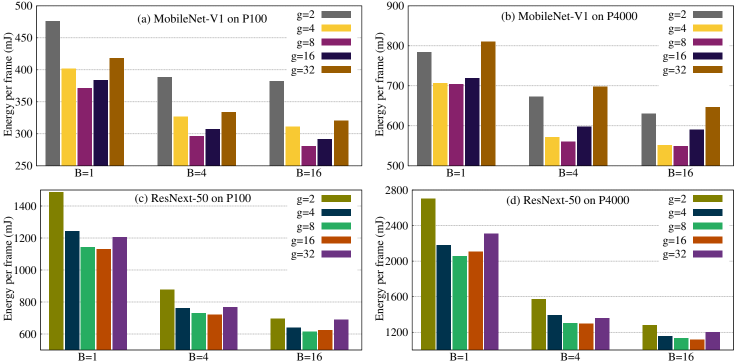


Figure: Energy per frame for MobileNet-V1 (a, b) and ResNeXt-50 (c, d).

Observation: At lower g computational cost is consequential while at higher g effect of lower data reuse outweigh the benefit of reduced computations.

Balancing computations and data reuse

$$M_c \times \left(\frac{\alpha}{AI}\right)^{\beta} = const; \text{ where } AI = \frac{M_c}{A+P}$$
$$\implies (M_c)^{1-\beta} \times (A+P)^{\beta} = const \times \alpha^{-\beta}$$

α and β are two platform-dependent variables such that $\alpha, \beta \in (0,1]$.

α accounts for the memory-hierarchy and β for the disparity in energy consumption.

$$(M_c)^{1-\beta} \times P^{\beta} = \gamma \ \ (\gamma = const \times \alpha^{-\beta})$$

$$\implies \left(\frac{m \times n \times h \times w \times d_k^2}{g}\right)^{1-\beta} \times \left(\frac{m \times n \times d_k^2}{g}\right)^{\beta} = \gamma$$

$$\implies g = \frac{m \times n \times d_k^2 \times (h \times w)^{1-\beta}}{\gamma} \implies g = f(m, n, d_k, h, w) \qquad (1)$$

Balancing computations and data reuse

$$M_c \times \left(\frac{\alpha}{AI}\right)^{\beta} = const; \text{ where } AI = \frac{M_c}{A+P}$$
$$\implies (M_c)^{1-\beta} \times (A+P)^{\beta} = const \times \alpha^{-\beta}$$

α and β are two platform-dependent variables such that $\alpha, \beta \in (0, 1]$.

α accounts for the memory-hierarchy and β for the disparity in energy consumption.

$$(M_c)^{1-\beta} \times P^{\beta} = \gamma \ \ (\gamma = const \times \alpha^{-\beta})$$

$$\implies \left(\frac{m \times n \times h \times w \times d_k^2}{g}\right)^{1-\beta} \times \left(\frac{m \times n \times d_k^2}{g}\right)^{\beta} = \gamma$$

$$\implies g = \frac{m \times n \times d_k^2 \times (h \times w)^{1-\beta}}{\gamma} \implies g = f(m, n, d_k, h, w) \qquad (1)$$

Takeaway: Keeping "**number of groups** (g)" constant (FgGC module) in DNNs is *incongruous* with energy-efficiency.

E2GC module: Balancing computations and data reuse

$$g \propto m \times n \times (h \times w)^{(1-\beta)}$$
$$\text{Let } h = w = d_f \text{ and } \beta = 0.5 \text{ (Since, } \beta \in (0,1] \text{)}$$
$$\implies g \propto m \times n \times d_f \tag{2}$$

To avoid representational bottleneck in DNNs, $\frac{d_k}{2}$ and $2n$ occurs in one block.

$$\text{Hence,} \quad g \propto m \implies G = \eta \ \left(\text{since, } G = \frac{m}{g}\right) \tag{3}$$

E2GC module: Balancing computations and data reuse

$$g \propto m \times n \times (h \times w)^{(1-\beta)}$$

$$\text{Let } h = w = d_f \text{ and } \beta = 0.5 \text{ (Since, } \beta \in (0,1] \text{)}$$

$$\implies g \propto m \times n \times d_f \qquad (2)$$

To avoid representational bottleneck in DNNs, $\frac{d_k}{2}$ and $2n$ occurs in one block.

$$\text{Hence, } \quad g \propto m \implies G = \eta \ \left(\text{since, } G = \frac{m}{g}\ \right) \qquad (3)$$

Takeaway: Keeping "**number of channels (G)**" constant in the groups of GConv of DNNs is **congruous** with energy-efficiency.

E2GC vs. FgGC

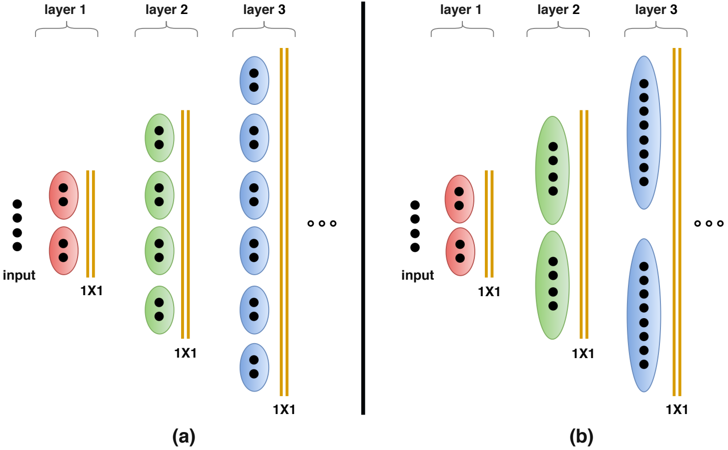


Figure: (a) Proposed E2GC module where G remain constant in all the conv layers (b) Conventional FgGC module where g remains same in all the conv layers.

Experimental Setup

Table: GPU specifications used in our experiments

GPU	# core	L2 size	Peak bandwidth	Peak Throughput	CMR
P100	3584	4 MB	549 GB/s	9.3 TFLOPS	16.94 FLOPs/Byte
P4000	1792	2 MB	243 GB/s	5.2 TFLOPS	21.4 FLOPs/Byte

$$\text{Energy per frame (EPF)} = \frac{(\text{average power}) \times (FP_t + BP_t)}{(\text{batch size})}$$

- FP_t and BP_t are average of 100 iterations.
- Power measurement: `nvidia-smi`
- Deep learning framework: `Caffe` and `PyTorch`

Results: MobileNet-V1 with E2GC module

Table: Energy per frame (*EPF*) comparison of MobileNet-V1 with E2GC/FgGC module.

MobileNet-V1	Params ($\times10^6$)	MACs ($\times10^6$)	EPF on P100 (millijoule)			EPF on P4000 (millijoule)		
			B=1	B=4	B=16	B=1	B=4	B=16
E2GC (G=1)	4.20	568.74	689	630	613	1607	1448	1373
E2GC (G=2)	4.25	586.13	538	482	450	1169	1014	983
E2GC (G=4)	4.34	620.90	421	357	330	1005	846	816
E2GC (G=8)	4.52	690.44	**373**	316	302	780	630	592
E2GC (G=16)	4.87	829.53	370	284	277	689	551	507
E2GC (G=32)	5.59	1107.71	363	281	265	648	501	480
FgGC (g=2)	16.72	2690.06	476	388	383	785	673	631
FgGC (g=4)	10.44	1620.71	402	327	312	706	572	551
FgGC (g=8)	7.30	1086.03	372	297	281	704	561	550
FgGC (g=16)	5.73	818.69	384	308	292	719	598	591
FgGC (g=32)	4.95	685.02	**418**	334	321	811	698	647

MobileNet-V1 with E2GC module is **10.8%** and **8.7%** more energy efficient and parameter efficient (respectively) than **MobileNet-V1 with F2GC** module (#MACs ≈ 690M) ☺

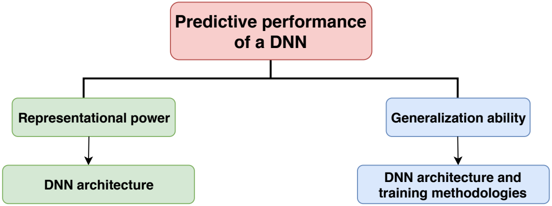
Results: ResNeXt-50 with E2GC module

Table: Energy per frame (*EPF*) comparison of ResNeXt-50 with E2GC/FgGC module.

ResNeXt-50	Params ($\times 10^6$)	MACs ($\times 10^9$)	EPF on P100 (millijoule)			EPF on P4000 (millijoule)		
			B=1	B=4	B=16	B=1	B=4	B=16
E2GC (G=1)	23.61	4.02	2185	1248	712	5780	2371	1434
E2GC (G=2)	23.68	4.05	1921	1000	678	4661	2109	1347
E2GC (G=4)	23.82	4.10	1476	804	667	3431	1674	1292
E2GC (G=8)	24.09	4.20	**1162**	742	631	2337	1318	1152
E2GC (G=16)	24.63	4.40	1132	722	619	2220	1251	1080
E2GC (G=32)	25.72	4.80	1088	694	597	2063	1189	1049
FgGC (g=2)	46.18	7.70	1485	878	695	2702	1572	1276
FgGC (g=4)	34.86	5.85	1241	761	638	2181	1391	1155
FgGC (g=8)	29.20	4.92	1142	730	612	2055	1302	1129
FgGC (g=16)	26.37	4.46	1131	722	624	2106	1295	1117
FgGC (g=32)	24.96	4.23	**1204**	766	687	2310	1356	1200

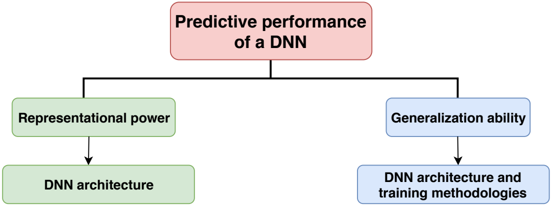
ResNeXt-50 with E2GC module is **4%** and **3.5%** more energy-efficient and parameter-efficient (respectively) than **ResNeXt-50 with F2GC** module (#MACs ≈ 4.2B) ☺

Predictive performance of a network



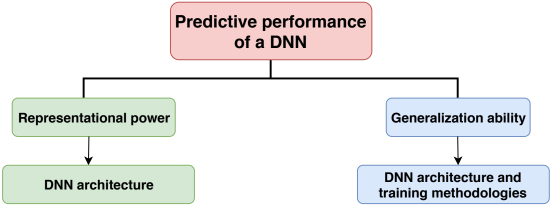
- GConv acts as *implicit* regularizer.
  - Structured **DropConnect** [ICML'13].
  - **DropBlock** [NeurIPS'18] .
- Changing G affects regularization.
  - Lower $G \implies$ stronger regularization.
  - Higher $G \implies$ weaker regularization.

Predictive performance of a network



- GConv acts as *implicit* regularizer.
  - Structured **DropConnect** [ICML'13].
  - **DropBlock** [NeurIPS'18] .
- Changing G affects regularization.
  - Lower G $\implies$ stronger regularization.
  - Higher G $\implies$ weaker regularization.
- Changing G alters the **representational** power of network.
  - Higher G captures more variations of complex concepts.

Predictive performance of a network



- GConv acts as *implicit* regularizer.
  - Structured **DropConnect** [ICML'13].
  - **DropBlock** [NeurIPS'18] .
- Changing G affects regularization.
  - Lower $G \implies$ stronger regularization.
  - Higher $G \implies$ weaker regularization.
- Changing G alters the **representational** power of network.
  - Higher G captures more variations of complex concepts.

Key observation: Changing G enables a trade-off between **representational power** and **generalization** ability of network.

Results: Predictive performance

Table: Top-1 accuracy on ImageNet-1k with E2GC module.

E2GC module	Params ($\times 10^6$)	MACs ($\times 10^6$)	ImageNet-1K (Top-1 %)
MobileNet-V1 (G=1)	4.20	568.74	**70.65**
MobileNet-V1 (G=2)	4.25	586.13	**72.76**
MobileNet-V1 (G=4)	4.34	620.90	72.24
MobileNet-V1 (G=8)	4.52	690.44	71.85
MobileNet-V1 (G=16)	4.87	829.53	71.18
MobileNet-V1 (G=32)	5.59	1107.71	72.76
ResNeXt-50 (G=1)	23.61	4.02	**74.43**
ResNeXt-50 (G=2)	23.68	4.05	**77.04**
ResNeXt-50 (G=4)	23.82	4.10	77.22
ResNeXt-50 (G=8)	24.09	4.20	77.60
ResNeXt-50 (G=16)	24.63	4.40	77.64
ResNeXt-50 (G=32)	25.72	4.80	77.45

Results: Predictive performance

Table: Top-1 accuracy on ImageNet-1k with E2GC module.

E2GC module	Params ($\times 10^6$)	MACs ($\times 10^6$)	ImageNet-1K (Top-1 %)
MobileNet-V1 (G=1)	4.20	568.74	**70.65**
MobileNet-V1 (G=2)	4.25	586.13	**72.76**
MobileNet-V1 (G=4)	4.34	620.90	72.24
MobileNet-V1 (G=8)	4.52	690.44	71.85
MobileNet-V1 (G=16)	4.87	829.53	71.18
MobileNet-V1 (G=32)	5.59	1107.71	72.76
ResNeXt-50 (G=1)	23.61	4.02	**74.43**
ResNeXt-50 (G=2)	23.68	4.05	**77.04**
ResNeXt-50 (G=4)	23.82	4.10	77.22
ResNeXt-50 (G=8)	24.09	4.20	77.60
ResNeXt-50 (G=16)	24.63	4.40	77.64
ResNeXt-50 (G=32)	25.72	4.80	77.45

Takeaway: Depthwise convolution has low representational power and increasing G enables network to extract more semantic information.

Conclusion

GConv enables a trade-off between computational cost and memory cost; between **representational power** and **generalization ability**.

- Fixing g in layers of a DNNs is *not amenable* for higher energy efficiency.
  - Selection of g/G should balance the computation with data reuse.
- Depthwise convolution inefficient in extracting semantic features.
  - Increases G helps in achieving better representational power.

Thanks for your attention..!!!

Thanks for your attention..!!!

Q & A?

code available at
https://github.com/iithcandle/E2GC-release

Additional slide: 1

MobileNet-V1	Params ($\times 10^6$)	MACs ($\times 10^6$)	EPF on P100 (millijoule)			EPF on P4000 (millijoule)			Accuracy (Top-1 %)	
			B=1	B=4	B=16	B=1	B=4	B=16	ImageNet-1K	Food-101
E2GC (G=1)	4.20	568.74	689	630	613	1607	1448	1373	**70.65**	79.96
E2GC (G=2)	4.25	586.13	538	482	450	1169	1014	983	**72.76**	80.48
E2GC (G=4)	4.34	620.90	421	357	330	1005	846	816	72.24	80.23
E2GC (G=8)	4.52	690.44	373	316	302	780	630	592	71.85	80.03
E2GC (G=16)	4.87	829.53	370	284	277	689	551	507	71.18	79.79
E2GC (G=32)	5.59	1107.71	363	281	265	648	501	480	72.76	79.69
FgGC (g=2)	16.72	2690.06	476	388	383	785	673	631	74.43	78.60
FgGC (g=4)	10.44	1620.71	402	327	312	706	572	551	73.59	79.06
FgGC (g=8)	7.30	1086.03	372	297	281	704	561	550	73.34	79.60
FgGC (g=16)	5.73	818.69	384	308	292	719	598	591	72.60	80.03
FgGC (g=32)	4.95	685.02	418	334	321	811	698	647	72.20	80.13

Additional slide: 2

ResNeXt-50	Params ($\times 10^6$)	MACs ($\times 10^9$)	EPF on P100 (millijoule)			EPF on P4000 (millijoule)			Accuracy (Top-1 %)	
			B=1	B=4	B=16	B=1	B=4	B=16	ImageNet-1K	Food-101
E2GC (G=1)	23.61	4.02	2185	1248	712	5780	2371	1434	**74.43**	82.62
E2GC (G=2)	23.68	4.05	1921	1000	678	4661	2109	1347	**77.04**	82.78
E2GC (G=4)	23.82	4.10	1476	804	667	3431	1674	1292	77.22	82.72
E2GC (G=8)	24.09	4.20	1162	742	631	2337	1318	1152	77.60	82.10
E2GC (G=16)	24.63	4.40	1132	722	619	2220	1251	1080	77.64	82.03
E2GC (G=32)	25.72	4.80	1088	694	597	2063	1189	1049	77.45	81.46
FgGC (g=2)	46.18	7.70	1485	878	695	2702	1572	1276	77.58	78.03
FgGC (g=4)	34.86	5.85	1241	761	638	2181	1391	1155	77.62	78.16
FgGC (g=8)	29.20	4.92	1142	730	612	2055	1302	1129	77.64	78.88
FgGC (g=16)	26.37	4.46	1131	722	624	2106	1295	1117	77.72	80.00
FgGC (g=32)	24.96	4.23	1204	766	687	2310	1356	1200	77.80	80.05

Additional slide: 3

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

Andrew G. Howard Menglong Zhu Bo Chen Dmitry Kalenichenko
Weijun Wang Tobias Weyand Marco Andreetto Hartwig Adam

Google Inc.
{howarda,menglong,bochen,dkalenichenko,weijunw,weyand,anm,hadam}@google.com

4.1. Model Choices

First we show results for MobileNet with depthwise separable convolutions compared to a model built with full convolutions. In Table 4 we see that using depthwise separable convolutions compared to full convolutions only reduces

accuracy by 1% on ImageNet was saving tremendously on mult-adds and parameters.

We next show results comparing thinner models with width multiplier to shallower models using less layers. To make MobileNet shallower, the 5 layers of separable filters with feature size $14 \times 14 \times 512$ in Table 1 are removed. Table 5 shows that at similar computation and number of parameters, that making MobileNets thinner is 3% better than making them shallower.

Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2