Maior Dúvida da Aula

1. Como podemos identificar se cada camada da arquitetura está funcionando corretamente?
2. As FCs são obrigatoriamente as últimas? Ou podem estar em outra posição?
3. Se o objetivo do pooling é diminuir o tamanho da matriz, por que não só aplicar mais uma camada de convolução?
4. É realmente impossível obter resultados satisfatórios ao treinar do zero uma rede neural convolucional de muitos parâmetros com poucos dados?
5. Existe algum teorema que define um melhor hiperparâmetro (como tamanho do filtro) ou alguma forma de determinar sem ser experimentalmente algum dos componentes de arquiteturas de rede, como as de CNNs?

6. Não compreendi muito bem como o dropblock atua na camada convolucional. São desconsiderados filtros aleatórios do bloco convolucional ?

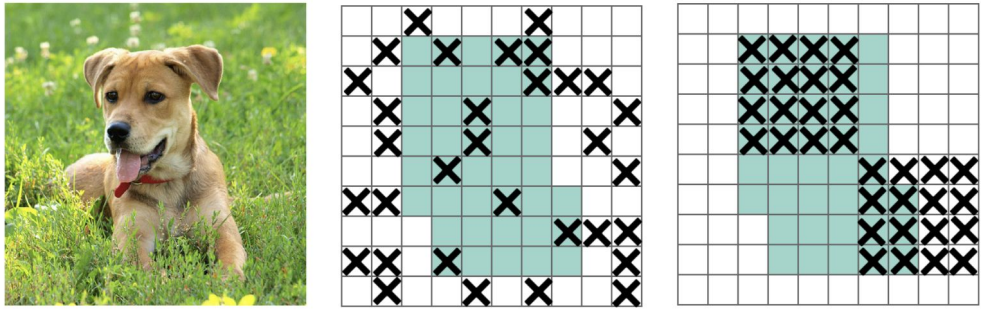
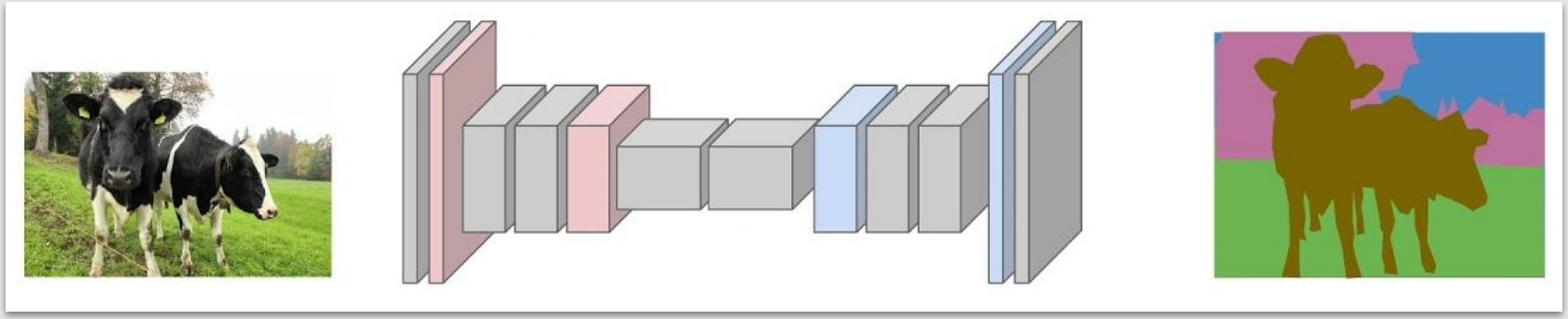


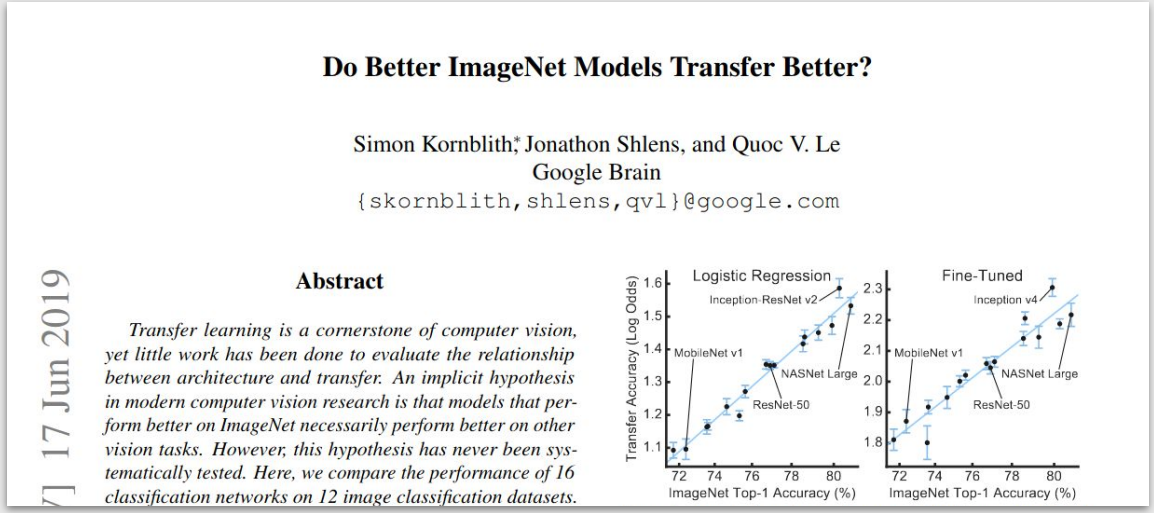
Figure 1: (a) input image to a convolutional neural network. The green regions in (b) and (c) include the activation units which contain semantic information in the input image. Dropping out activations at random is not effective in removing semantic information because nearby activations contain closely related information. Instead, dropping continuous regions can remove certain semantic information (e.g., head or feet) and consequently enforcing remaining units to learn features for classifying input image.

"DropBlock: A regularization method for convolutional networks", NeurIPS 2018

7. Nas arquiteturas de CNN que foram mostradas em aula, as dimensões dos mapas de ativação, ou diminuem ou são mantidos constantes, à medida que os dados passam pelas camadas das redes. Faz sentido aumentar as dimensões dos mapas de ativação em alguma camada da rede?



8. Gostaria de saber se as arquiteturas de melhor acurácia no ImageNet também são as melhores em tarefas de classificação de imagens em datasets mais “especializados” - como nos voltados unicamente a identificação de melanomas, sinais de trânsito, etc.



Do Better ImageNet Models Transfer Better?

Simon Kornblith, Jonathon Shlens, and Quoc V. Le
Google Brain
{skornblith,shlens,qvl}@google.com

Abstract

Transfer learning is a cornerstone of computer vision, yet little work has been done to evaluate the relationship between architecture and transfer. An implicit hypothesis in modern computer vision research is that models that perform better on ImageNet necessarily perform better on other vision tasks. However, this hypothesis has never been systematically tested. Here, we compare the performance of 16 classification networks on 12 image classification datasets.

CVPR 2019 (oral), https://arxiv.org/pdf/1805.08974.pdf
CVPR 2019 https://arxiv.org/pdf/1904.12724v1.pdf

9. Quais seriam as melhores arquiteturas de redes neurais para o problema de classificação?

10. Gostaria de um panorama geral sobre o hardware das CNNs, qual placa de vídeo e processador melhor se adequa a determinadas arquiteturas, em quais parâmetros devemos nos basear, etc.

11. Fiquei com dúvida em como foi usado o transfer learning da VGG que você usou para o seu trabalho de lesão de pele. Porque você escolheu usar apenas as últimas duas camadas? Você usou a VGG para detectar oque era uma mancha e depois o SVM para classificar em benigna e maligna?

CNN Architectures

Machine Learning

Prof. Sandra Avila
Institute of Computing (IC/Unicamp)

MC886/MO444, November 1, 2022

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - GoogLeNet (2014)
  - ResNet (2015)

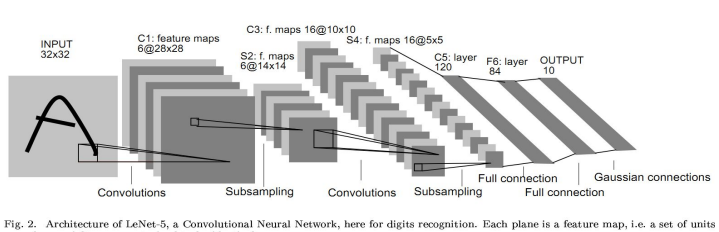
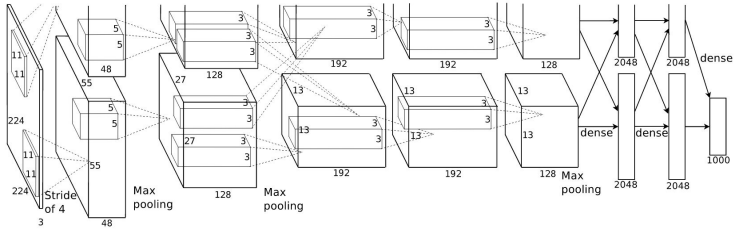
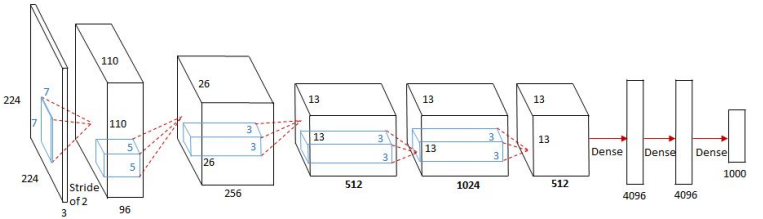
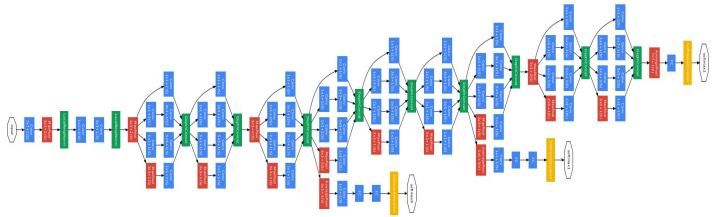
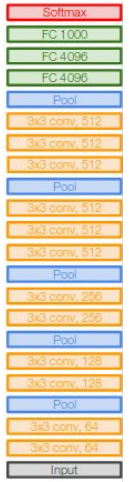


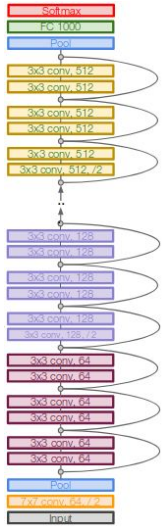
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.











Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - **GoogLeNet (2014)**
  - ResNet (2015)

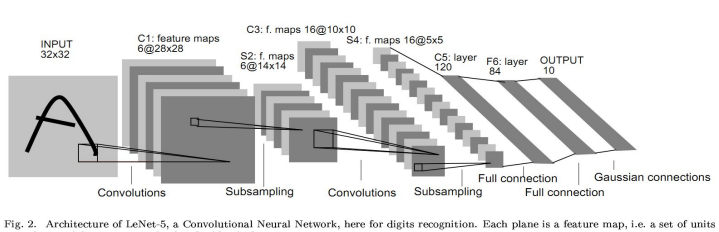
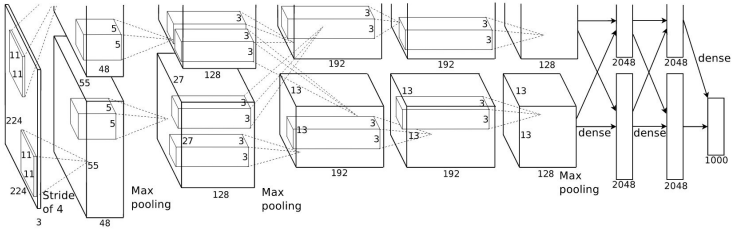
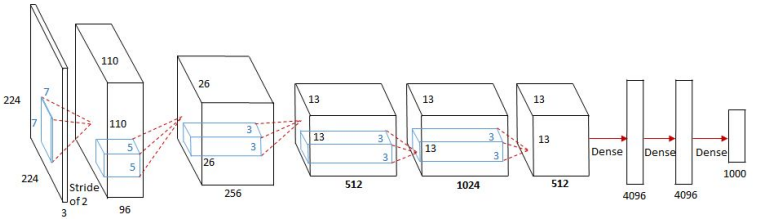
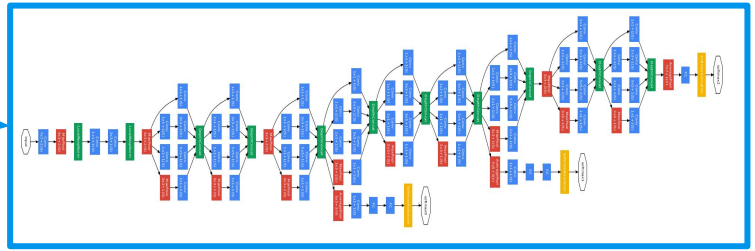
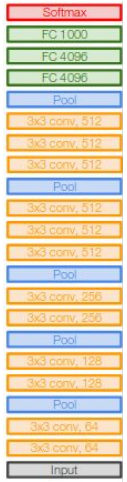


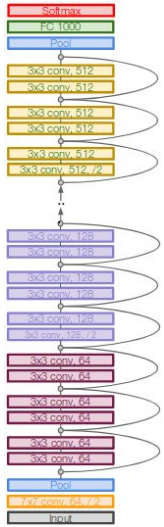
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

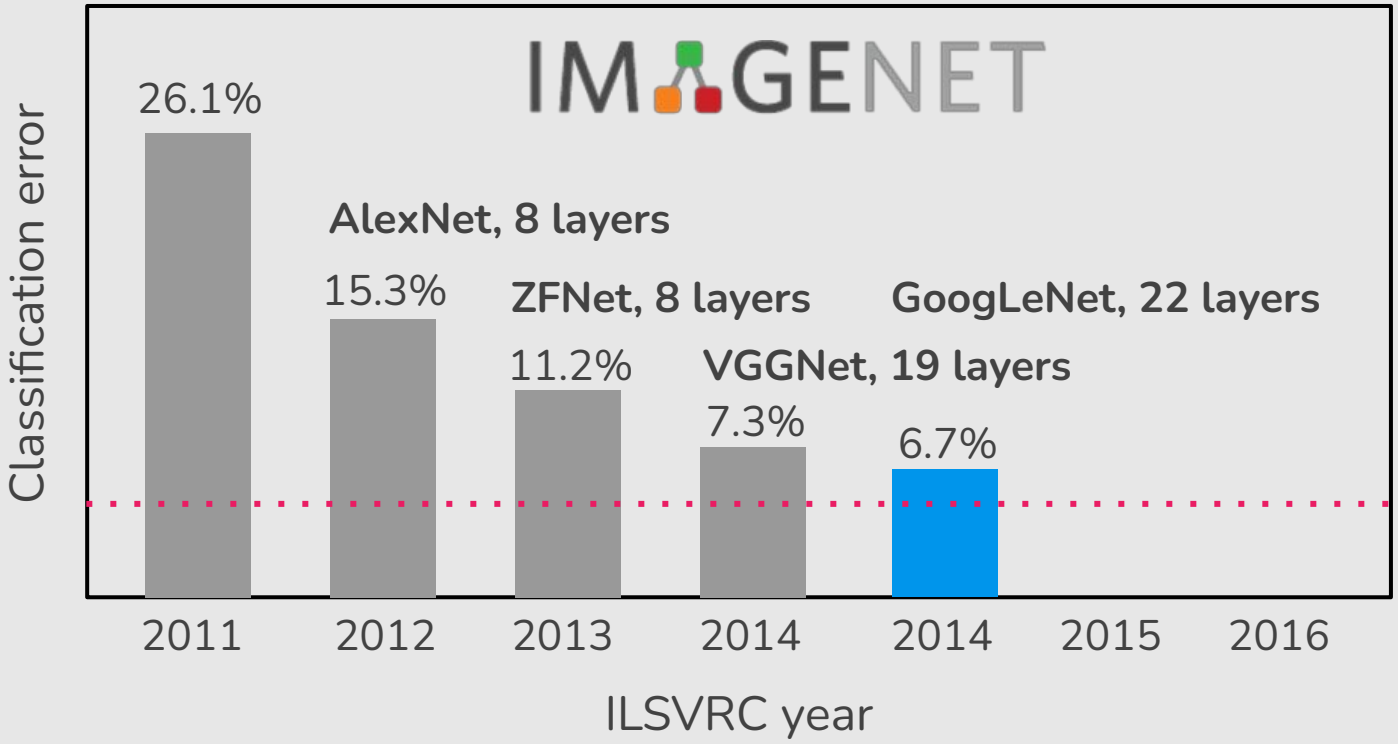












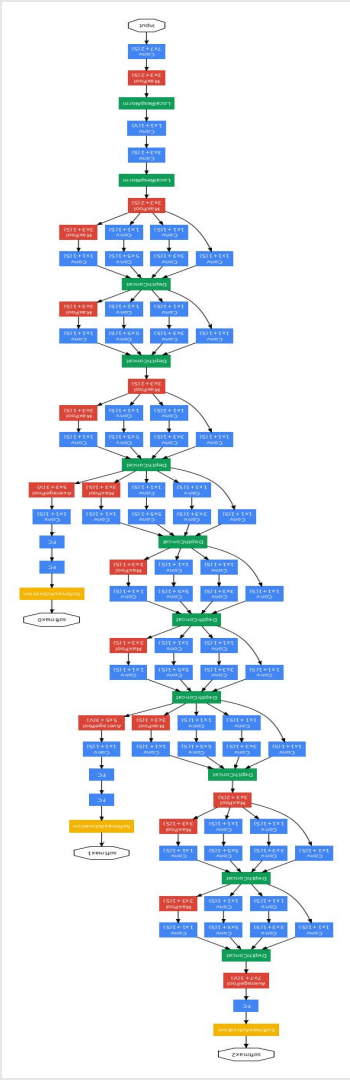
"Very Deep Convolutional Networks for Large-Scale Image Recognition", https://arxiv.org/pdf/1409.1556

GoogLeNet [Szegedy et al., 2014]

Deeper networks, with computational efficiency

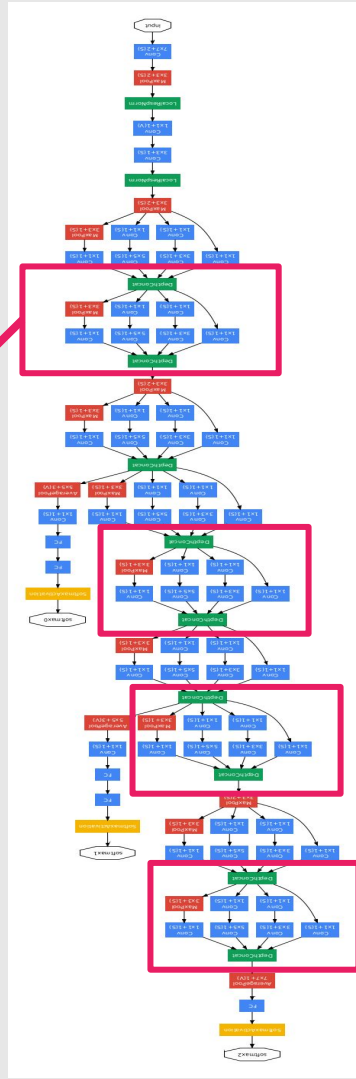
- 22 layers
- Inception module
- No FC layers
- Only 5 million parameters!

  12x less than AlexNet

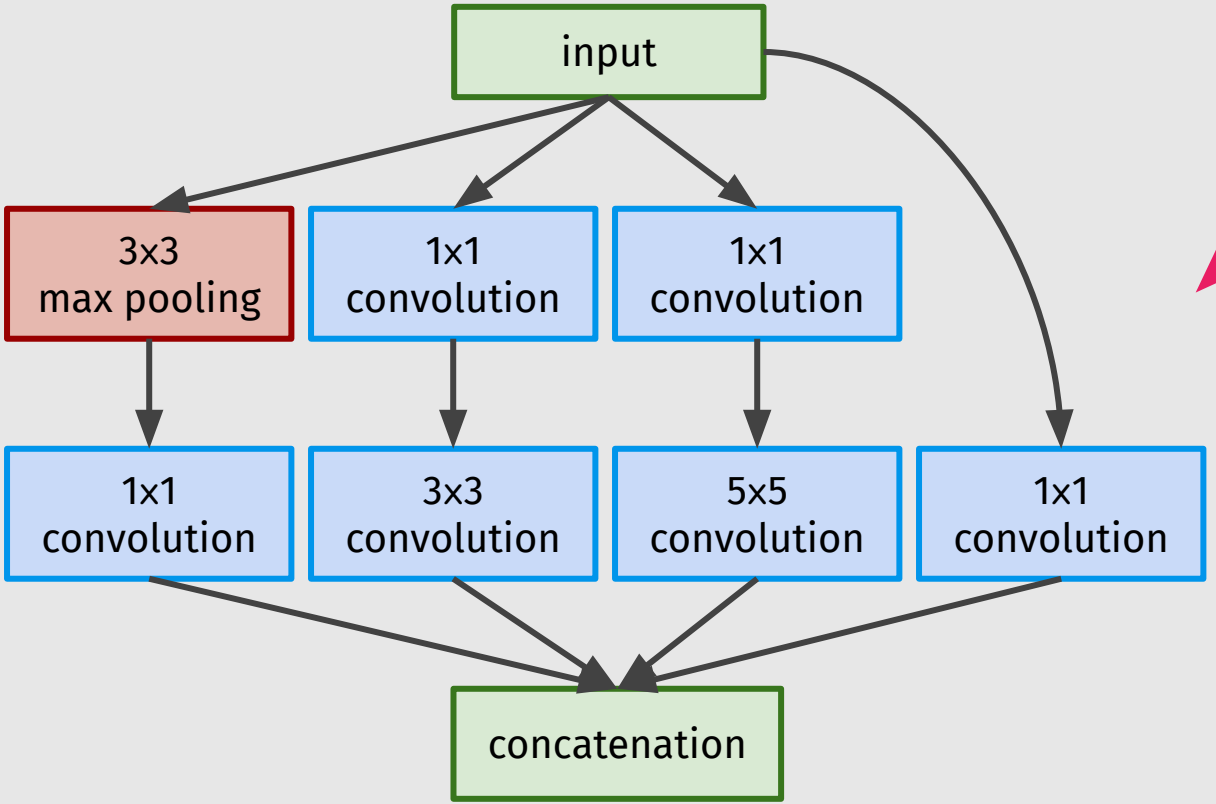


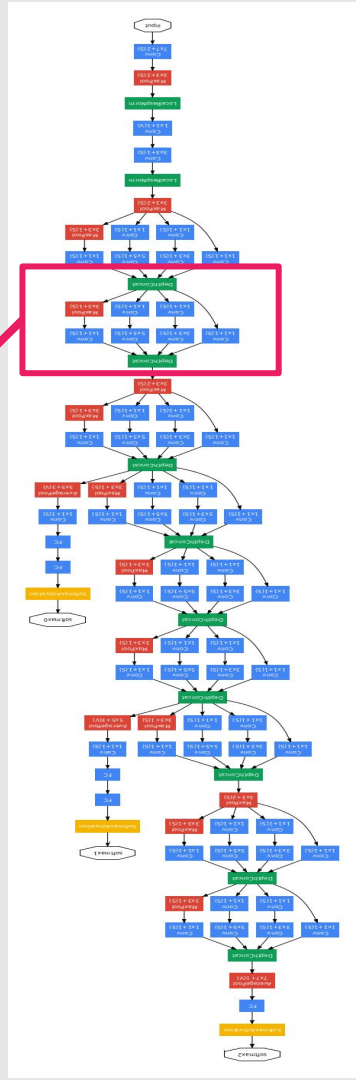
GoogLeNet [Szegedy et al., 2014]

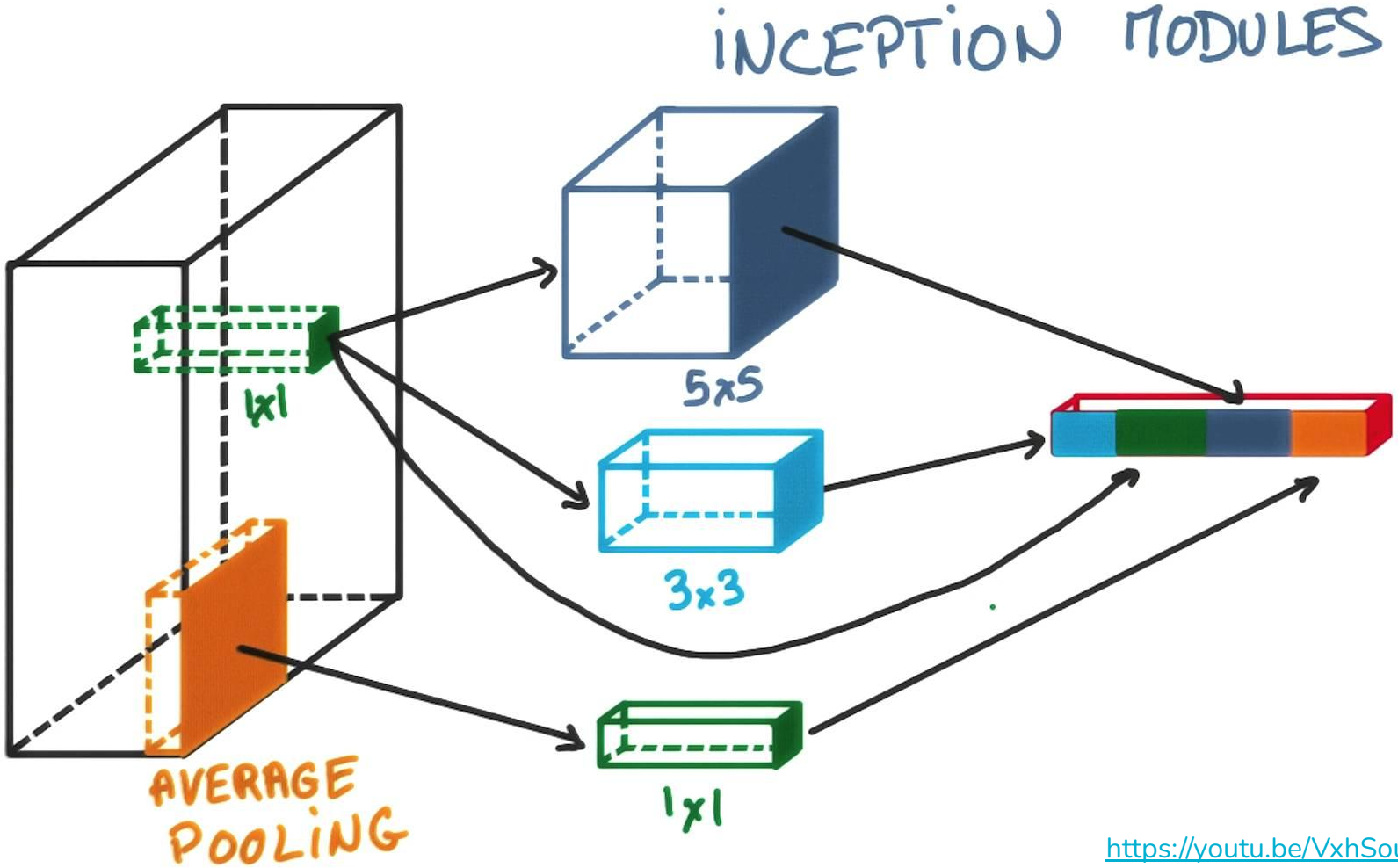
Inception module: design a good local network topology (network within a network) and then stack these modules on top of each other.



GoogLeNet [Szegedy et al., 2014]

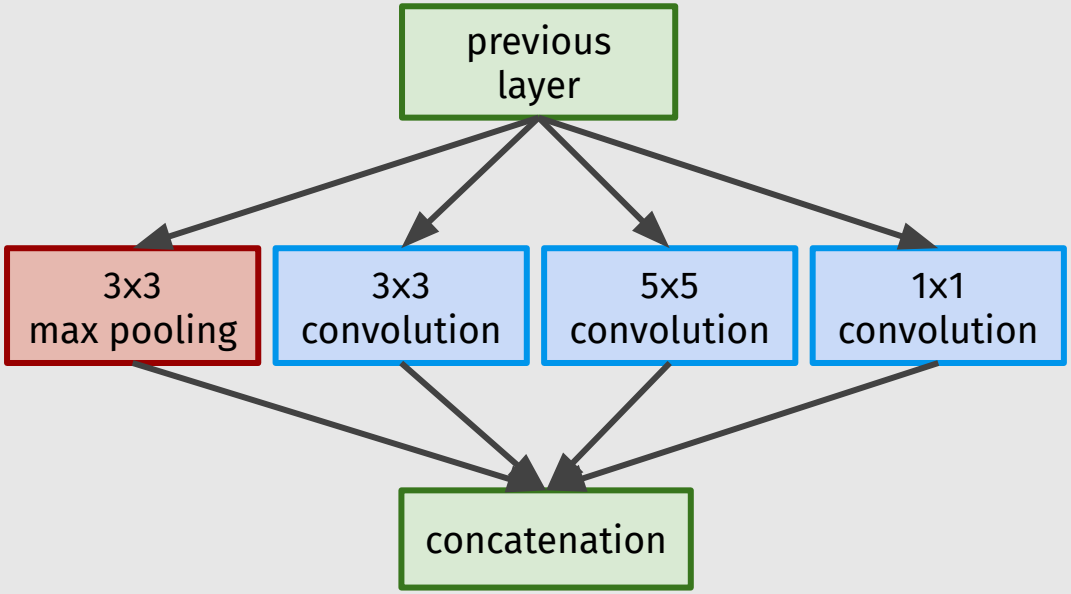






https://youtu.be/VxhSouuSZDY

GoogLeNet [Szegedy et al., 2014]



Naive Inception Module

Apply parallel filters on the input from previous layer:

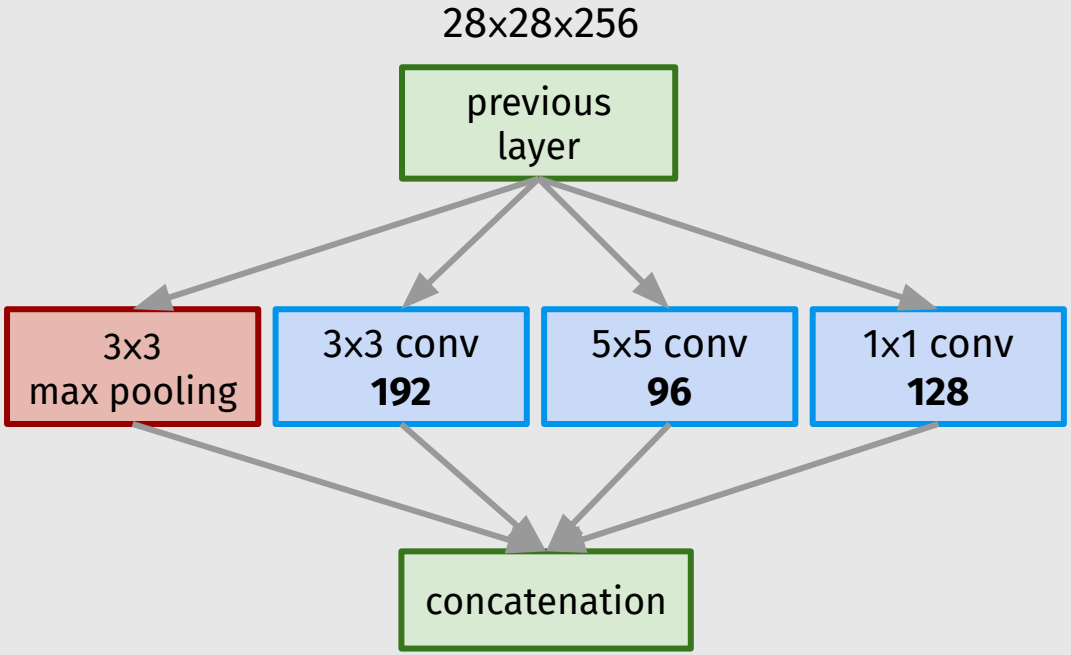
- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

Q: What is the problem with this? Computational complexity!

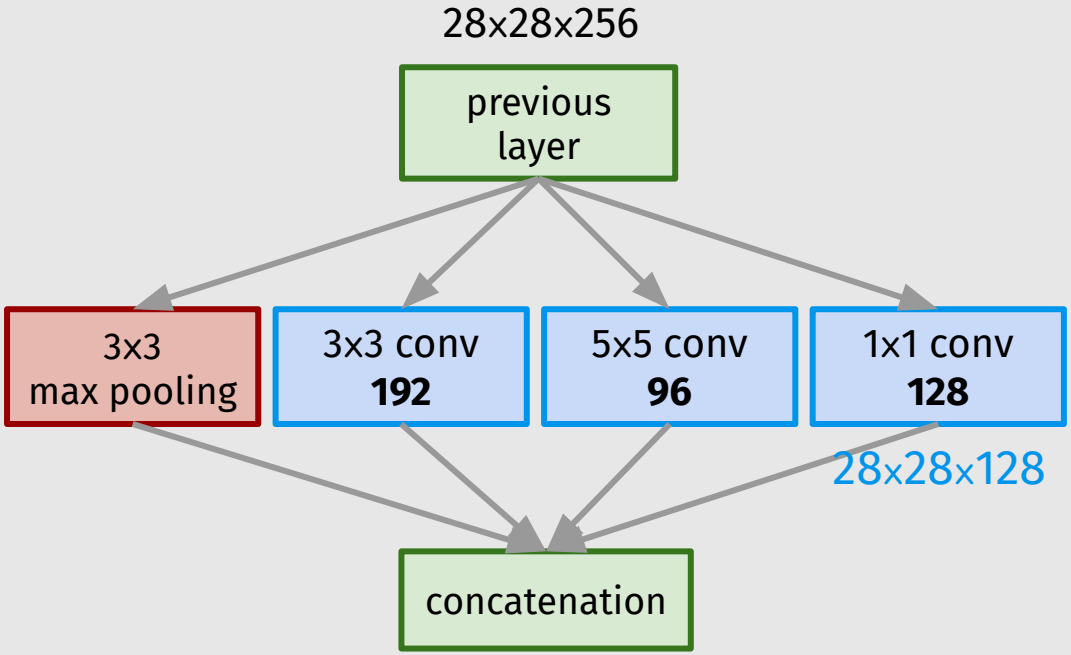
GoogLeNet [Szegedy et al., 2014]

Example: What is the output size of the **1x1 conv, with 128 filters**?



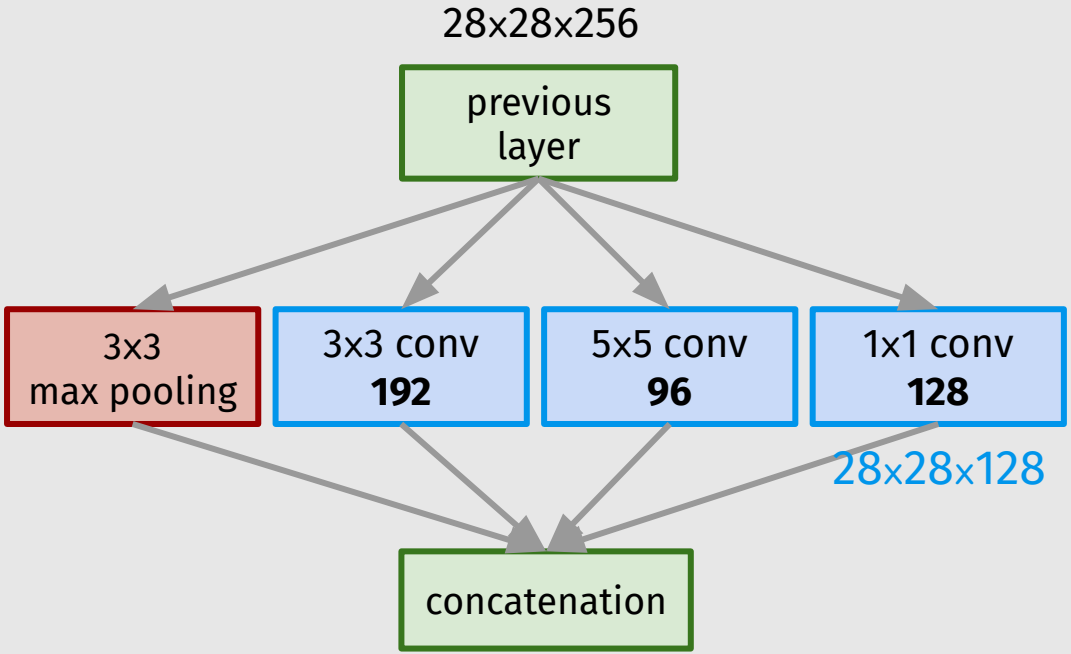
GoogLeNet [Szegedy et al., 2014]

Example: What is the output size of the **1x1 conv, with 128 filters**?



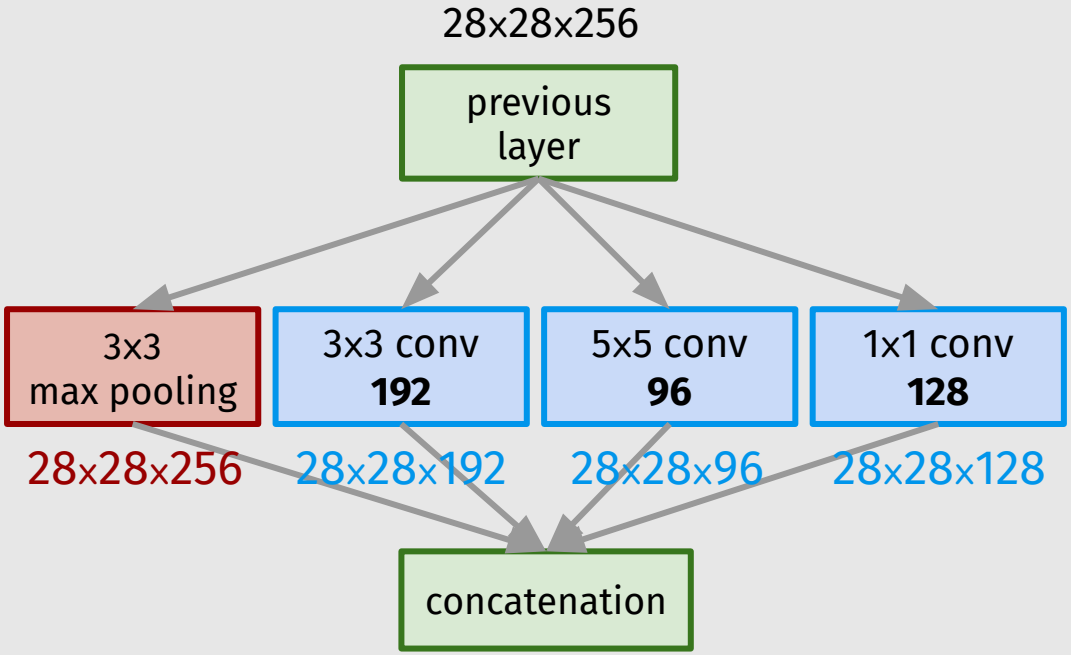
GoogLeNet [Szegedy et al., 2014]

Example: What are the output sizes of all different filter operations?



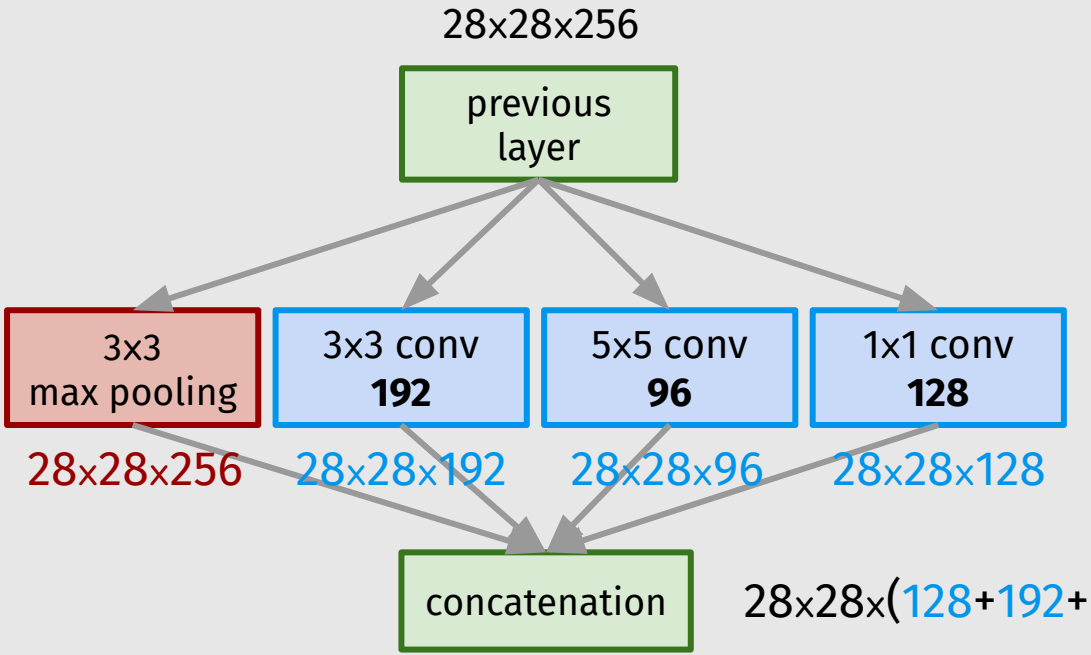
GoogLeNet [Szegedy et al., 2014]

Example: What are the output sizes of all different filter operations?



GoogLeNet [Szegedy et al., 2014]

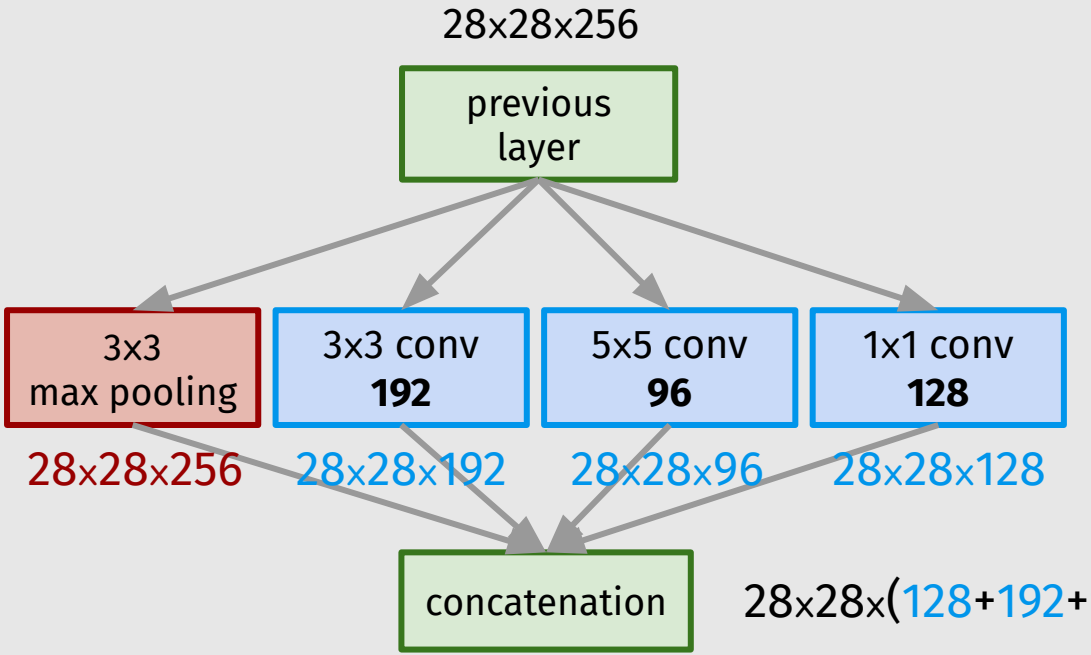
Example: What is output size after filter concatenation?



28x28x(128+192+96+256) = 28x28x672

GoogLeNet [Szegedy et al., 2014]

Example: What is output size after filter concatenation?

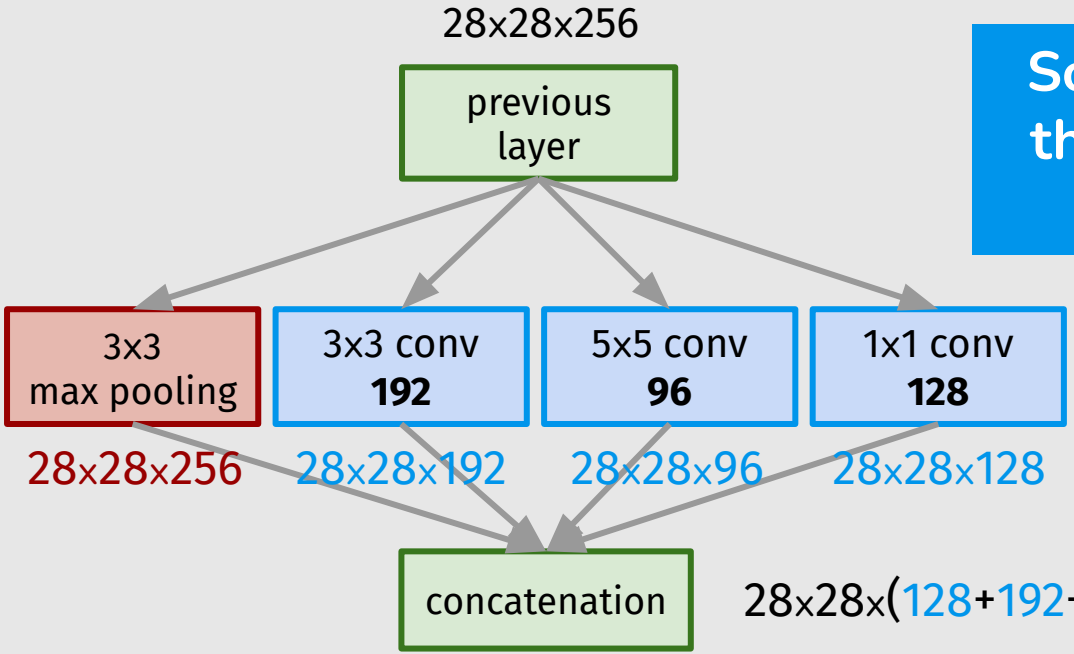


Total: 854M ops!!!

28x28x(128+192+96+256) = 28x28x672

GoogLeNet [Szegedy et al., 2014]

Example: What is output size after filter concatenation?



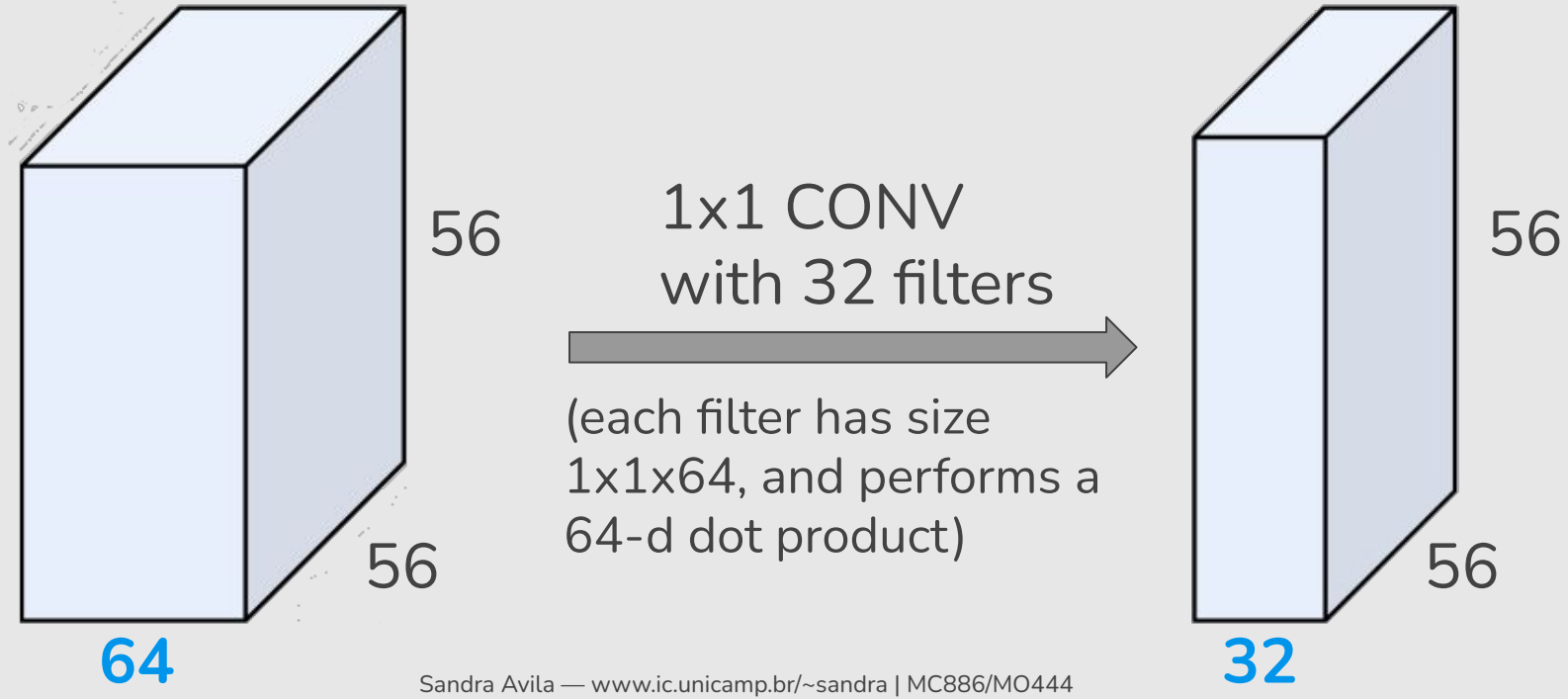
Solution: "bottleneck" layers that use 1x1 convolutions to reduce feature depth

Total: 854M ops!!!

28x28x(128+192+96+256) = 28x28x672

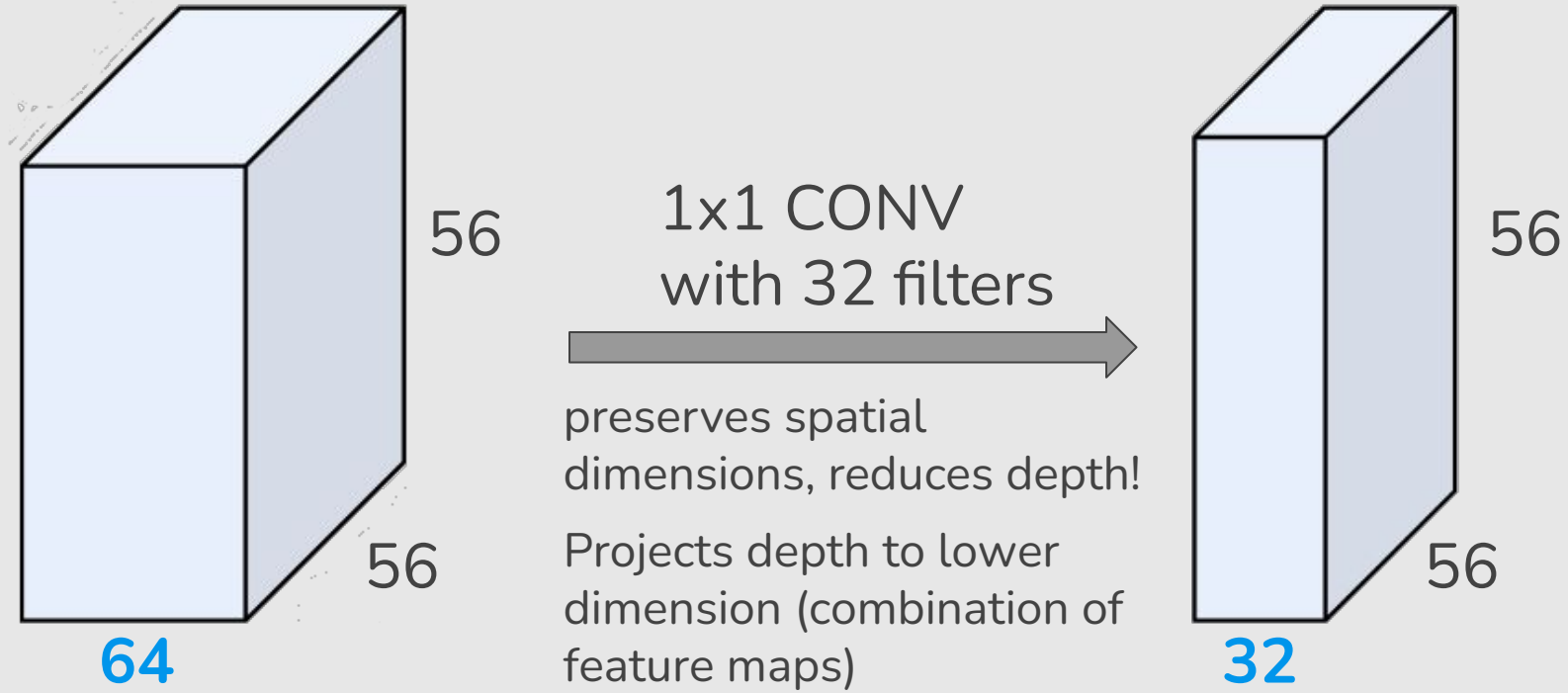
GoogLeNet [Szegedy et al., 2014]

Reminder: 1x1 convolutions

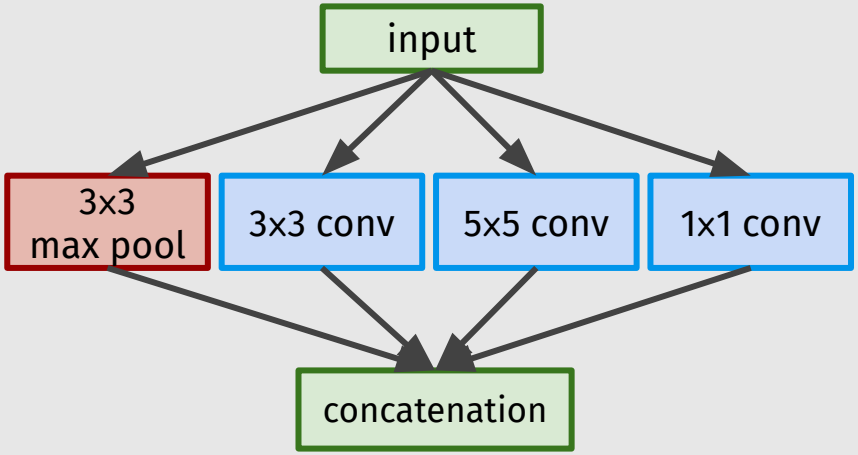


GoogLeNet [Szegedy et al., 2014]

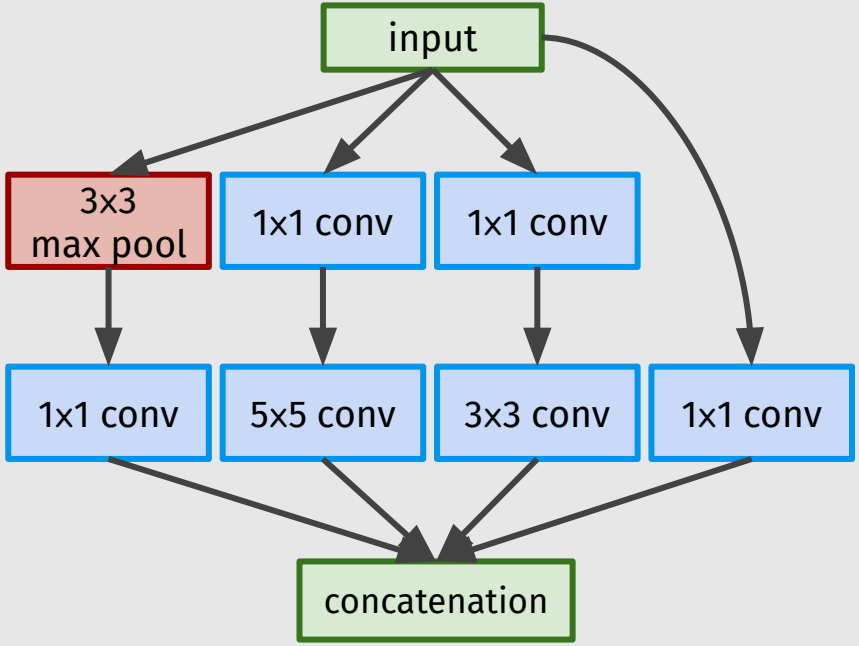
Reminder: 1x1 convolutions



GoogLeNet [Szegedy et al., 2014]



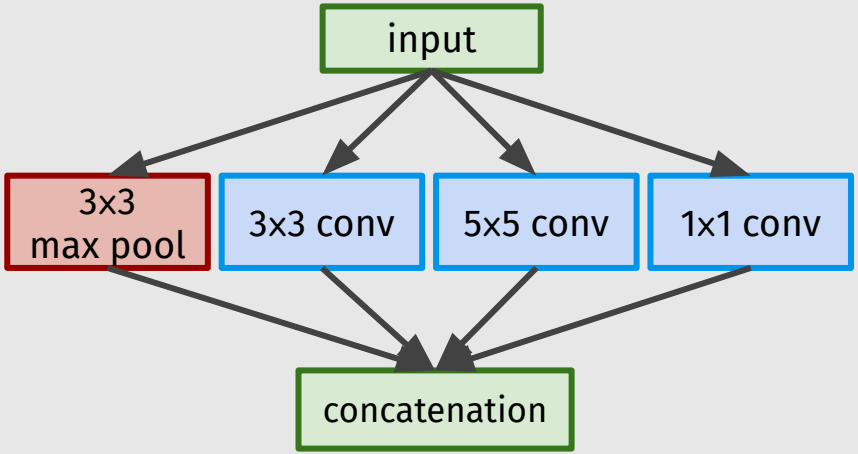
Naive Inception Module



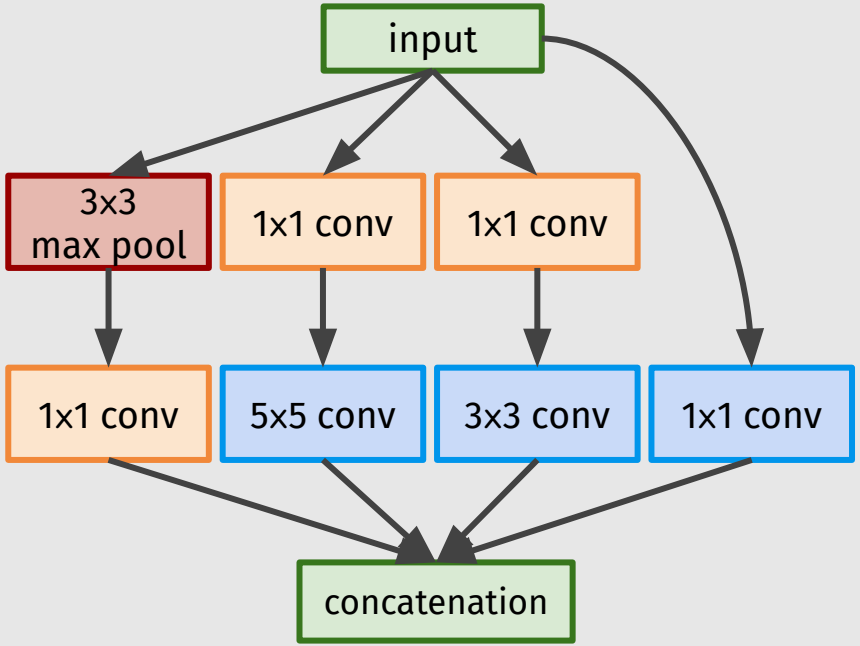
Inception Module

GoogLeNet [Szegedy et al., 2014]

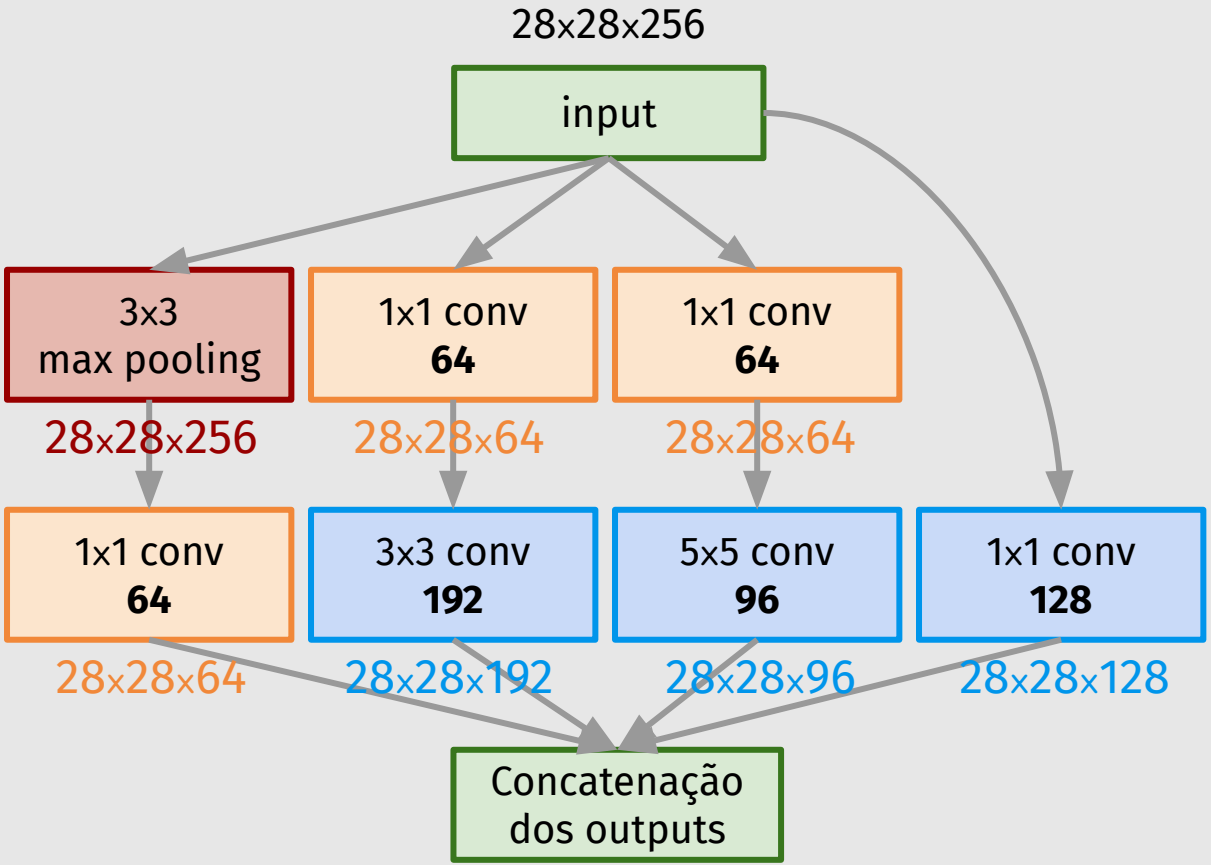
1x1 conv
“bottleneck” layers



Naive Inception Module



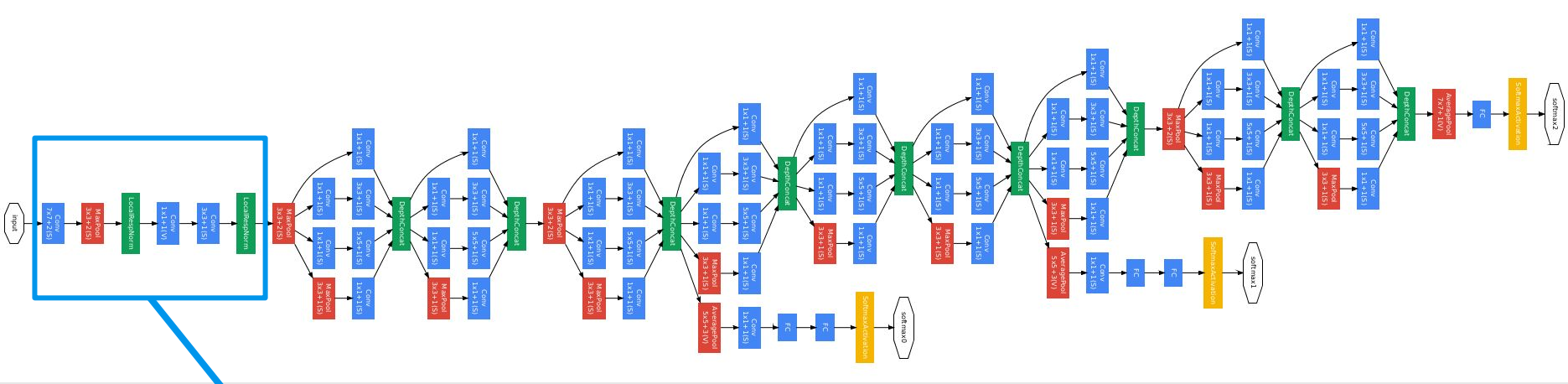
Inception Module



28x28x(128+192+96+64) = 28x28x480

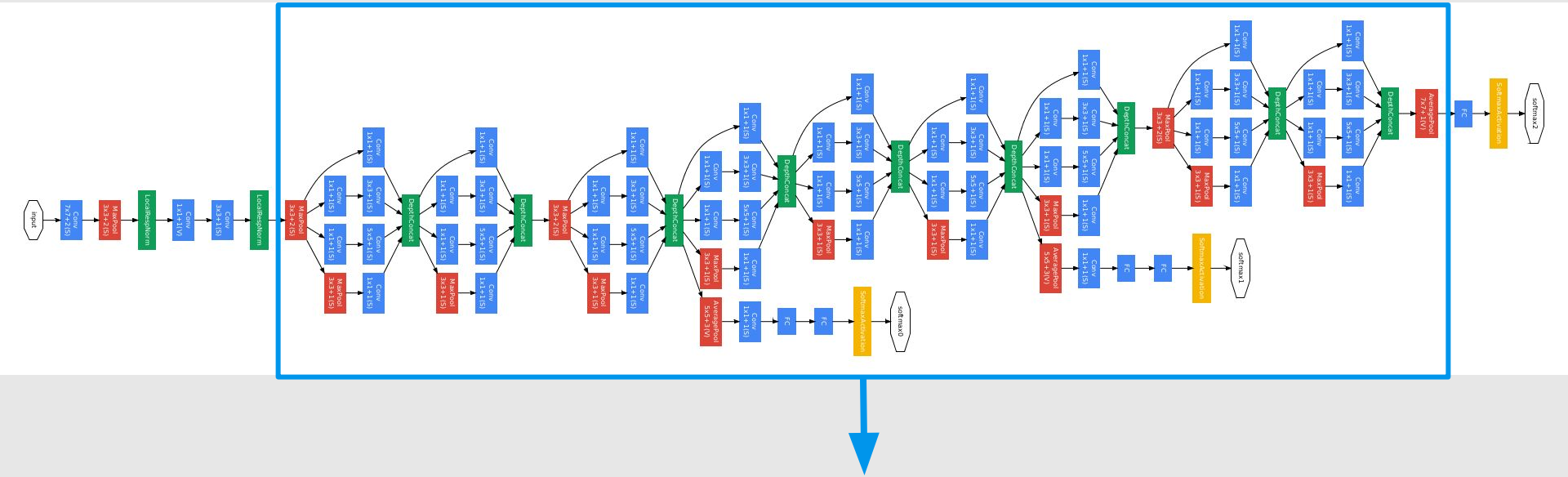
358M ops!!!
(vs. 854M)

GoogLeNet [Szegedy et al., 2014]



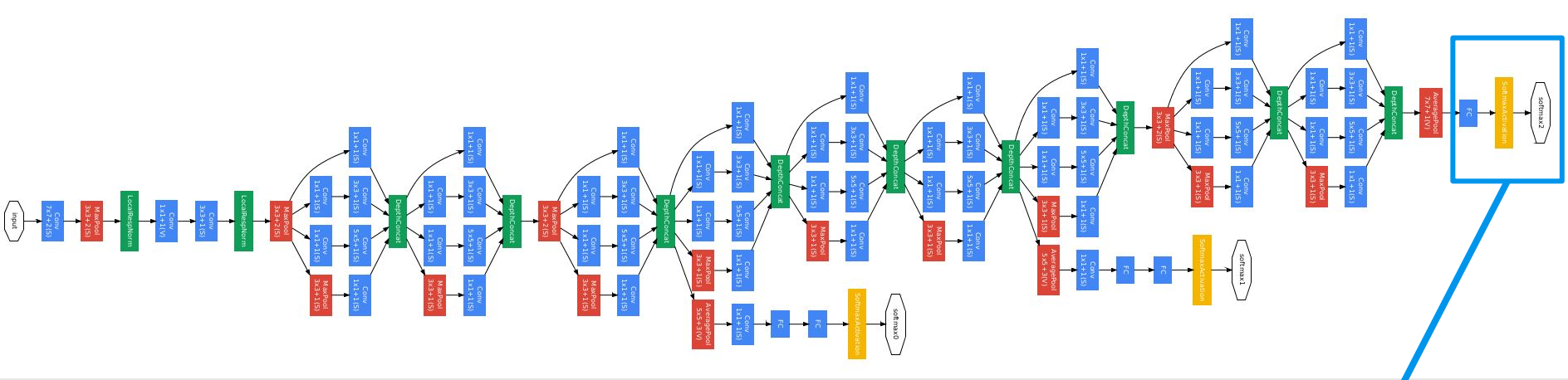
Conv-Pool 2x Conv-Pool

GoogLeNet [Szegedy et al., 2014]



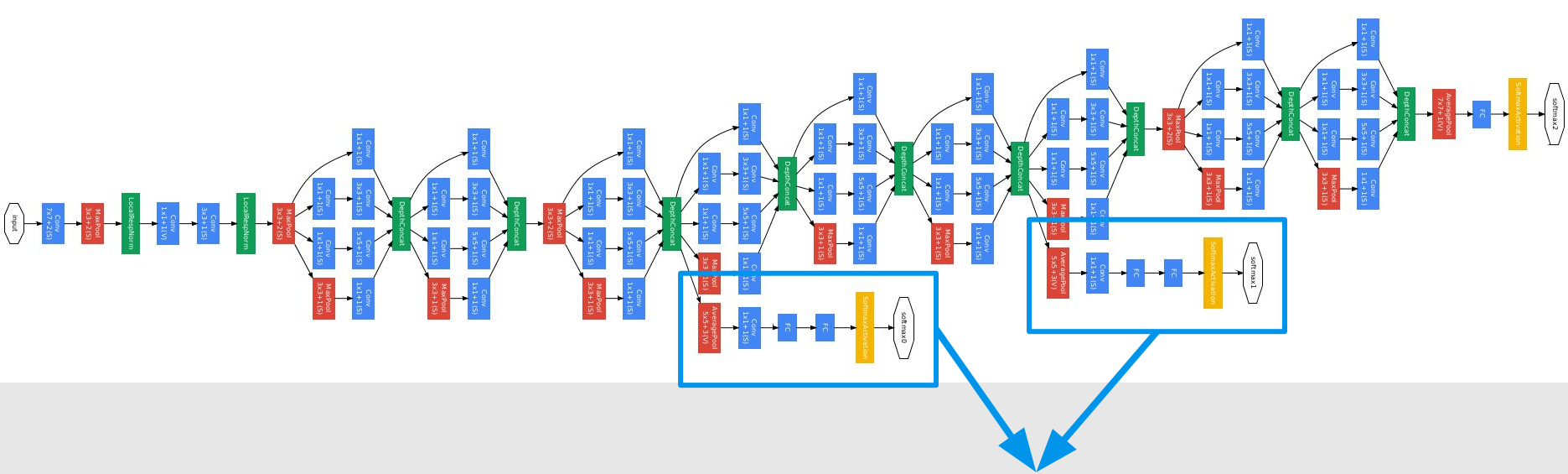
Stacked Inception Modules

GoogLeNet [Szegedy et al., 2014]

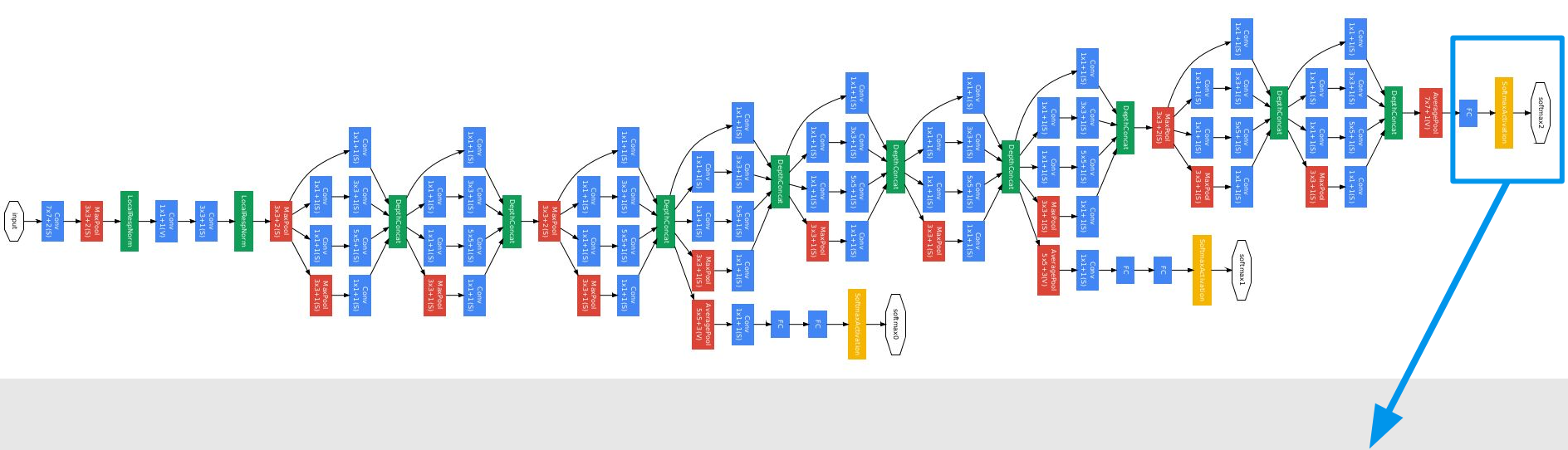


Classifier Output

GoogLeNet [Szegedy et al., 2014]



GoogLeNet [Szegedy et al., 2014]



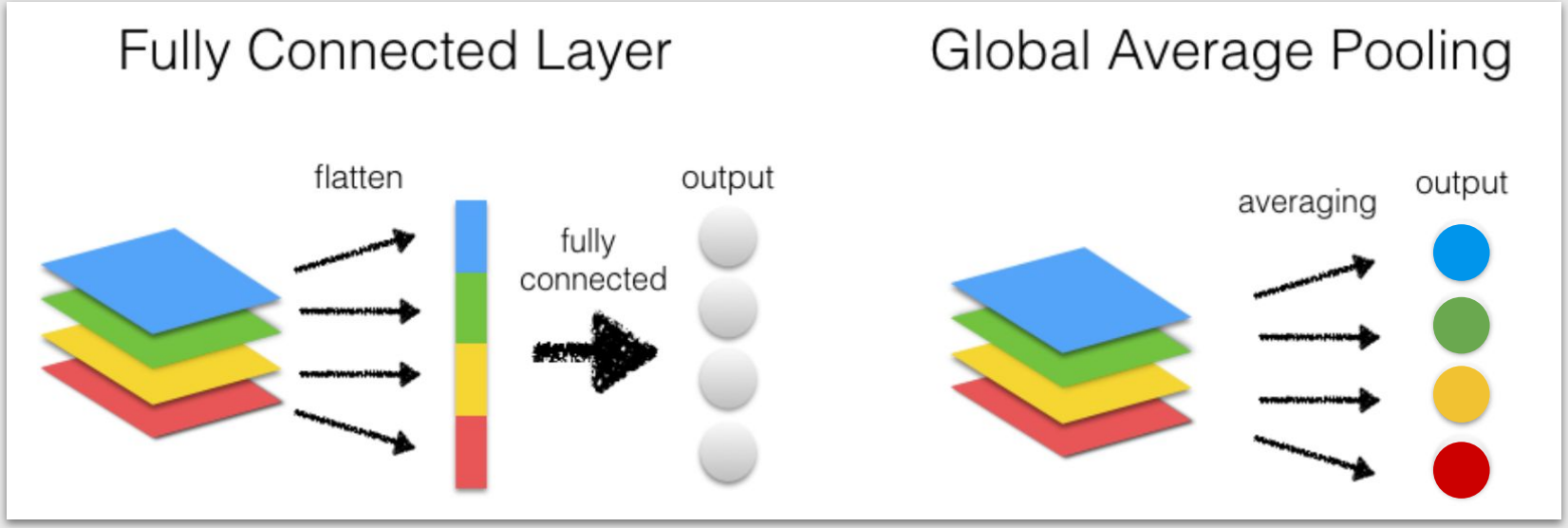
The total loss function is a weighted sum of the auxiliary loss and the real loss.

```
total_loss = real_loss + 0.3*aux_loss_1 + 0.3*aux_loss_2
```

GoogLeNet [Szegedy et al., 2014]

- GoogLeNet has **9 inception modules** stacked linearly.
- It is **22 layers deep** (27, including the pooling layers).
- It uses **global average pooling** at the end of the last inception module.
- GoogLeNet = Inception v1
- Inception v2, v3, v4, Inception-ResNet v1, v2:
  https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

GoogLeNet [Szegedy et al., 2014]



GoogLeNet [Szegedy et al., 2014]

- GoogLeNet has **9 inception modules** stacked linearly.
- It is **22 layers deep** (27, including the pooling layers).
- It uses **global average pooling** at the end of the last inception module.
- GoogLeNet = Inception v1
- Inception v2, v3, v4, Inception-ResNet v1, v2:
  https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - GoogLeNet (2014)
  - **ResNet (2015)**

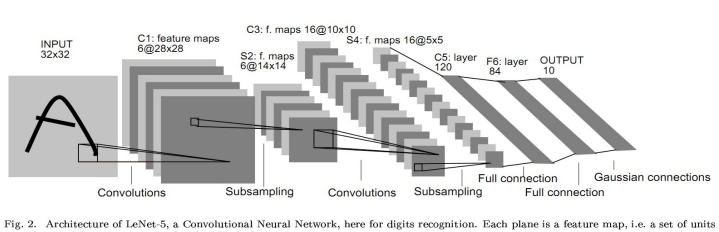
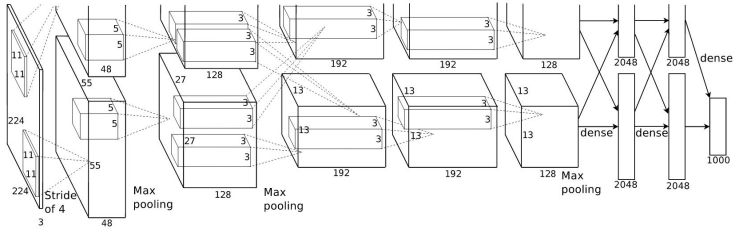
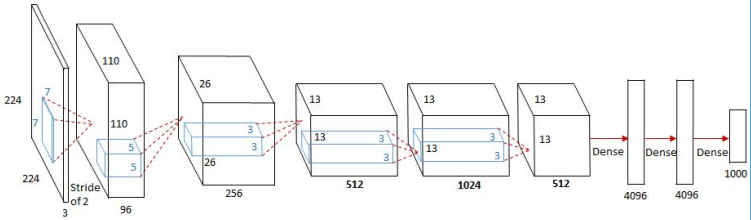
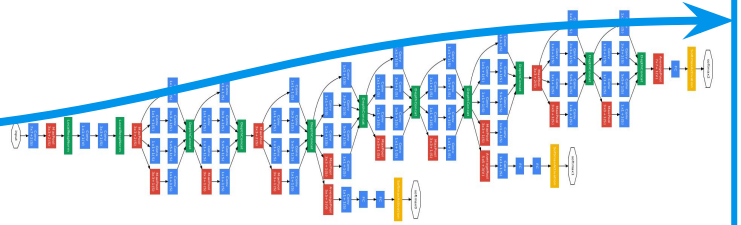
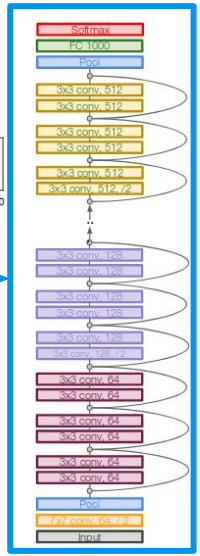


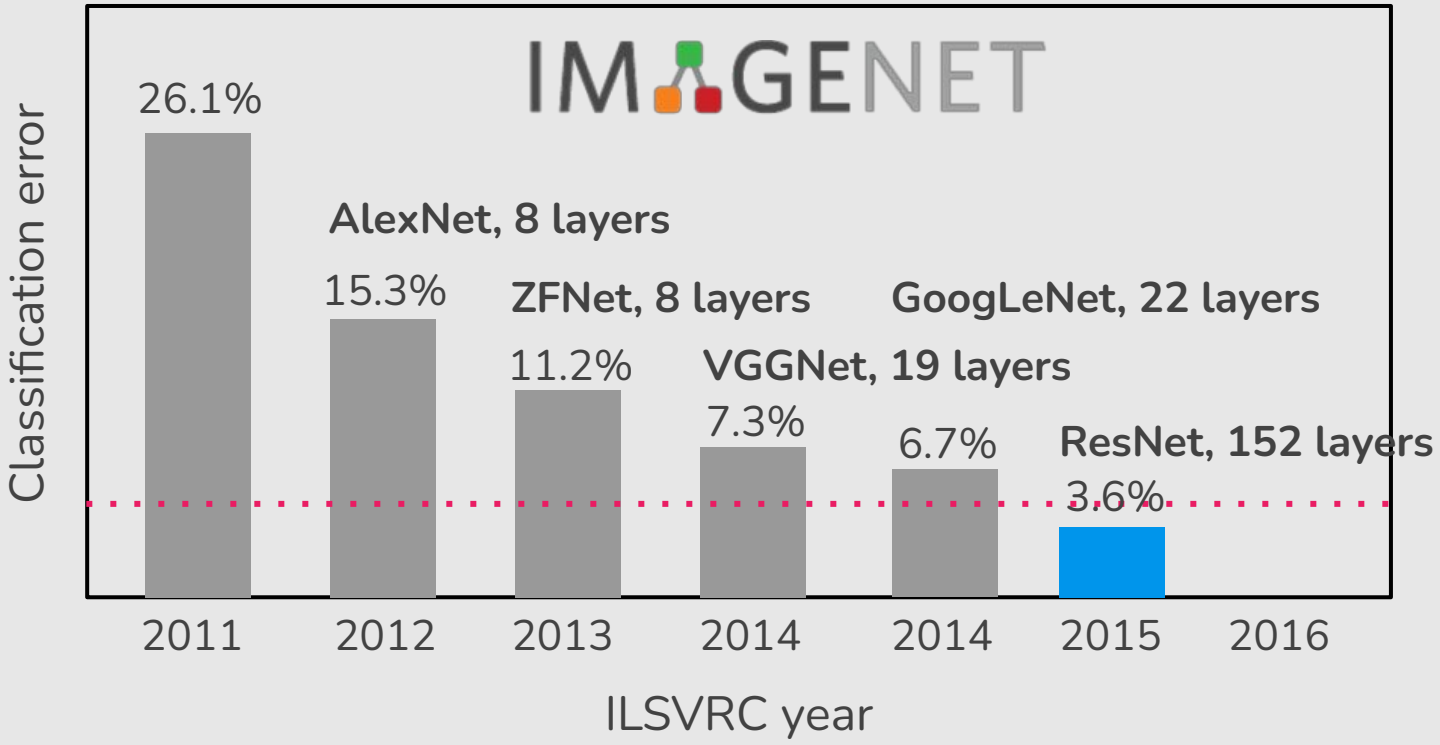
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.











"Deep Residual Learning for Image Recognition", CVPR 2016, https://arxiv.org/pdf/1512.03385

Today's Agenda

- CNN Architectures
  - LeNet (1998)
  - AlexNet (2012)
  - ZFNet (2013)
  - VGGNet (2014)
  - GoogLeNet (2014)
  - **ResNet (2015)**

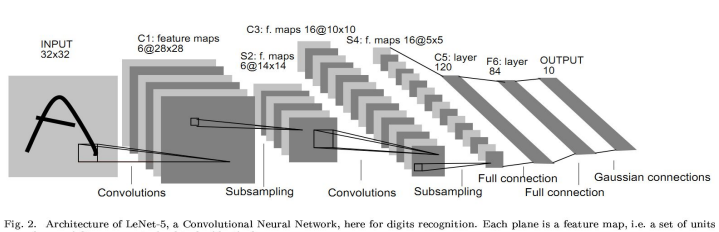
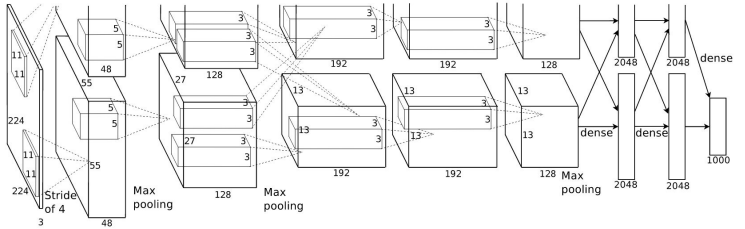
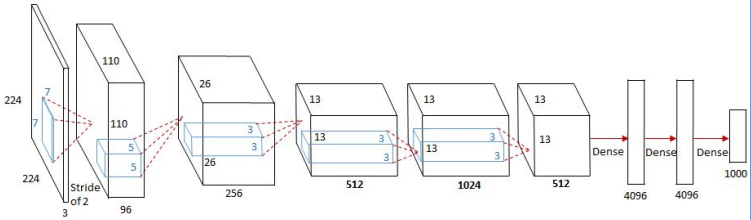
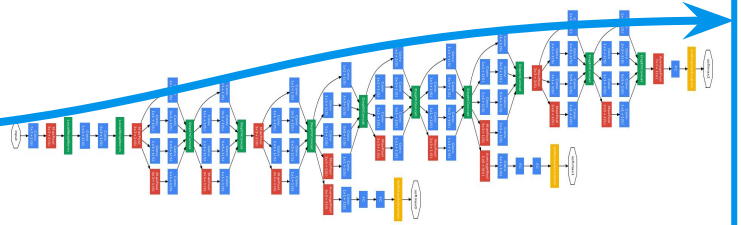
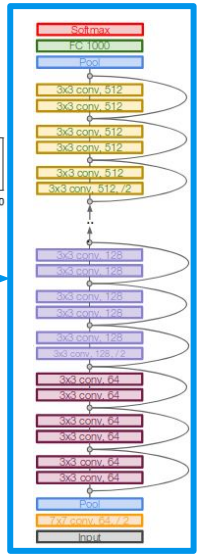


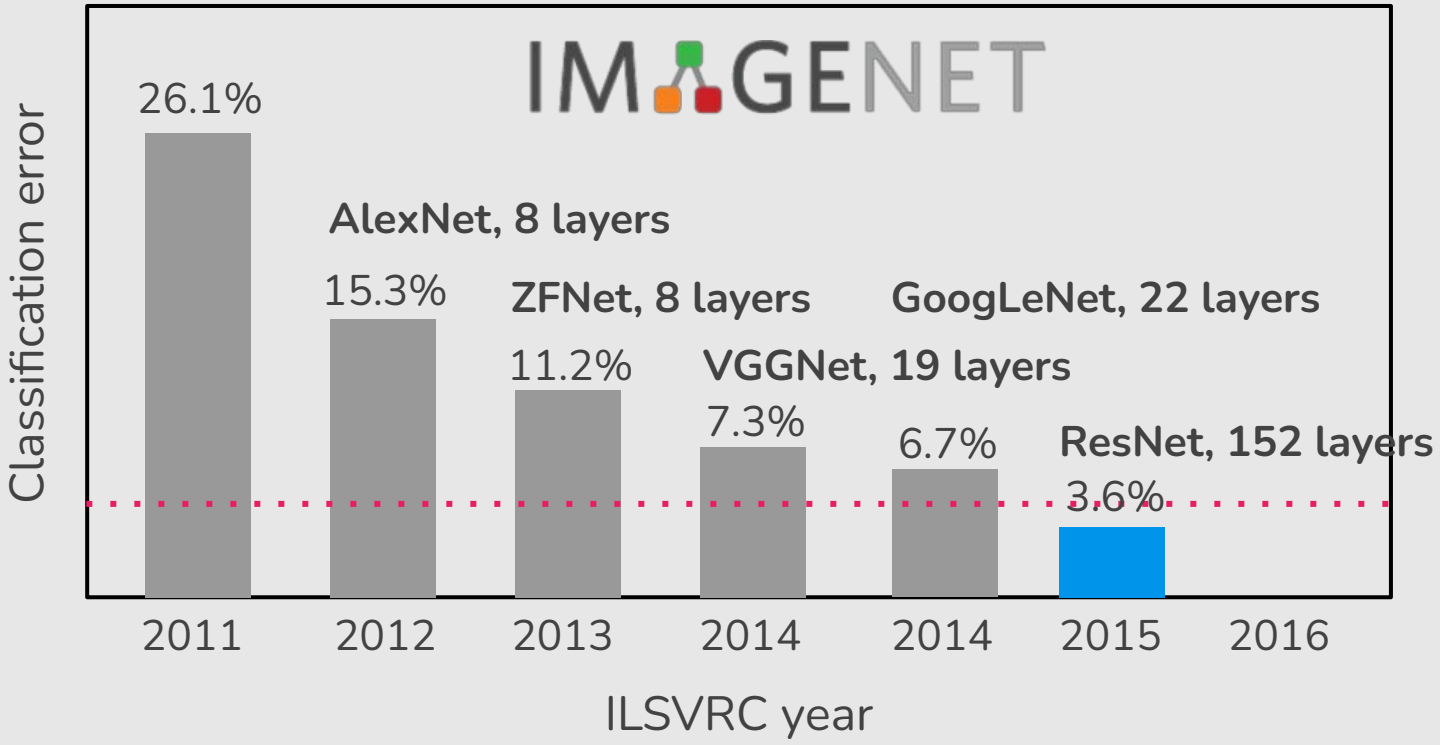
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.











"Deep Residual Learning for Image Recognition", CVPR 2016, https://arxiv.org/pdf/1512.03385

ResNet [He et al., 2015]

ResNet @ ILSVRC & COCO 2015 Competitions

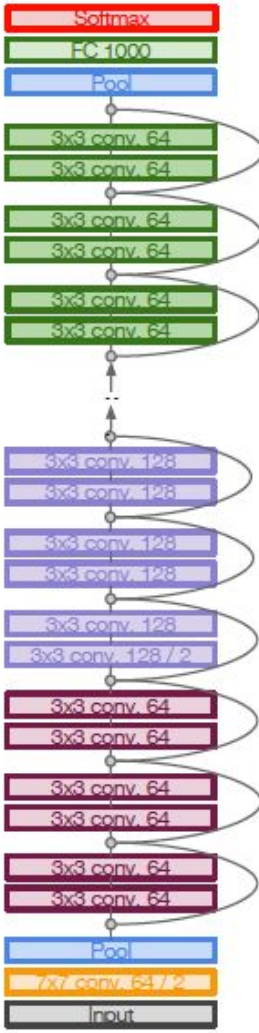
1st place in ALL five main tracks

- ImageNet Classification: “Ultra-deep” 152-layer nets
- ImageNet Detection: 16% better than 2nd
- ImageNet Localization: 27% better than 2nd
- COCO Detection: 11% better than 2nd
- COCO Segmentation: 12% better than 2nd

ResNet [He et al., 2015]

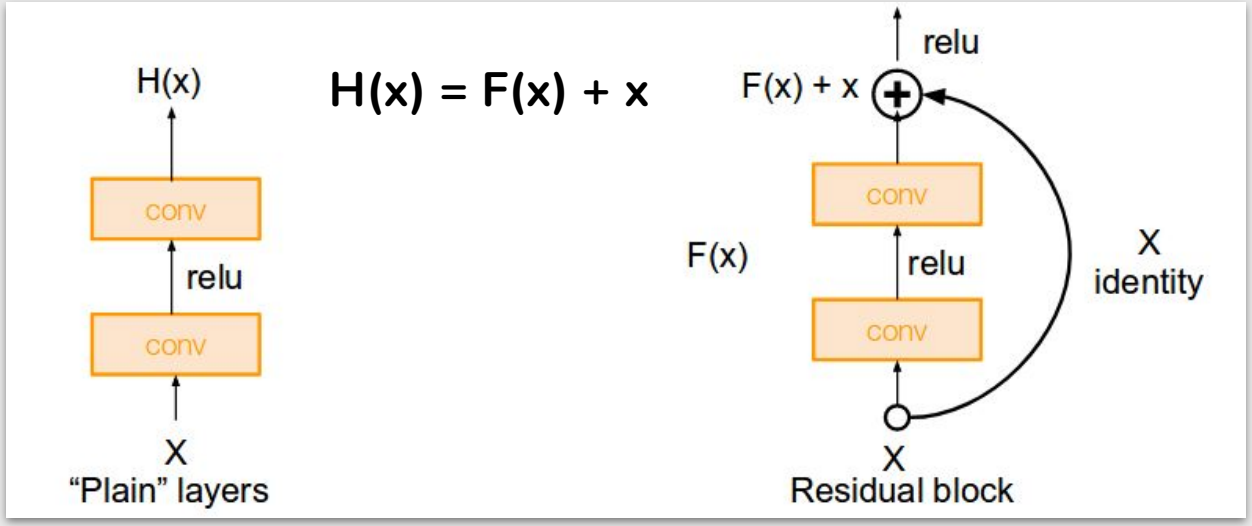
Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)



ResNet [He et al., 2015]

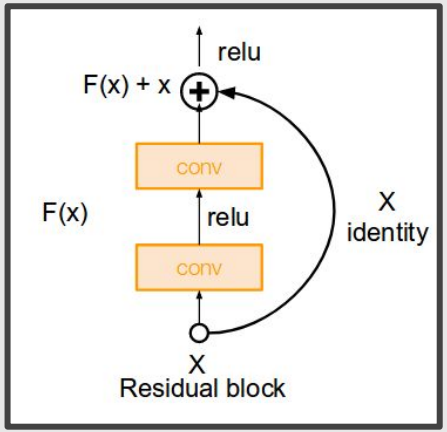
Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

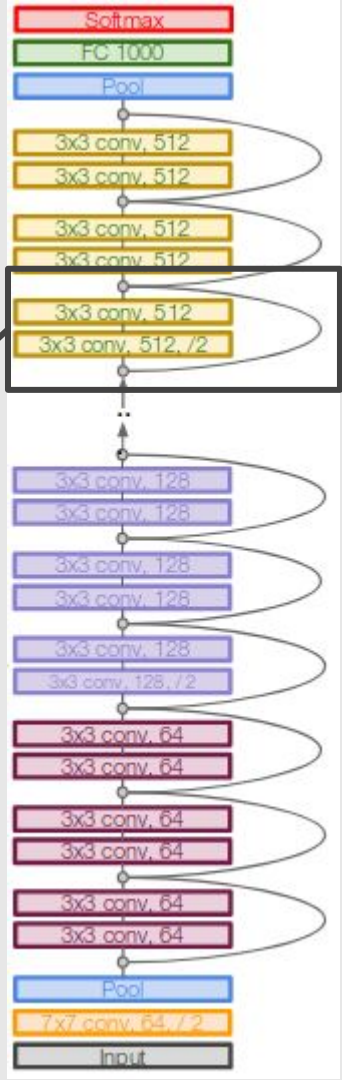


ResNet [He et al., 2015]

Full ResNet architecture:

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning
- No FC layers at the end (only FC 1000 to output classes)





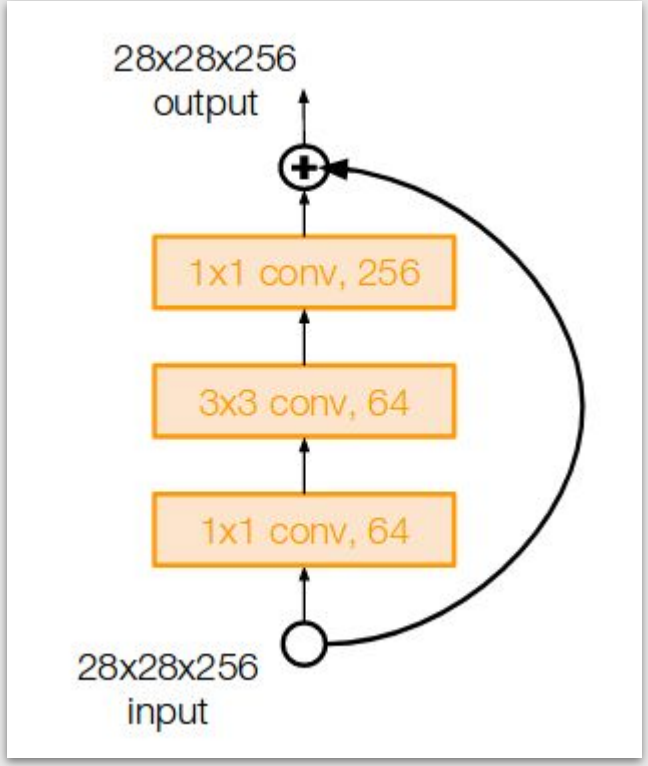
ResNet [He et al., 2015]

For deeper networks (**ResNet-50+**), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)

1x1 conv, 256 filters projects back to 256 feature maps (28x28x256)

3x3 conv operates over only 64 feature maps

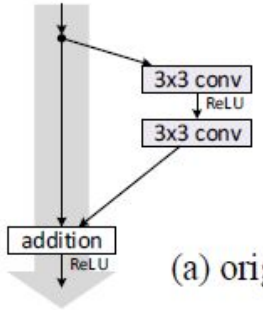
1x1 conv, 64 filters to project to 28x28x64



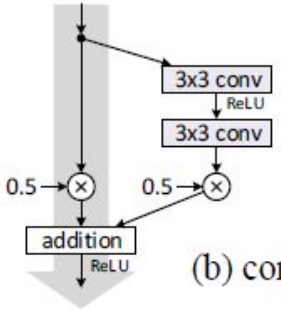
* ResNet-110 on CIFAR-10

$h(x) = x$

error: 6.6%



(a) original



(b) constant scaling

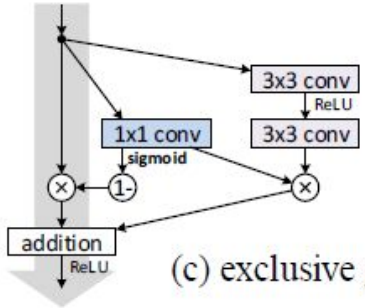
$h(x) = 0.5x$

error: 12.4%

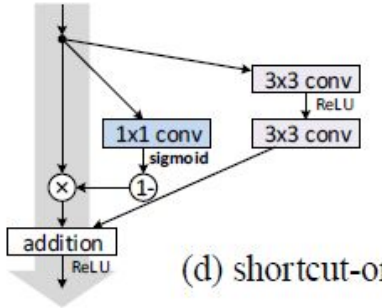
$h(x) = \text{gate} \cdot x$

error: 8.7%

*similar to "Highway Network"



(c) exclusive gating



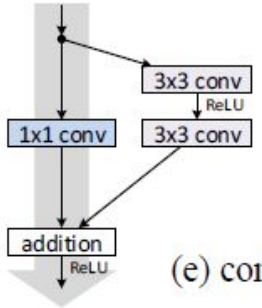
(d) shortcut-only gating

$h(x) = \text{gate} \cdot x$

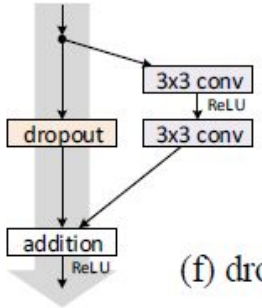
error: 12.9%

$h(x) = \text{conv}(x)$

error: 12.2%



(e) conv shortcut



(f) dropout shortcut

$h(x) = \text{dropout}(x)$

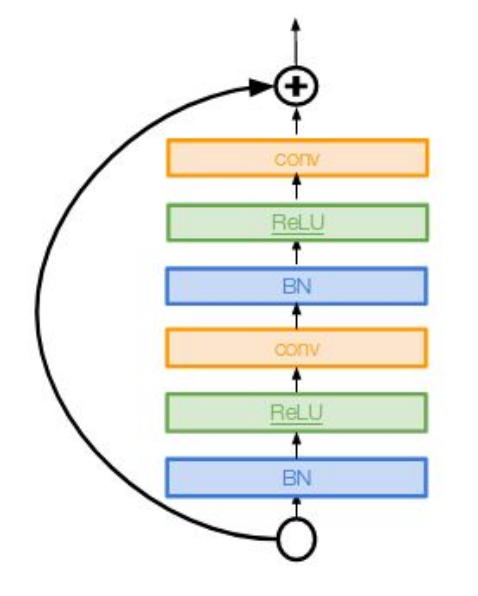
error: > 20%

Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Identity Mappings in Deep Residual Networks". arXiv 2016.

Improving ResNet ...

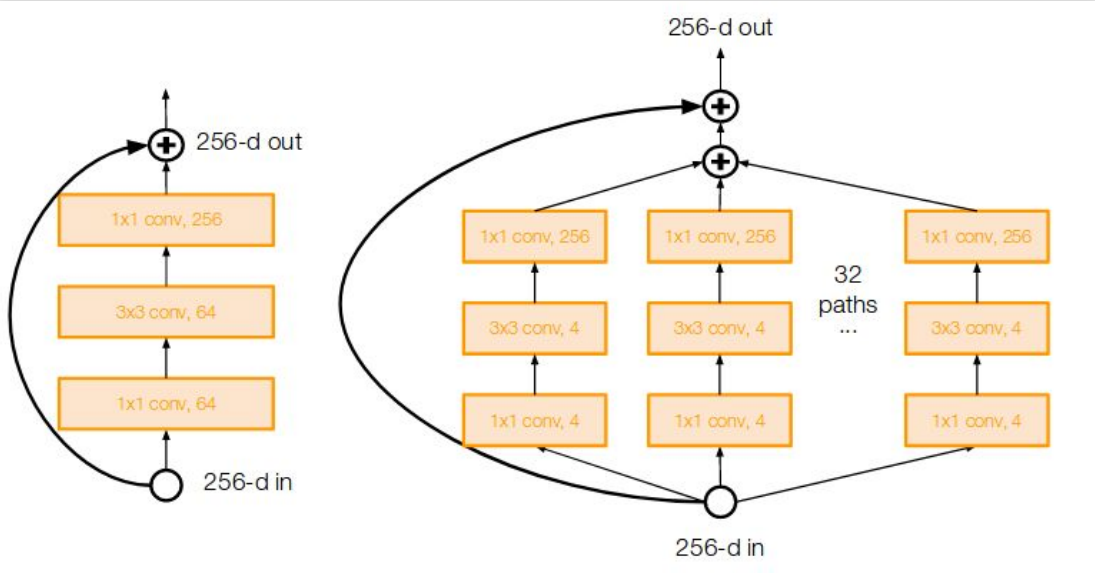
Identity Mappings in Deep Residual Networks [He et al., 2016]

- Creates a more direct path for propagating information throughout network (moves activation to residual mapping pathway)
- Gives better performance



Improving ResNet ...

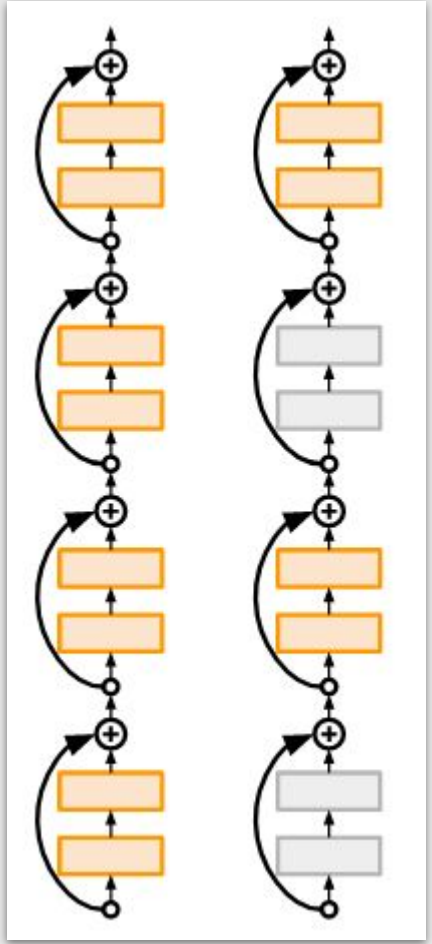
Aggregated Residual Transformations for Deep Neural Networks (**ResNeXt**) [Xie et al., 2016]



Improving ResNet ...

Deep Networks with Stochastic Depth [Huang et al., 2016]

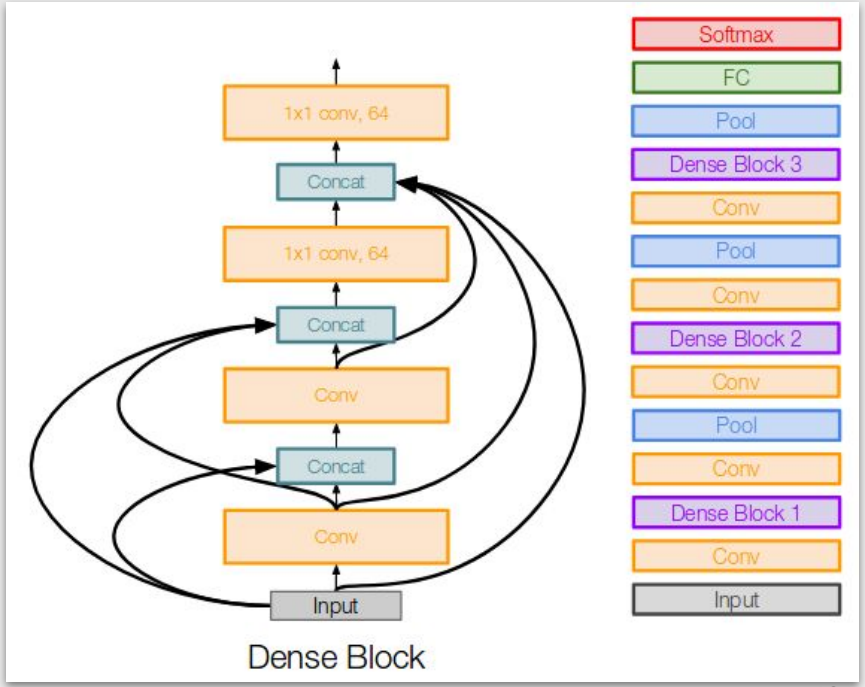
- Motivation: reduce vanishing gradients
- **Randomly drop** a subset of layers during each training pass
- Bypass with identity function

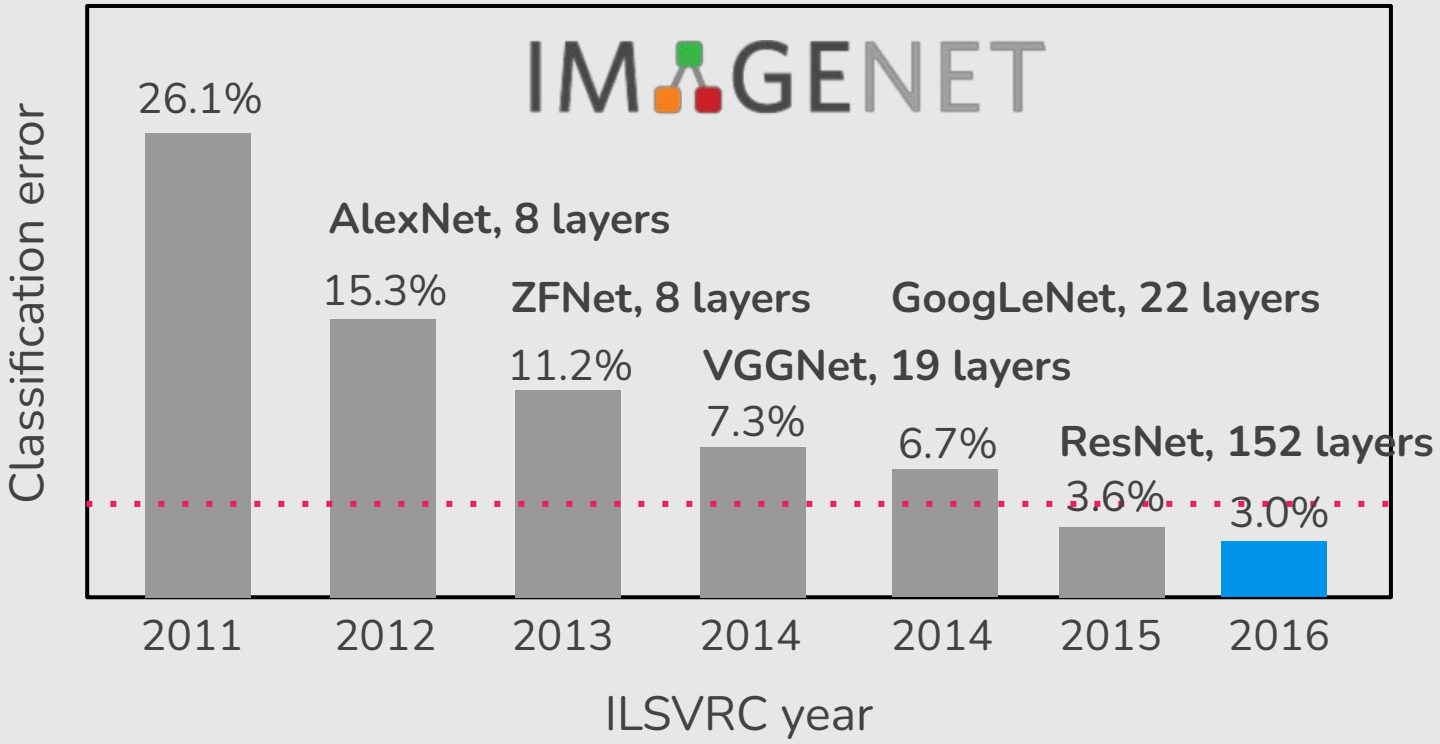


Beyond ResNet ...

Densely Connected Convolutional Networks (**DenseNet**) [Huang et al., 2017]

- Each layer is connected to every other layer in feedforward fashion



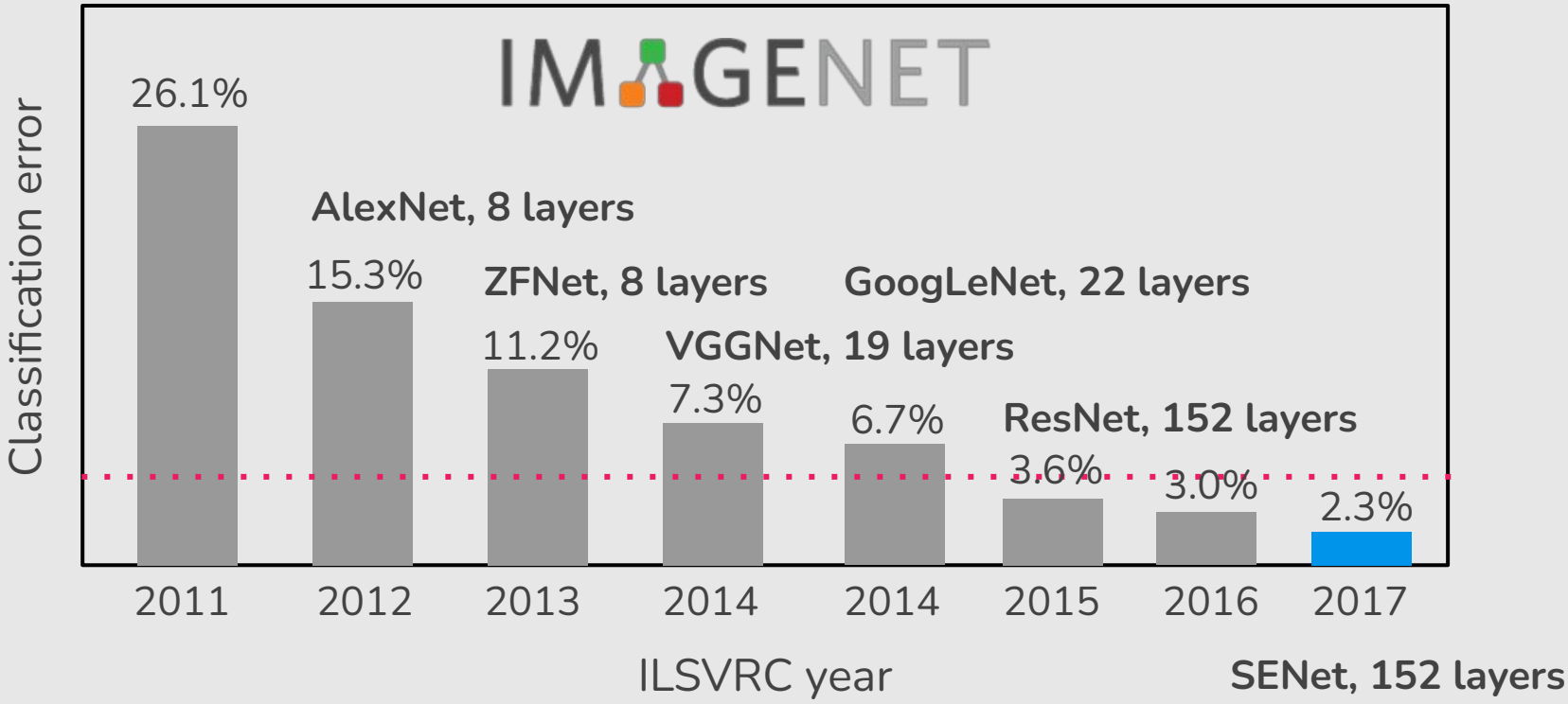


"Good Practices for Deep Feature Fusion", ECCV 2016,
http://image-net.org/challenges/talks/2016/Trimps-Soushen@ILSVRC2016.pdf (Slides only)

Good Practices for Deep Feature Fusion
[Shao et al., 2016]

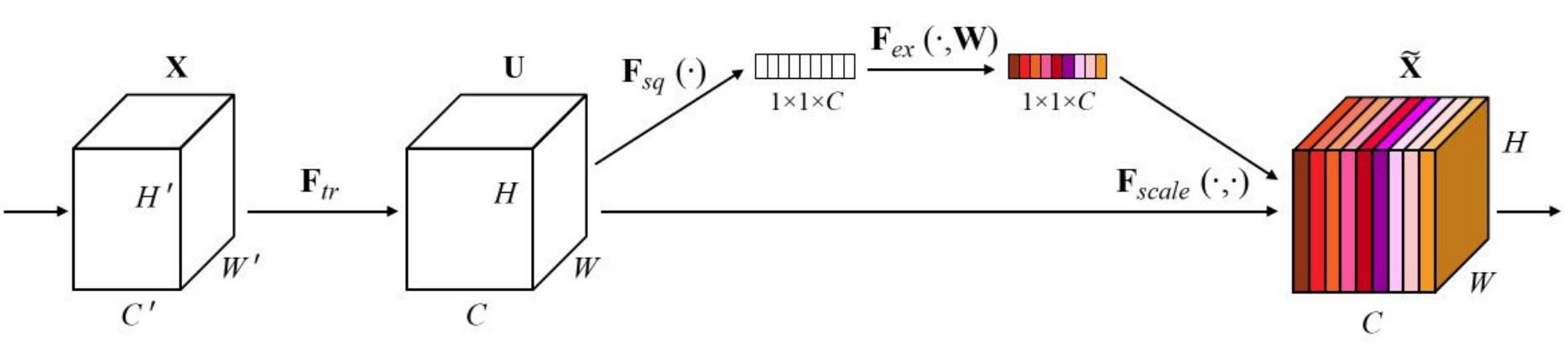
- Training
  - Multi-scale augmentation & large mini-batch size
- Testing
  - Multi-scale & flip & dense fusion

	Error (%)
Inception-v3	4.20
Inception-v4	4.01
Inception-ResNet-v2	3.52
ResNet-200	4.26
Wrm-68-3	4.65
Fusion (Test)	**2.99**

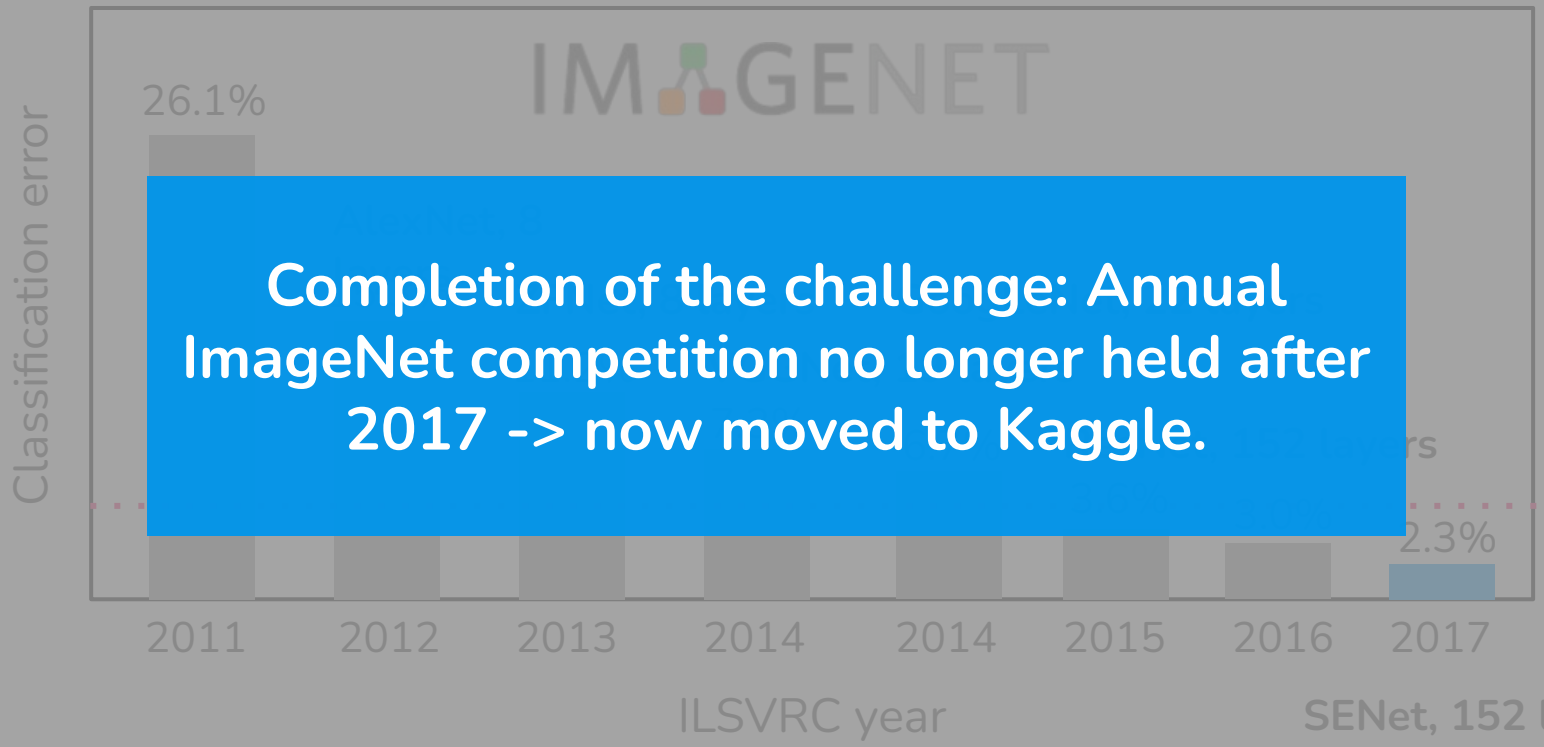


"Squeeze-and-Excitation Networks", CVPR 2018, https://arxiv.org/pdf/1709.01507

SENets [Hu et al. 2017]



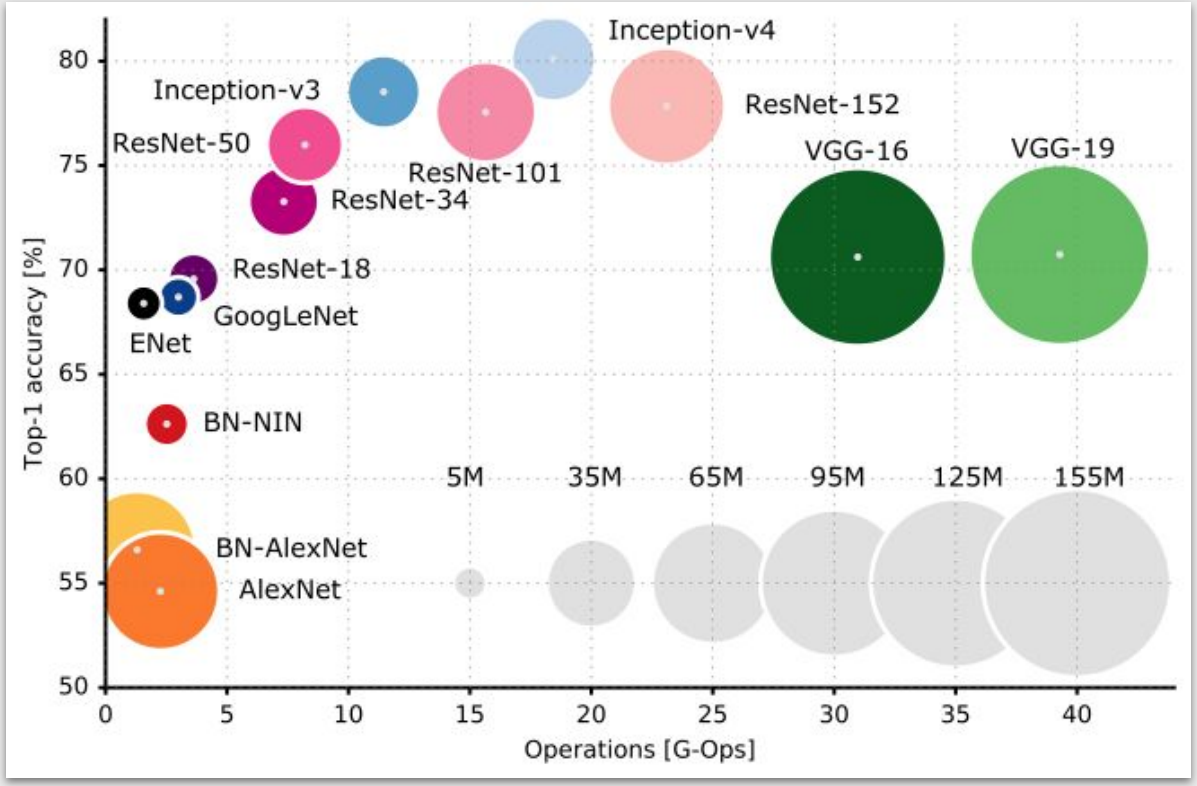
Add a “feature recalibration” module that **learns** to **adaptively reweight feature maps**.



Completion of the challenge: Annual ImageNet competition no longer held after 2017 -> now moved to Kaggle.

"Squeeze-and-Excitation Networks", CVPR 2018, https://arxiv.org/pdf/1709.01507

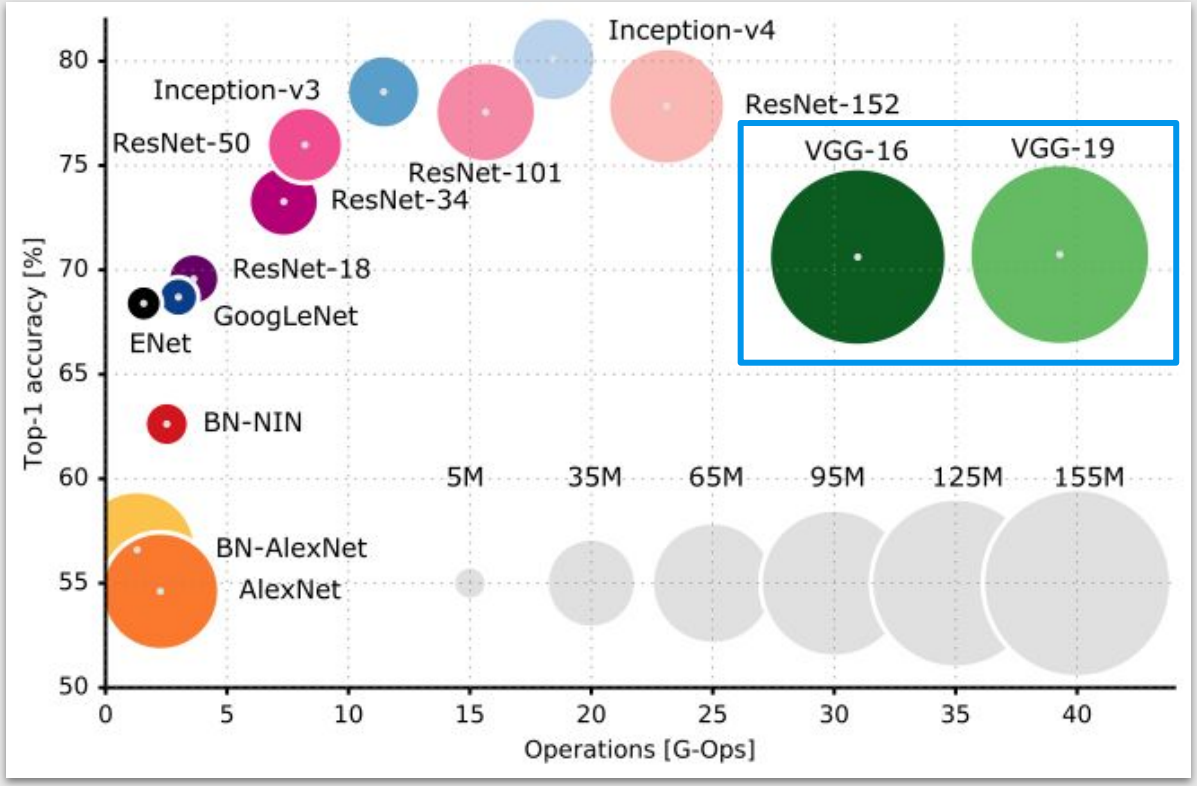
The size of the blobs is proportional to the number of network parameters.



https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba

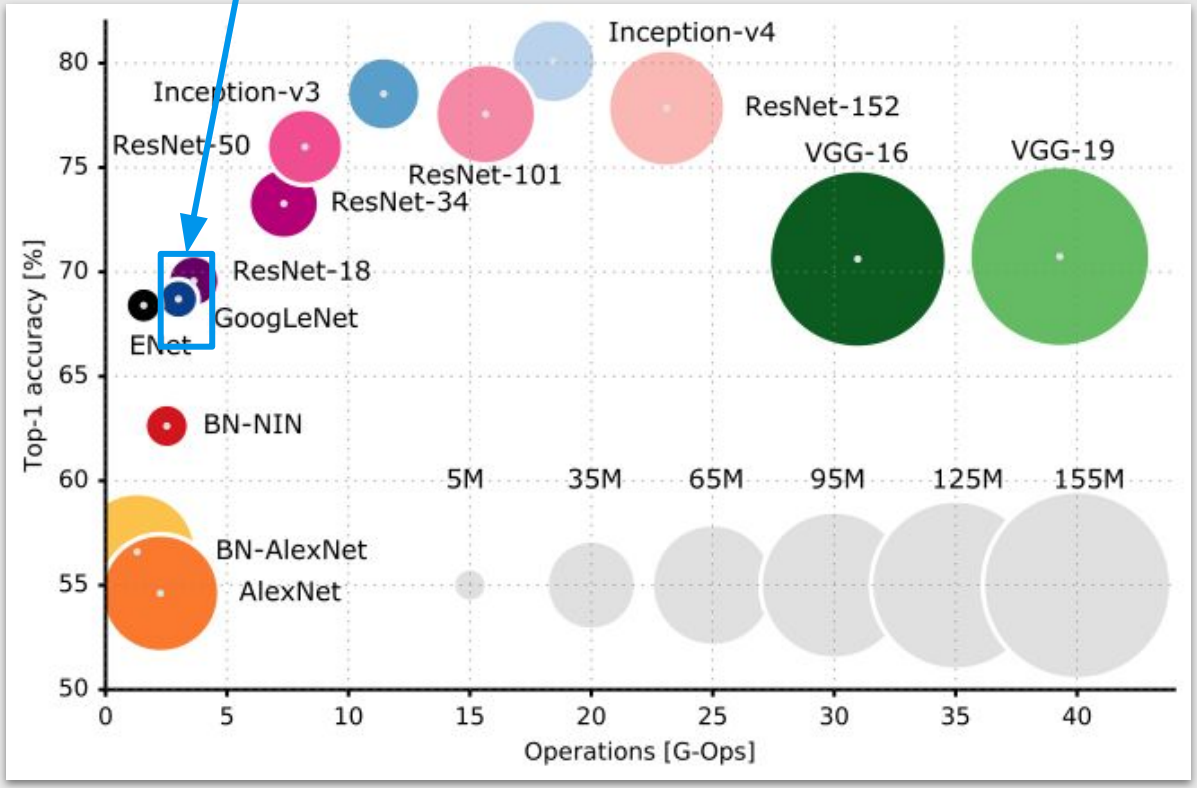
VGG: Highest memory, most operations

The size of the blobs is proportional to the number of network parameters.



https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba

GoogLeNet: most efficient

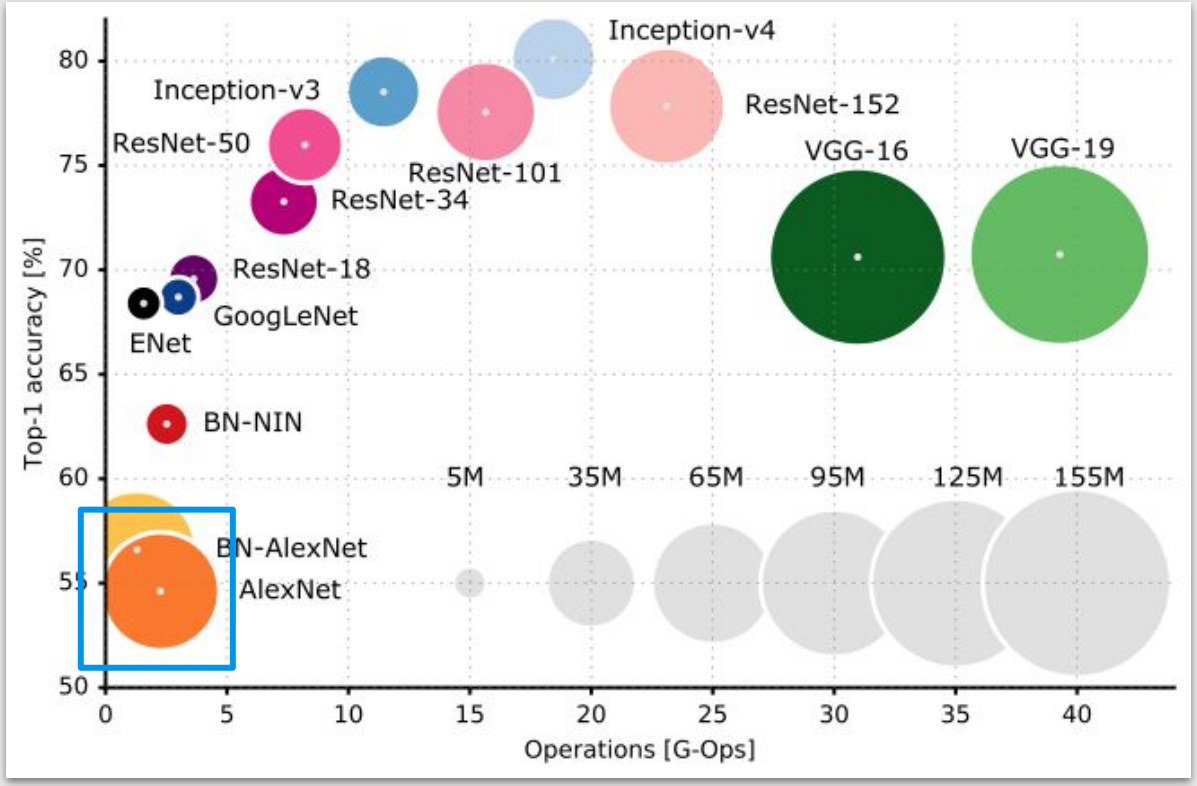


https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba

The size of the blobs is proportional to the number of network parameters.

AlexNet: Smaller compute, still memory heavy, lower accuracy

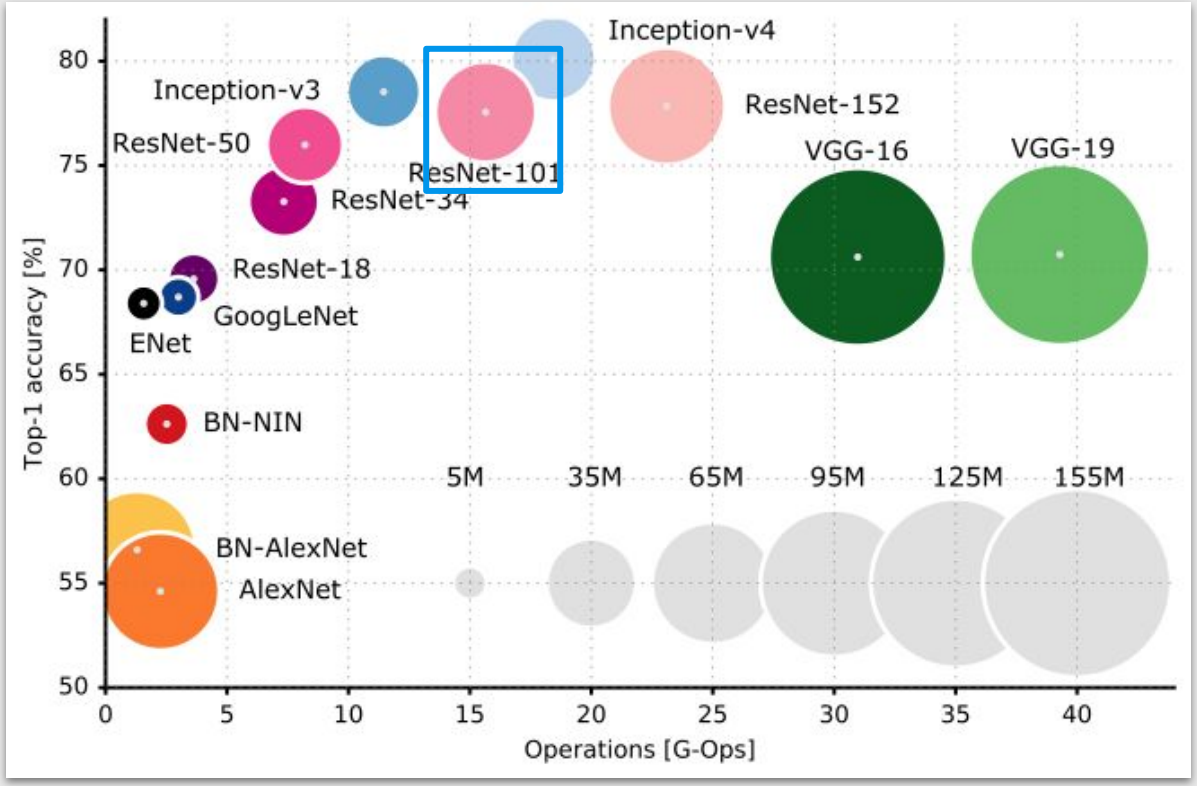
The size of the blobs is proportional to the number of network parameters.



https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba

ResNet: Moderate efficiency depending on model, highest accuracy

The size of the blobs is proportional to the number of network parameters.



https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba

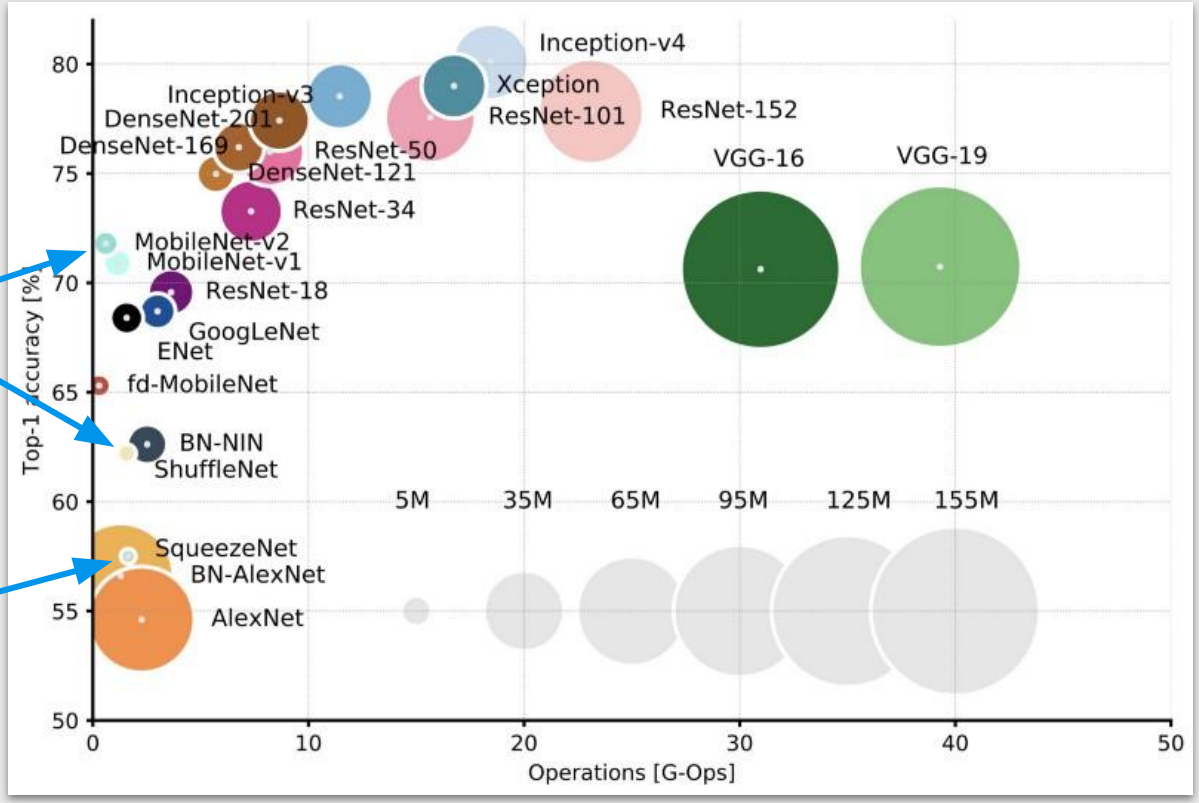
ShuffleNet

MobileNet

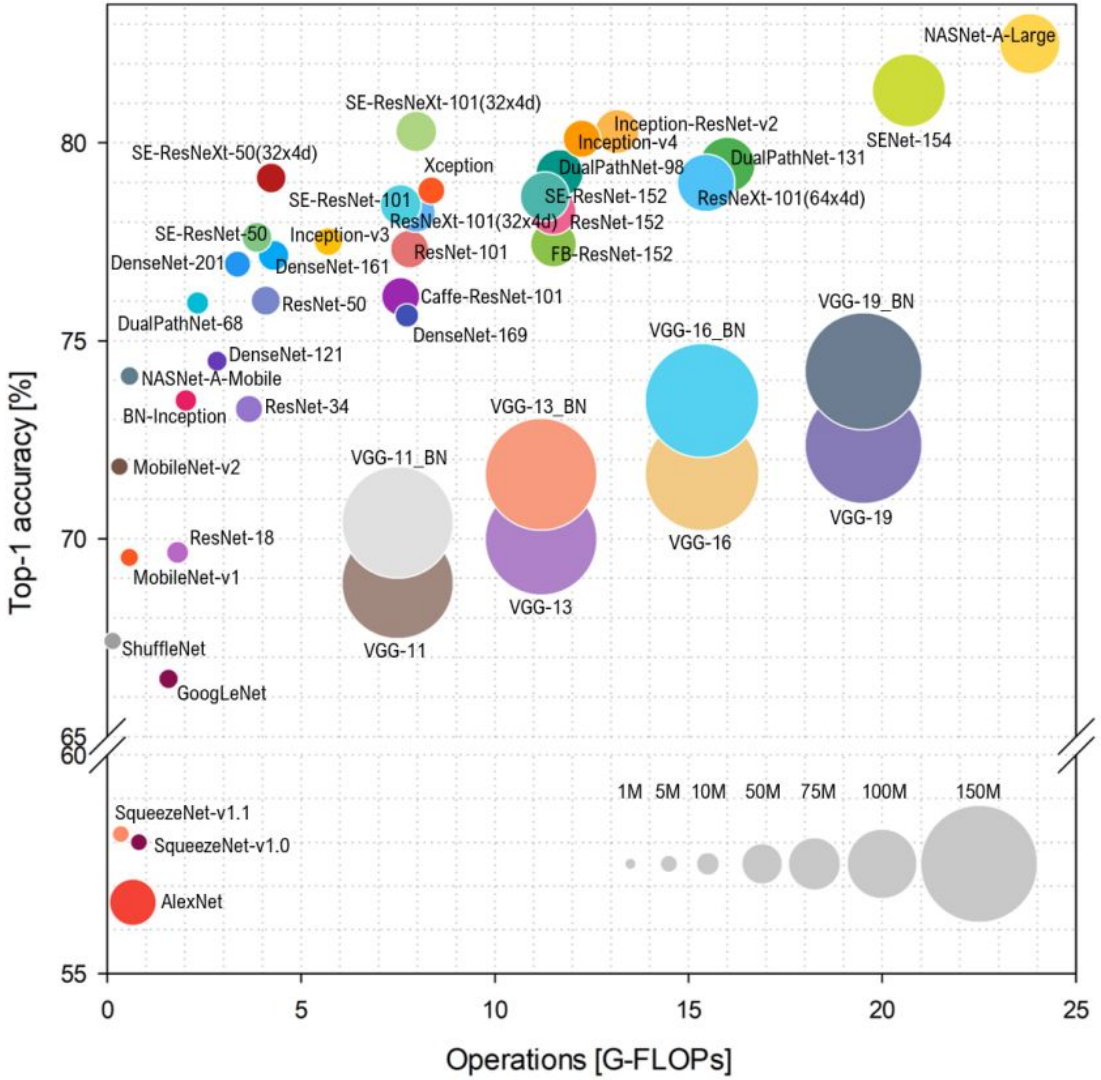
Xception

DenseNet

SqueezeNet



https://medium.com/towards-data-science/neural-network-architectures-156e5bad51ba



“Benchmark Analysis of Representative Deep Neural Network Architectures”, Nov. 2018.
https://doi.org/10.1109/ACCESS.2018.2877890

https://arxiv.org/pdf/1905.11946.pdf, ICML 2019

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Mingxing Tan [1] Quoc V. Le [1]

Abstract

Convolutional Neural Networks (ConvNets) are commonly developed at a fixed resource budget, and then scaled up for better accuracy if more resources are available. In this paper, we systematically study model scaling and identify that carefully balancing network depth, width, and resolution can lead to better performance. Based on this observation, we propose a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective *compound coefficient*. We demonstrate the effectiveness of this method on scaling up MobileNets and ResNet.

To go even further, we use neural architecture search to design a new baseline network

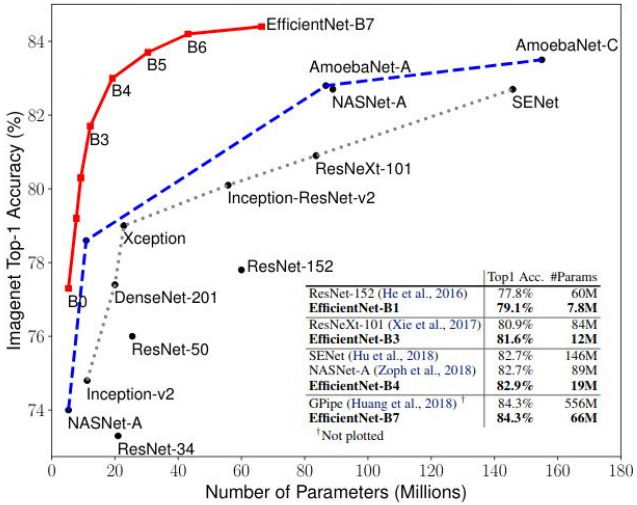
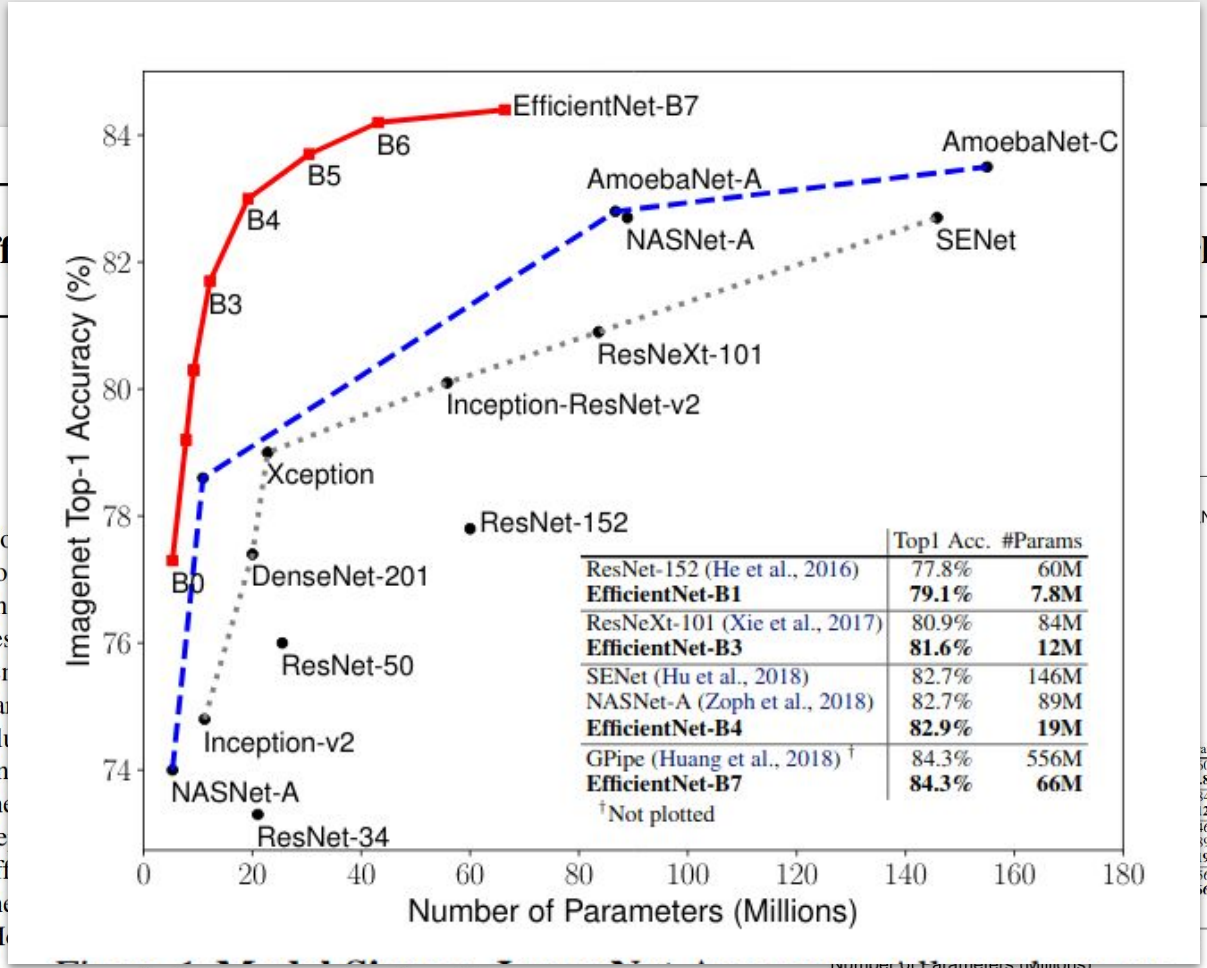


Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

To go even further, we use neural architecture search to design a new baseline network

Figure 1. **Model Size vs. ImageNet Accuracy.** All numbers are

https://keras.io/api/applications/

Keras Applications

Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning.

Weights are downloaded automatically when instantiating a model. They are stored at `~/.keras/models/`.

Upon instantiation, the models will be built according to the image data format set in your Keras configuration file at `~/.keras/keras.json`. For instance, if you have set `image_data_format=channels_last`, then any model loaded from this repository will get built according to the TensorFlow data format convention, "Height-Width-Depth".

Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5

Keras Applications
- Available models
- Usage examples for image classification models
  - Classify ImageNet classes with ResNet50
  - Extract features with VGG16
  - Extract features from an arbitrary intermediate layer with VGG19
  - Fine-tune InceptionV3 on a new set of classes
  - Build InceptionV3 over a custom input tensor

https://keras.io/api/applications/

Code examples
Why choose Keras?
Community & governance
Contributing to Keras
KerasTuner
KerasCV
KerasNLP

ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
InceptionV3	92	77.9%	93.7%	23.9M	189	42.2	6.9
InceptionResNetV2	215	80.3%	95.3%	55.9M	449	130.2	10.0
MobileNet	16	70.4%	89.5%	4.3M	55	22.6	3.4
MobileNetV2	14	71.3%	90.1%	3.5M	105	25.9	3.8
DenseNet121	33	75.0%	92.3%	8.1M	242	77.1	5.4
DenseNet169	57	76.2%	93.2%	14.3M	338	96.4	6.3
DenseNet201	80	77.3%	93.6%	20.2M	402	127.2	6.7
NASNetMobile	23	74.4%	91.9%	5.3M	389	27.0	6.7
NASNetLarge	343	82.5%	96.0%	88.9M	533	344.5	20.0
EfficientNetB0	29	77.1%	93.3%	5.3M	132	46.0	4.9
EfficientNetB1	31	79.1%	94.4%	7.9M	186	60.2	5.6
EfficientNetB2	36	80.1%	94.9%	9.2M	186	80.8	6.5
EfficientNetB3	48	81.6%	95.7%	12.3M	210	140.0	8.8
EfficientNetB4	75	82.9%	96.4%	19.5M	258	308.3	15.1
EfficientNetB5	118	83.6%	96.7%	30.6M	312	579.2	25.3
EfficientNetB6	166	84.0%	96.8%	43.3M	360	958.1	40.4
EfficientNetB7	256	84.3%	97.0%	66.7M	438	1578.9	61.6
EfficientNetV2B0	29	78.7%	94.3%	7.2M	-	-	-
EfficientNetV2B1	34	79.8%	95.0%	8.2M	-	-	-
EfficientNetV2B2	42	80.5%	95.1%	10.2M	-	-	-
EfficientNetV2B3	59	82.0%	95.8%	14.5M	-	-	-
EfficientNetV2S	88	83.9%	96.7%	21.6M	-	-	-
EfficientNetV2M	220	85.3%	97.4%	54.4M	-	-	-
EfficientNetV2L	479	85.7%	97.5%	119.0M	-	-	-
ConvNeXtTiny	109.42	81.3%	-	28.6M	-	-	-

Keras Applications

- Available models
- Usage examples for image classification models
  - Classify ImageNet classes with ResNet50
  - Extract features with VGG16
  - Extract features from an arbitrary intermediate layer with VGG19
  - Fine-tune InceptionV3 on a new set of classes
  - Build InceptionV3 over a custom input tensor

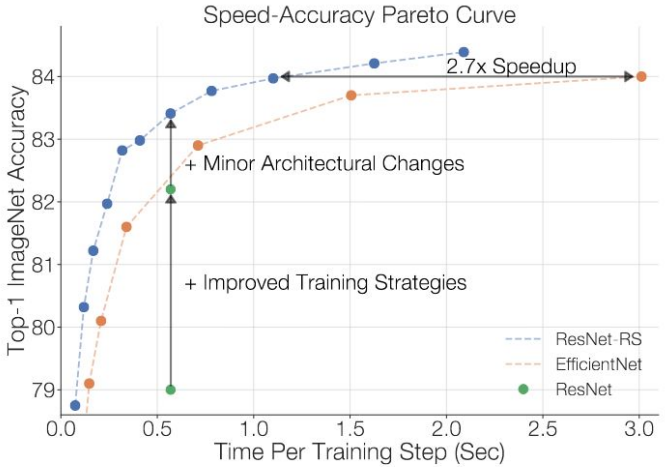
https://arxiv.org/pdf/2103.07579.pdf, NeurIPS 2021

Revisiting ResNets: Improved Training and Scaling Strategies

Irwan Bello [1] William Fedus [1] Xianzhi Du [1] Ekin D. Cubuk [1] Aravind Srinivas [2] Tsung-Yi Lin [1]
Jonathon Shlens [1] Barret Zoph [1]

Abstract

Novel computer vision architectures monopolize the spotlight, but the impact of the model architecture is often conflated with simultaneous changes to training methodology and scaling strategies. Our work revisits the canonical ResNet (He et al., 2015) and studies these three aspects in an effort to disentangle them. Perhaps surprisingly, we find that training and scaling strategies may matter more than architectural changes, and further, that the resulting ResNets match recent state-of-the-art models. We show that the best performing scaling strategy depends on the training regime and offer



https://arxiv.org/pdf/2103.07579.pdf, NeurIPS 2021

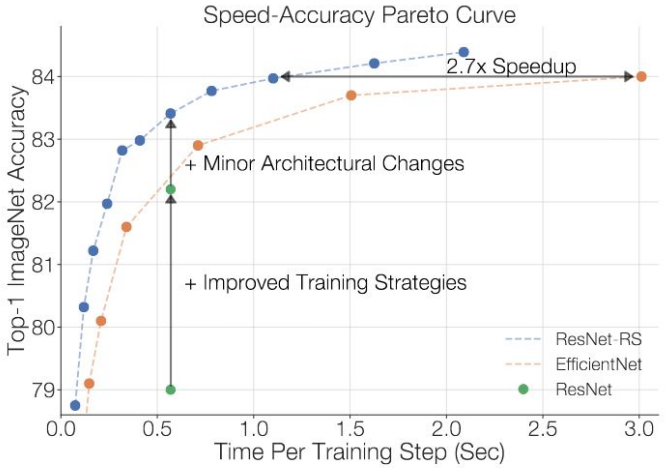
Improvements	Top-1	Δ
ResNet-200	79.0	—
+ Cosine LR Decay	79.3	**+0.3**
+ Increase training epochs	78.8 †	-0.5
+ EMA of weights	79.1	**+0.3**
+ Label Smoothing	80.4	**+1.3**
+ Stochastic Depth	80.6	**+0.2**
+ RandAugment	81.0	**+0.4**
+ Dropout on FC	80.7 ‡	-0.3
+ Decrease weight decay	82.2	**+1.5**
+ Squeeze-and-Excitation	82.9	**+0.7**
+ ResNet-D	83.4	**+0.5**

Table 1. **Additive study of the ResNet-RS training recipe.** The colors refer to **Training Methods**, **Regularization Methods** and **Architecture Improvements**. The baseline ResNet-200 was trained for the standard 90 epochs using a stepwise learning rate decay schedule. The image resolution is 256×256. All

scaling strategies may matter more than architectural changes, and further, that the resulting ResNets match recent state-of-the-art models. We show that the best performing scaling strategy depends on the training regime and offer

raining and Scaling Strategies

D. Cubuk [1] Aravind Srinivas [2] Tsung-Yi Lin [1]
Barret Zoph [1]



Summary: CNN Architectures

- Many popular architectures available in **model zoos**
- **ResNet and EfficientNet** currently good defaults to use
- Networks have gotten increasingly deep over time
- Many other aspects of network architectures are also **continuously being investigated and improved**
- Even more recent trend towards **meta-learning**

References

Machine/Deep Learning Books

- “Hands-On Machine Learning with Scikit-Learn and TensorFlow” (2016), **Convolutional Neural Networks,** Chap. 13
- “Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow” (2019), **Deep Computer Vision Using Convolutional Neural Networks**, Chap. 14

Deep Learning Courses

- “Convolutional Neural Networks for Visual Recognition” (2017), **CNN Architectures**, Fei-Fei Li & others https://youtu.be/DAOcjicFr1Y (80 min)
- Deeplearning.ai (2017), “Convolutional Neural Networks” C4W2L01-L08 (3-18min) https://www.youtube.com/playlist?list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF

That's all!

@sandraavilabr