Convolutional Neural Networks

IE643 - Lectures 15 & 16

Sep 30 & Oct 4, 2022.

1. Popular CNN Architectures
   - DenseNet
   - EfficientNet
   - Region CNN
   - 3D CNN
   - U-Net
2. Is CNN a black-box?

CNN - DenseNet

Densely Connected Convolutional Networks

Gao Huang*
Cornell University
gh349@cornell.edu

Zhuang Liu*
Tsinghua University
liuzhuang13@mails.tsinghua.edu.cn

Laurens van der Maaten
Facebook AI Research
lvdmaaten@fb.com

Kilian Q. Weinberger
Cornell University
kqw4@cornell.edu

CNN - DenseNet

- **Aim**: To facilitate flow of gradients from last layers to the initial layers **avoiding vanishing gradient** issue.
- A densenet block is designed which can facilitate the flow of gradients.

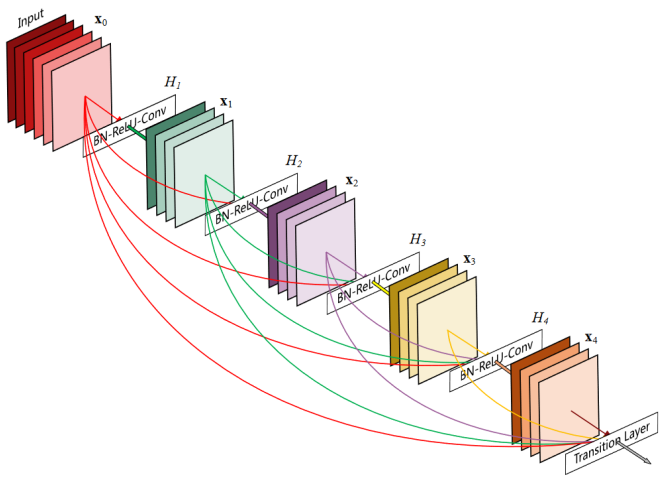


Figure : A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

- In each densenet block of L layers, the ℓ-th layer $(0 \leq \ell \leq L-1)$ receives feature maps from **all the previous** $\ell - 1$ **layers**.
- Hence there would be $\frac{L(L+1)}{2}$ feature maps in the densenet block.

CNN - DenseNet

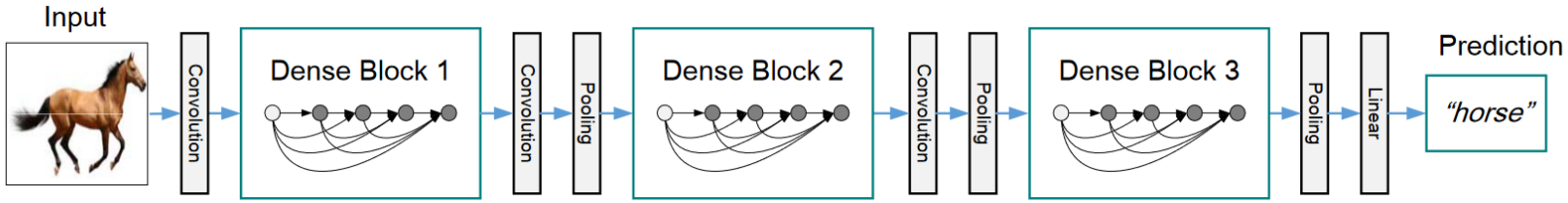


Figure : A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

- At ℓ-th layer the operation would be of the form:

$$M_\ell = H_\ell[M_0; M_1; \ldots; M_{\ell-1}]$$

where $[M_0; M_1; \ldots; M_{\ell-1}]$ denotes concatenation of feature maps $M_0, M_1, \ldots, M_{\ell-1}$ at corresponding layers.

- H_ℓ is composition of Batch normalization, ReLU and 3×3 convolution.

$$H_\ell(M) = \text{Conv}_{3\times3}(ReLU(BN(M))), \ell \in \{0, 1, \ldots, L-1\}.$$

CNN - DenseNet

Batch Norm Operation

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Ref: S. Ioffe and C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. http://arxiv.org/abs/1502.03167

CNN - DenseNet

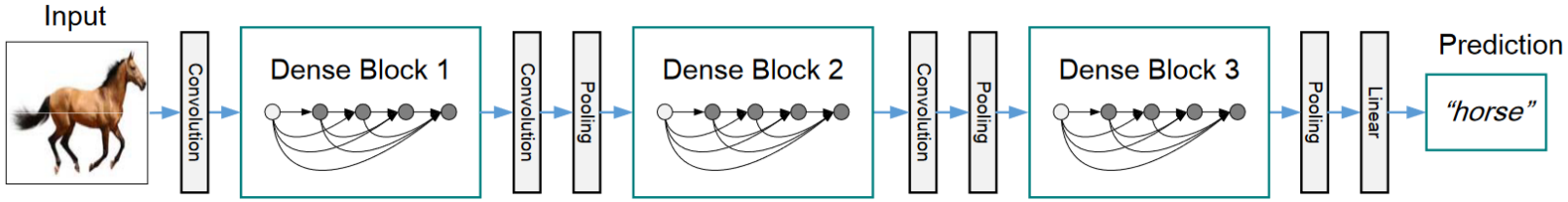


Figure : A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

- The densely connected layers are grouped to form **dense blocks**.
- The convolution and pooling layers between the dense blocks are called **transition layers**.
- At translation layers, the following operations are done:

$$H_{\text{trans}}(M) = \text{AvgPool}_{2\times 2}(\text{Conv}_{1\times 1}(BN(M)))$$

CNN - DenseNet

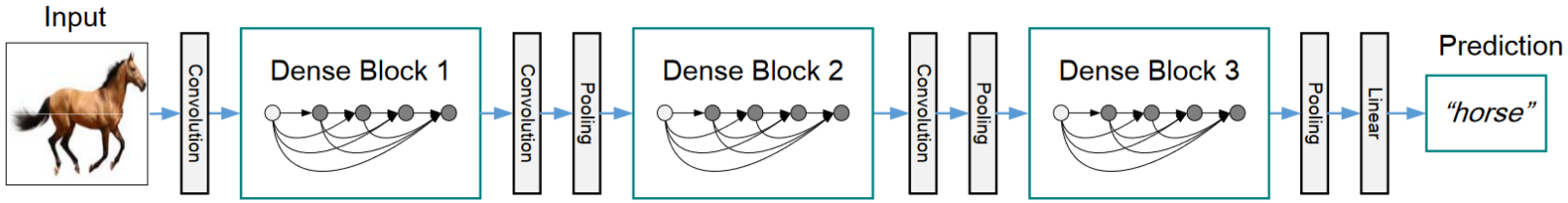


Figure : A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

- If each layer in dense block outputs k feature maps, then the input to the ℓ-th layer in the dense block is $k_0 + k * (\ell - 1)$.
- k_0 is the number of channels at the input layer.
- Thus the number of feature maps at each layer in a dense block is controlled by a factor k called the growth rate.
- k can typically be a small number. (e.g. $k = 12$)

CNN - DenseNet

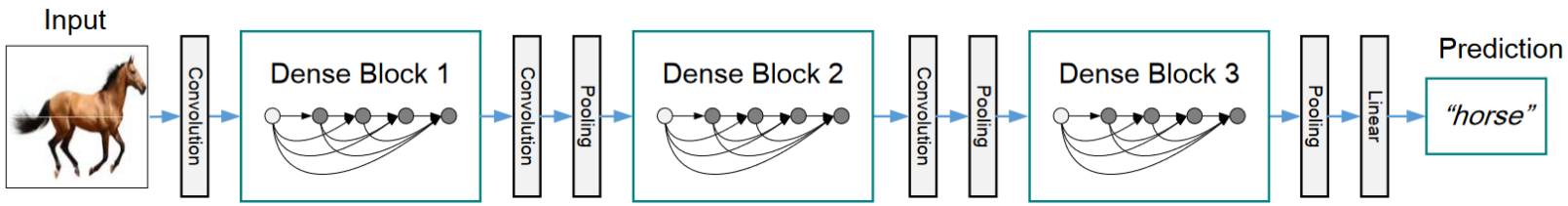


Figure : A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

Other heuristics:

- In DenseNet-B, H_ℓ a bottleneck layer of 1×1 convolution precedes the 3×3 convolution:

$$H_\ell(M) = \text{Conv}_{3\times3}(\text{Conv}_{1\times1}(ReLU(BN(M))))$$

CNN - DenseNet

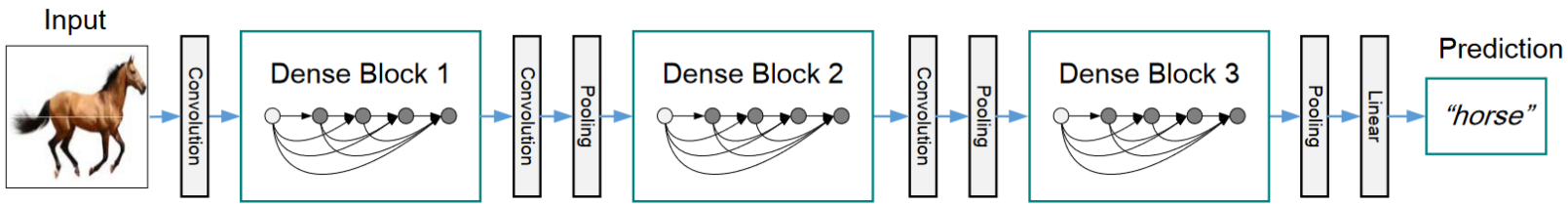


Figure : A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

Other heuristics:

- Compression introduced at transition layer where only $\lfloor \theta m \rfloor, \theta \in (0, 1]$ to pass on only a fraction of total m feature maps produced at the dense block.

CNN - DenseNet

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Table : DenseNet architectures for ImageNet. The growth rate for all the networks is $k = 32$. Note that each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv.

CNN - DenseNet

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [32]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [34]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [42]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

Table : Error rates (%) on CIFAR and SVHN datasets. k denotes network's growth rate. Results that surpass all competing methods are **bold** and the overall best results are **blue**. "+" indicates standard data augmentation (translation and/or mirroring). $*$ indicates results run by ourselves. All the results of DenseNets without data augmentation (C10, C100, SVHN) are obtained using Dropout. DenseNets achieve lower error rates while using fewer parameters than ResNet. Without data augmentation, DenseNet performs better by a large margin.

CNN - EfficientNet

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Mingxing Tan Quoc V. Le

CNN - EfficientNet

- A layer i in a CNN can be written as:

$$Y_i = F_i(X_{i_{\langle H_i, W_i, C_i \rangle}})$$

  where F_i is the operation on an input feature map X_i of dimensions $H_i \times W_i \times C_i$ to get the output feature map Y_i.
- **Note:** X_i and Y_i are tensors.
- Hence a CNN $\mathcal{N}$ can be written as:

$$\mathcal{N} = F_k(F_{k-1}(\ldots F_2(F_1(X_1))\ldots)).$$

CNN - EfficientNet

- A layer i in a CNN can be written as:

$$Y_i = F_i(X_{i_{\langle H_i, W_i, C_i \rangle}})$$

  where F_i is the operation on an input feature map X_i of dimensions $H_i \times W_i \times C_i$ to get the output feature map Y_i.

- (H_i, W_i) is called **resolution**.
- C_i (number of channels) is called **width**.
- If an operation F_i is possibly repeated L_i times, L_i is called **network length** or **depth**.

CNN - EfficientNet

We will use an equivalent notation to write the CNN $\mathcal{N}$:

$$\begin{aligned}\mathcal{N} &= F^{L_k}_{k_{\langle H_k, W_k, C_k \rangle}} \odot F^{L_{k-1}}_{k-1_{\langle H_{k-1}, W_{k-1}, C_{k-1} \rangle}} \odot \ldots \odot F^{L_1}_{1_{\langle H_1, W_1, C_1 \rangle}} \\ &= \bigodot_{j=1,2,\ldots,k} F^{L_j}_{j_{\langle H_j, W_j, C_j \rangle}}.\end{aligned}$$

CNN - EfficientNet

Optimization problem:

$$\max_{d,w,r} \quad Accuracy(\mathcal{N}(d,w,r))$$

$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}{}_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}$$

$$\text{Memory}(\mathcal{N}) \leq \text{target_memory}$$

$$\text{FLOPS}(\mathcal{N}) \leq \text{target_flops}$$

CNN - EfficientNet

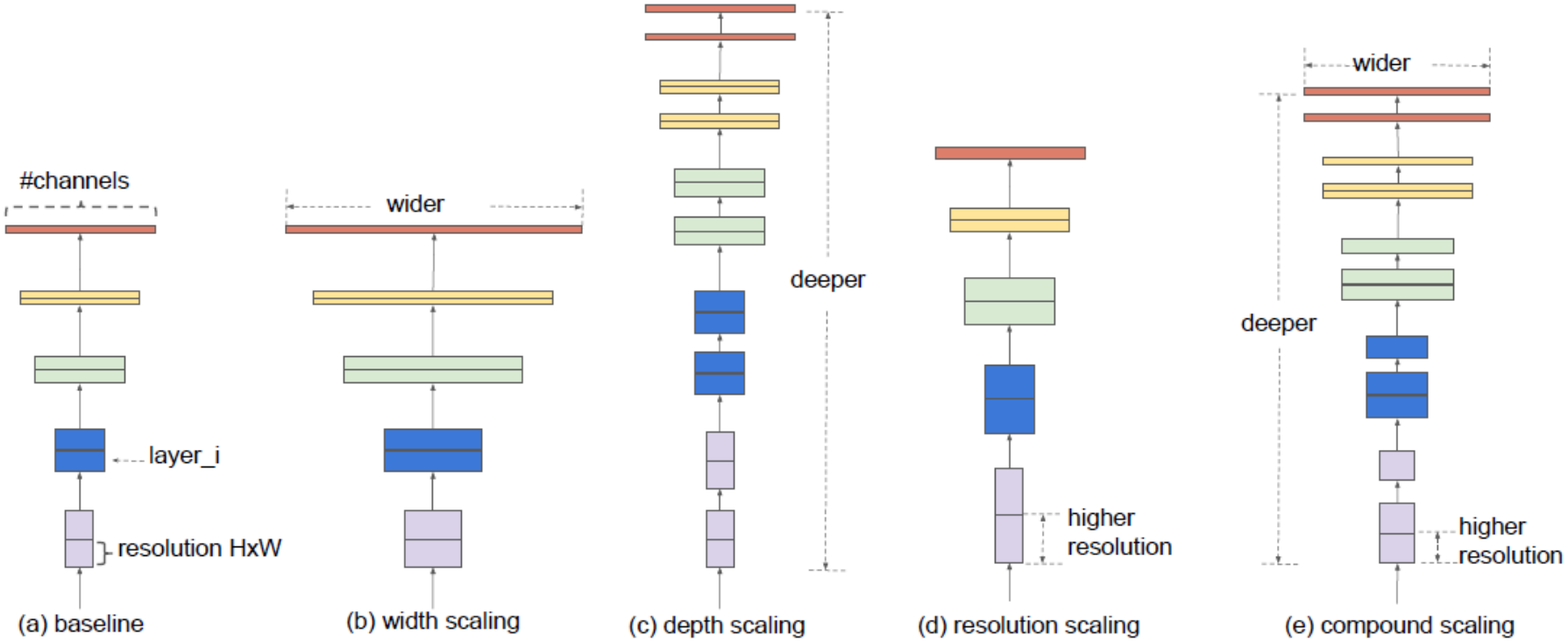


Figure . **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

CNN - EfficientNet

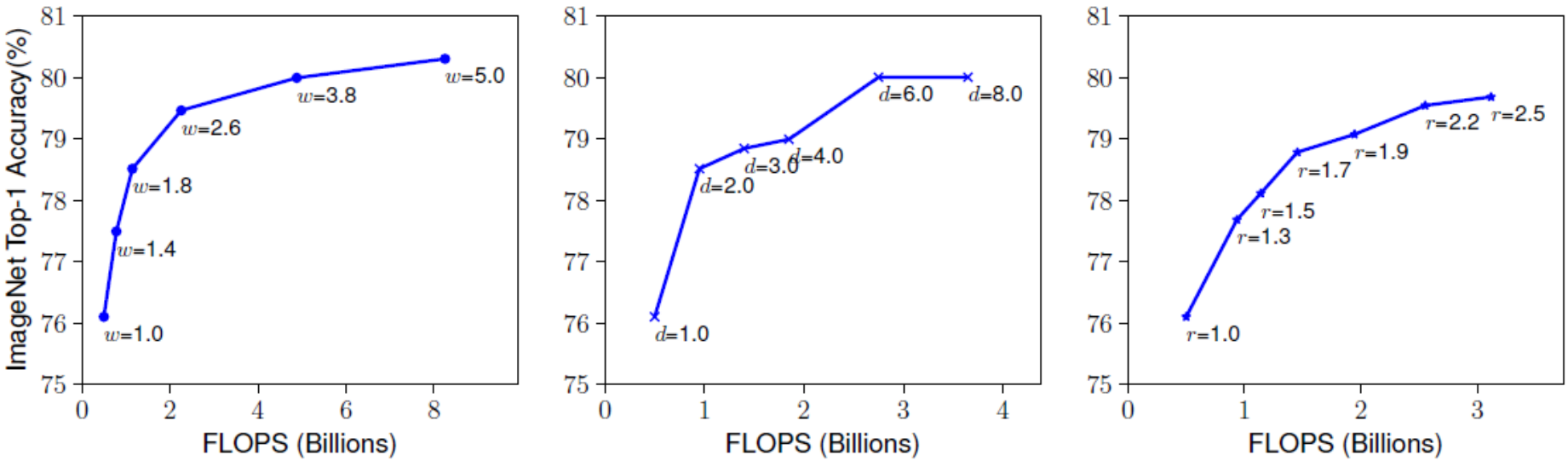


Figure . **Scaling Up a Baseline Model with Different Network Width (w), Depth (d), and Resolution (r) Coefficients.** Bigger networks with larger width, depth, or resolution tend to achieve higher accuracy, but the accuracy gain quickly saturate after reaching 80%, demonstrating the limitation of single dimension scaling.

CNN - EfficientNet

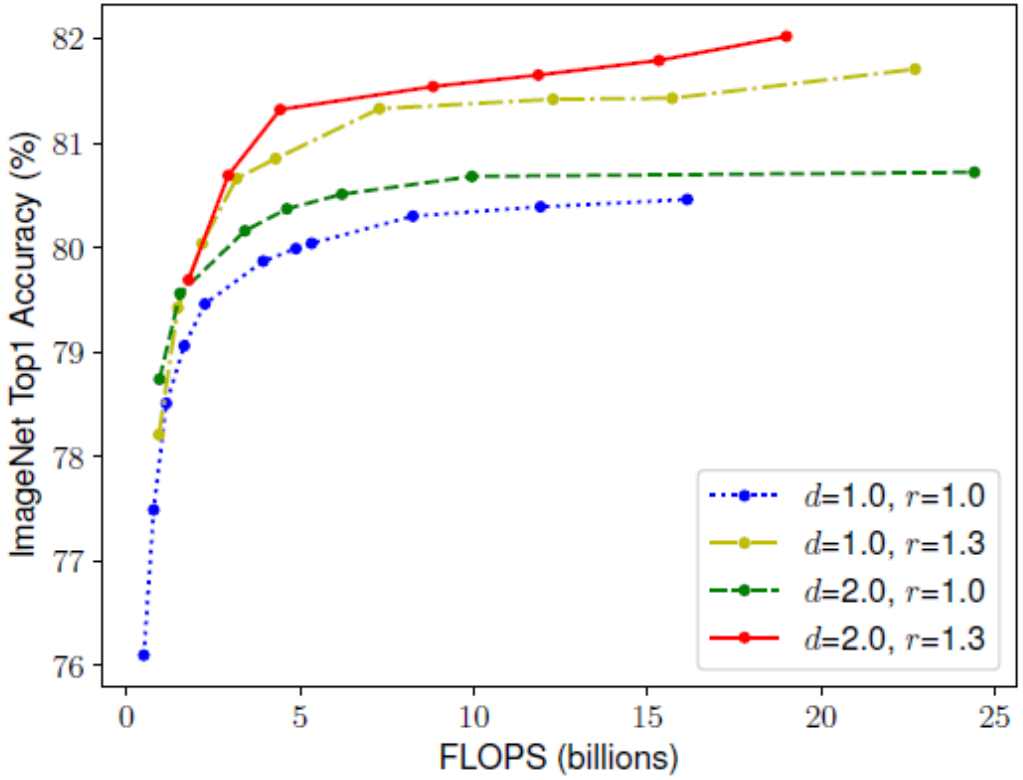


Figure. **Scaling Network Width for Different Baseline Networks**. Each dot in a line denotes a model with different width coefficient (w). The first baseline network (d=1.0, r=1.0) has 18 convolutional layers with resolution 224x224, while the last baseline (d=2.0, r=1.3) has 36 layers with resolution 299x299.

CNN - EfficientNet

- **Optimization problem:**

$$\max_{d,w,r} \quad Accuracy(\mathcal{N}(d,w,r))$$

$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}{}_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}$$

$$\text{Memory}(\mathcal{N}) \leq \text{target_memory}$$

$$\text{FLOPS}(\mathcal{N}) \leq \text{target_flops}$$

- **Parameter selection:**

$$\text{depth: } d = \alpha^\phi$$

$$\text{width: } w = \beta^\phi$$

$$\text{resolution: } r = \gamma^\phi$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

CNN - EfficientNet

Parameter selection procedure:

- STEP 1: we first fix $\phi = 1$, assuming twice more resources available, and do a small grid search of α, β, γ In particular, we find the best values for EfficientNet-B0 are $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$, under constraint of $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$.
- STEP 2: we then fix α, β, γ as constants and scale up baseline network with different ϕ to obtain EfficientNet-B1 to B7

CNN - EfficientNet

Table . **EfficientNet Performance Results on ImageNet** (Russakovsky et al., 2015). All EfficientNet models are scaled from our baseline EfficientNet-B0 using different compound coefficient ϕ . ConvNets with similar top-1/top-5 accuracy are grouped together for efficiency comparison. Our scaled EfficientNet models consistently reduce parameters and FLOPS by an order of magnitude (up to 8.4x parameter reduction and up to 16x FLOPS reduction) than existing ConvNets.

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPs	Ratio-to-EfficientNet
EfficientNet-B0	**77.1%**	**93.3%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**79.1%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**80.1%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.6%**	**95.7%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.9%**	**96.4%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.6%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.8%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.3%**	**97.0%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

R-CNN (Fast version)

Fast R-CNN

Ross Girshick
Microsoft Research

R-CNN (Fast version)

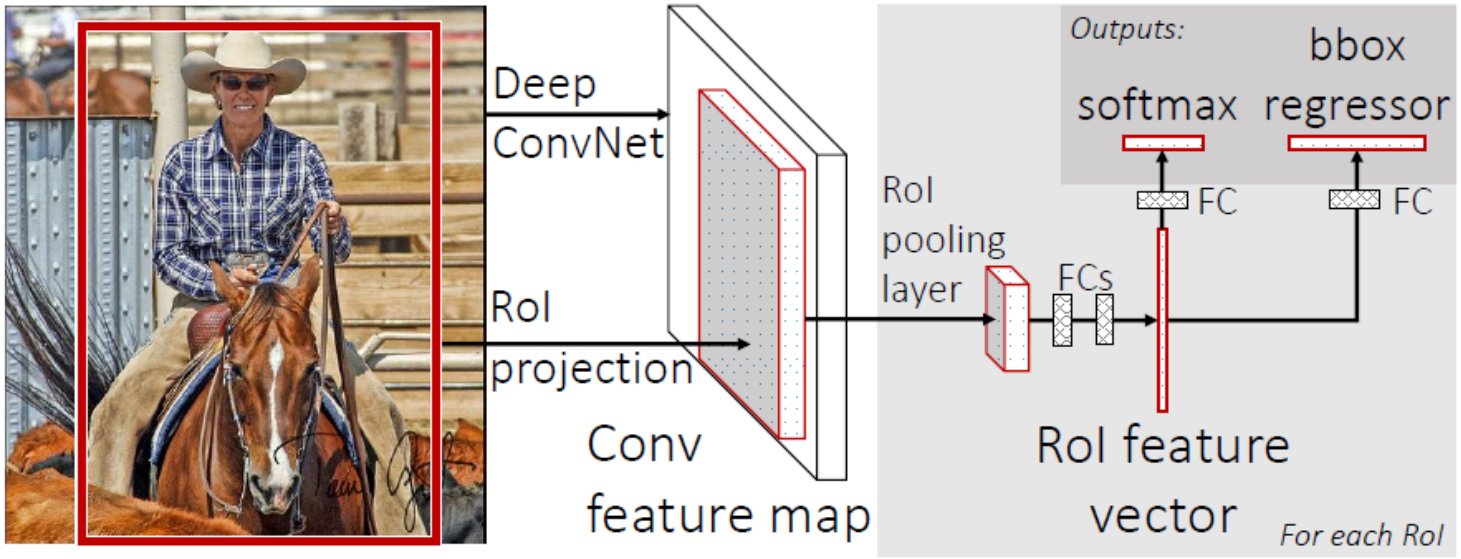


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

R-CNN (Fast version)

- The whole image is first processed by deep convolution networks to produce a conv feature map.
- For each Region of Interest (RoI) proposal, a RoI pooling layer extracts a fixed length feature vector from the feature map.
- Each such feature vector is then fed into sequence of fully connected layers with two output layers.
- The two output layers correspond to object detection and region of interest prediction

R-CNN (Fast version)

- RoI is quantified using (r, c, h, w) where (r, c) represent the top left corner and h, w represent the height and width of regions
- Conv feature map can be constructed from any pre-trained network like Alex Net, VGG Net, etc.

R-CNN (Fast version)

- Two Output Layers:
  - Predict the object class
  - Predict the RoI box
- Loss functions?

R-CNN (Fast version)

- Loss functions
  - $L_{objclass} = L(O_p, O_t)$
  - O_p, O_t represent the predicted class and true class
  - Cross-entropy used for $L_{onjclass}$
  - $L_{RoIBox} = L((r_p, c_p, h_p, w_p), (r_t, c_t, h_t, w_t))$
  - (r_p, c_p, h_p, w_p), (r_t, c_t, h_t, w_t) represent predicted and true RoI parameters.
  - Smooth L1 loss used for L_{RoIBox}

$$L_{RoIBox} = L1_{sm}(r_p - r_t) + L1_{sm}(c_p - c_t) + L1_{sm}(h_p - h_t) + L1_{sm}(w_p - w_t)$$

$$L1_{sm}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{else.} \end{cases}$$

- Total Loss: $L = L_{objclass} + \lambda L_{RoIBox}$
- Guess on λ value used?

R-CNN (Fast version)

- Mini-batch SGD used for training
- Testing: Multiple RoI should be sampled and the network is used for predicting the confidence scores and RoI predictions relative to each sampled RoI.

R-CNN (Fast version)

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11]†	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	07	73.4	77.0	63.4	45.4	**44.6**	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	**35.6**	66.8	67.2	70.4	**71.1**	66.0
FRCN [ours]	07	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
FRCN [ours]	07 \ diff	74.6	**79.0**	68.6	57.0	39.3	79.5	**78.6**	81.9	**48.0**	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	07+12	**77.0**	78.1	**69.3**	**59.4**	38.3	**81.6**	**78.6**	**86.7**	42.8	**78.8**	**68.9**	**84.7**	**82.0**	**76.6**	**69.9**	31.8	**70.1**	**74.8**	**80.4**	70.4	**70.0**

Table 1. **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07 \ diff**: **07** without "difficult" examples, **07+12**: union of **07** and VOC12 trainval. †SPPnet results were prepared by the authors of [11].

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

Table 2. **VOC 2010 test** detection average precision (%). BabyLearning uses a network based on [17]. All other methods use VGG16. Training set key: **12**: VOC12 trainval, **Prop.**: proprietary dataset, **12+seg**: **12** with segmentation annotations, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval.

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	**43.0**	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	**38.6**	**68.3**	58.0	68.7	63.3	63.8
R-CNN BB [10]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours]	07++12	**82.3**	**78.4**	**70.8**	**52.3**	38.7	**77.8**	**71.6**	**89.3**	**44.2**	**73.0**	**55.0**	**87.5**	**80.5**	**80.8**	**72.0**	35.1	**68.3**	**65.7**	**80.4**	**64.2**	**68.4**

Table 3. **VOC 2012 test** detection average precision (%). BabyLearning and NUS_NIN_c2000 use networks based on [17]. All other methods use VGG16. Training set key: see Table 2, **Unk.**: unknown.

R-CNN (Faster version)

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren* Kaiming He Ross Girshick Jian Sun
Microsoft Research
{v-shren, kahe, rbg, jiansun}@microsoft.com

R-CNN (Faster version)

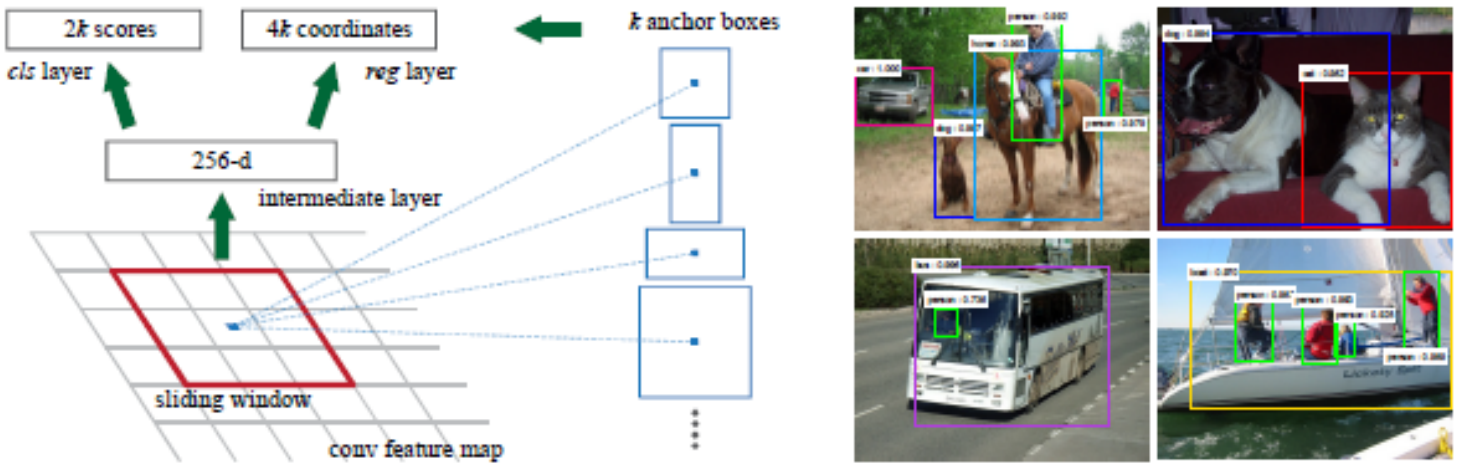


Figure 1: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

R-CNN (Faster version)

- Proposed Region Proposal Networks (RPN)
- Essentially similar to fast R-CNN
- Anchors used in training which overlap with the true RoI boxes
- This leads to some form of robustness in prediction
- Weight sharing between RPNs and Fast R-CNNs employed

R-CNN (Faster version)

Table 1: Detection results on **PASCAL VOC 2007 test set** (trained on VOC 2007 trainval). The detectors are Fast R-CNN with ZF, but using various proposal methods for training and testing.

train-time region proposals		**test-time region proposals**		
method	# boxes	method	# proposals	mAP (%)
SS	2k	SS	2k	58.7
EB	2k	EB	2k	58.6
RPN+ZF, shared	2k	RPN+ZF, shared	300	**59.9**

R-CNN (Faster version)

Table 2: Detection results on **PASCAL VOC 2007 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2k. †: this was reported in [5]; using the repository provided by this paper, this number is higher (68.0±0.3 in six runs).

method	# proposals	data	mAP (%)	time (ms)
SS	2k	07	66.9†	1830
SS	2k	07+12	70.0	1830
RPN+VGG, unshared	300	07	68.5	342
RPN+VGG, shared	300	07	69.9	**198**
RPN+VGG, shared	300	07+12	**73.2**	**198**

Table 3: Detection results on **PASCAL VOC 2012 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07++12": union set of VOC 2007 trainval+test and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2k. †: `http://host.robots.ox.ac.uk:8080/anonymous/HZJTQA.html`. ‡: `http://host.robots.ox.ac.uk:8080/anonymous/YNPLXB.html`

method	# proposals	data	mAP (%)
SS	2k	12	65.7
SS	2k	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**

3D CNN

3D Convolutional Neural Networks for Human Action Recognition

Shuiwang Ji, Wei Xu, Ming Yang, *Member*, *IEEE*, and Kai Yu, *Member*, *IEEE*

3D CNN

Recall: In 2D convolution, the activation at position (x, y) in f-th feature map at ℓ-th layer is obtained by:

$$a_{\ell f}^{xy} = \phi\left(\sum_{c=1}^{C_{(\ell-1)}} \sum_{p=0}^{P_\ell - 1} \sum_{q=0}^{Q_\ell - 1} a_{(\ell-1)c}^{(x+p)(y+q)} w_{\ell fc}^{pq} + b_{\ell f}\right)$$

where:

- C_ℓ denotes number of feature maps in layer $\ell - 1$
- (P_ℓ, Q_ℓ) denotes the height and width of kernel in layer ℓ
- $w_{\ell fc}^{xy}$ denotes the weight at position (x, y) in kernel associated with c-th feature map at layer $\ell - 1$

3D CNN

In 3D convolution, a temporal dimension is additionally considered.

Hence the activation at position (x, y, z) in f-th feature map at ℓ-th layer is obtained by:

$$a_{\ell f}^{xyz} = \phi\left(\sum_{c=1}^{C_{(\ell-1)}} \sum_{p=0}^{P_\ell-1} \sum_{q=0}^{Q_\ell-1} \sum_{r=0}^{R_\ell-1} a_{(\ell-1)c}^{(x+p)(y+q)(z+r)} w_{\ell fc}^{pqr} + b_{\ell f}\right)$$

where:

- R_ℓ denotes size of 3D kernel in temporal dimension
- $w_{\ell fc}^{xyz}$ denotes the weight at position (x, y, z) in kernel associated with c-th feature map at layer $\ell - 1$

3D CNN

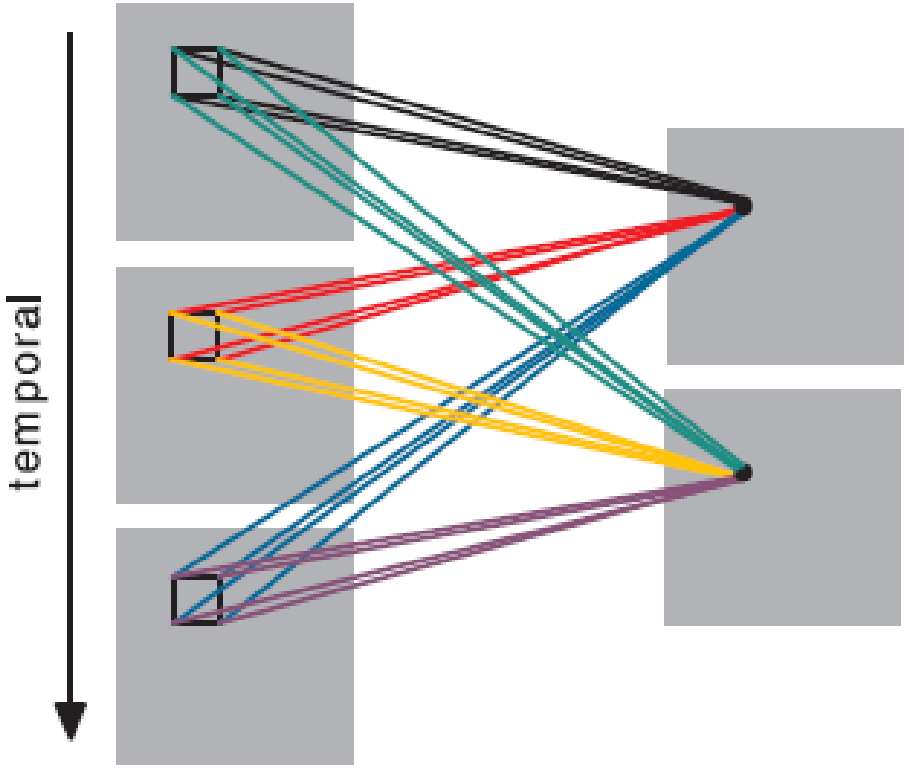


Fig. 2. Extraction of multiple features from contiguous frames. Multiple 3D convolutions can be applied to contiguous frames to extract multiple features. As in Fig. 1, the sets of connections are color-coded so that the shared weights are in the same color. Note that all six sets of connections do not share weights, resulting in two different feature maps on the right.

3D CNN

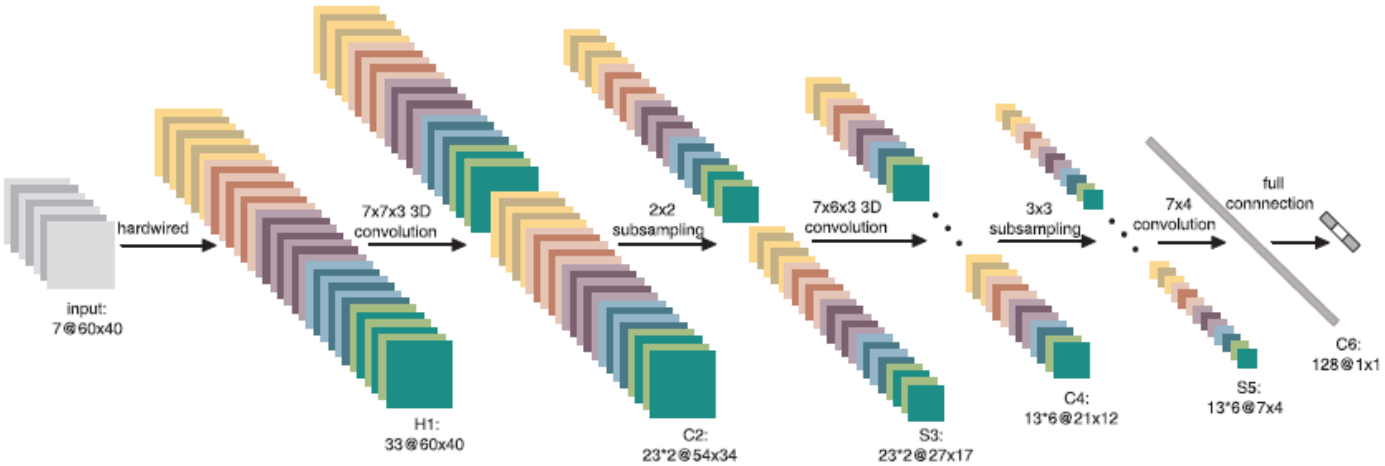


Fig. 3. A 3D CNN architecture for human action recognition. This architecture consists of one hardwired layer, three convolution layers, two subsampling layers, and one full connection layer. Detailed descriptions are given in the text.

3D CNN

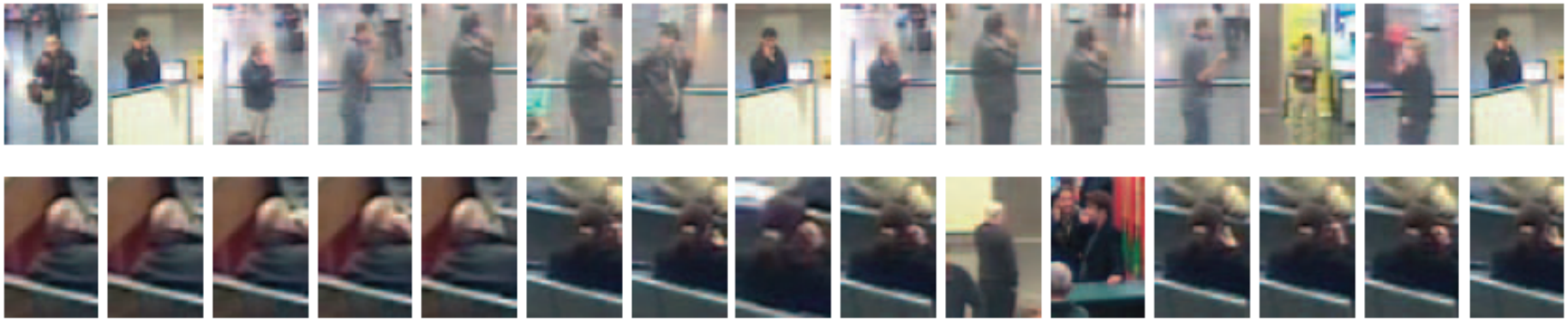


Fig. 10. Sample actions in the CellToEar class. The top row shows actions that are correctly recognized by the combined 3D CNN model, while t
bottom row shows those that are misclassified by the model.

3D CNN

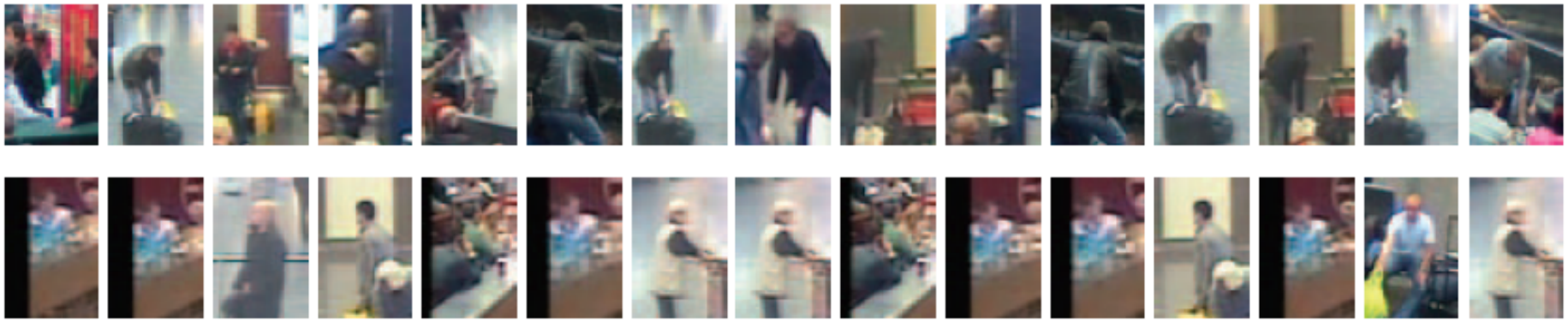


Fig. 11. Sample actions in the ObjectPut class. The top row shows actions that are correctly recognized by the combined 3D CNN model, while t
bottom row shows those that are misclassified by the model.

3D CNN

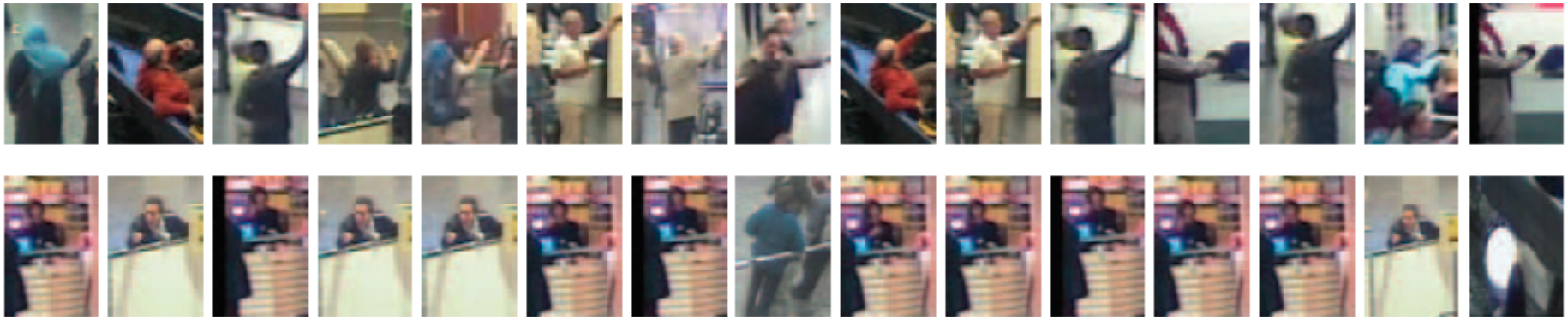


Fig. 12. Sample actions in the Pointing class. The top row shows actions that are correctly recognized by the combined 3D CNN model, while th bottom row shows those that are misclassified by the model.

CNN for 3D Modeling

Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views

Hao Su*, Charles R. Qi*, Yangyan Li, Leonidas J. Guibas
Stanford University

CNN for 3D Modeling

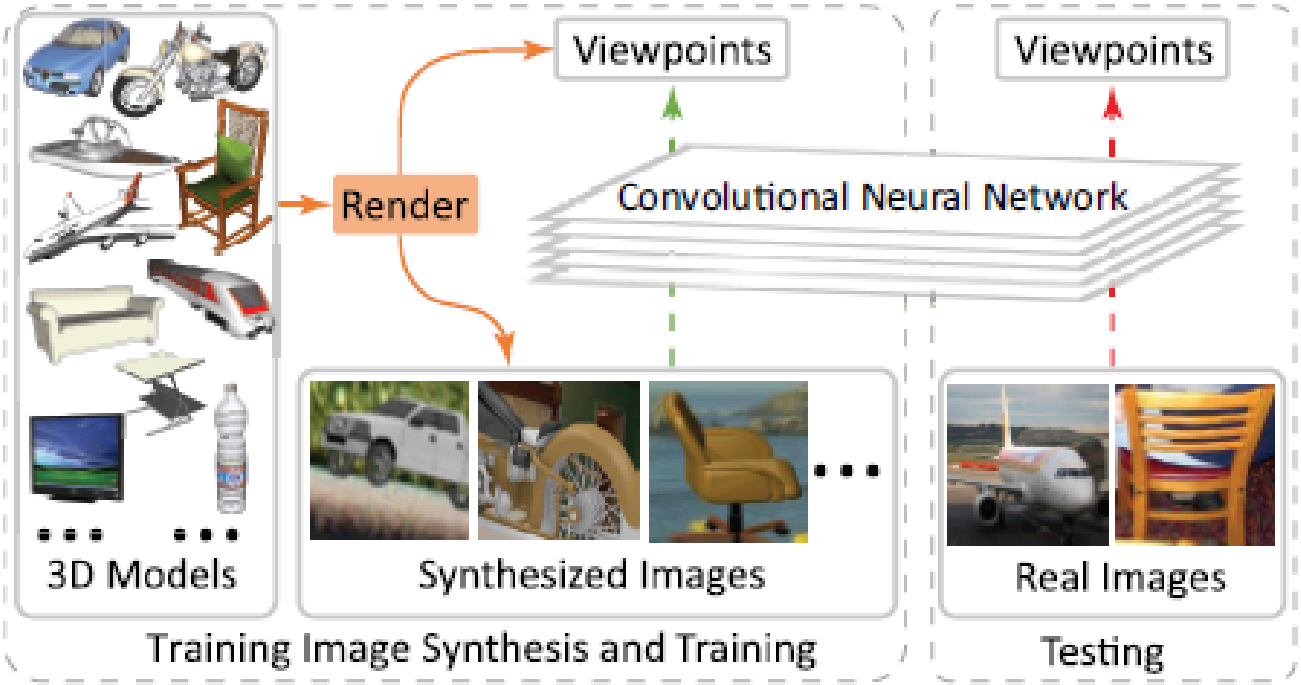


Figure 1. **System overview.** We synthesize training images by overlaying images rendered from large 3D model collections on top of real images. CNN is trained to map images to the ground truth object viewpoints. The training data is a combination of real images and synthesized images. The learned CNN is applied to estimate the viewpoints of objects in real images. Project homepage is at https://shapenet.cs.stanford.edu/projects/RenderForCNN

CNN for 3D Modeling

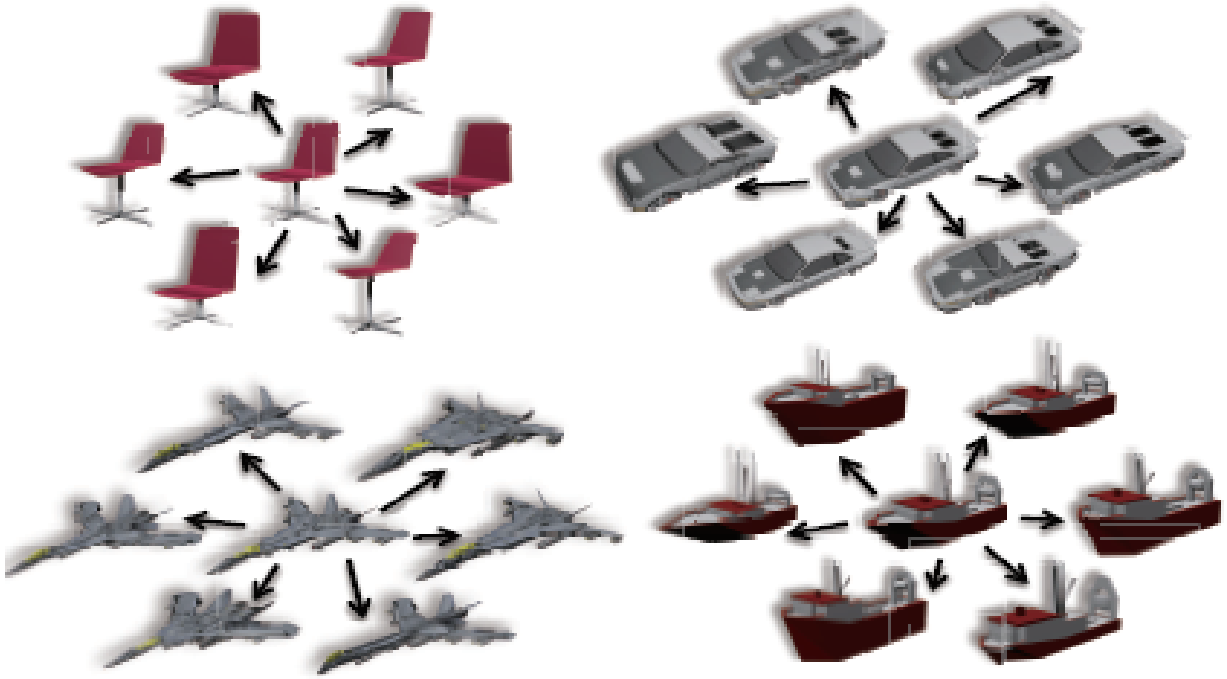


Figure 2. 3D model set augmentation by symmetry-preserving deformation.

CNN for 3D Modeling

Figure 3. **Synthetic image examples.** Three example images are shown for each of the 12 classes from PASCAL 3D+.

CNN for 3D Modeling

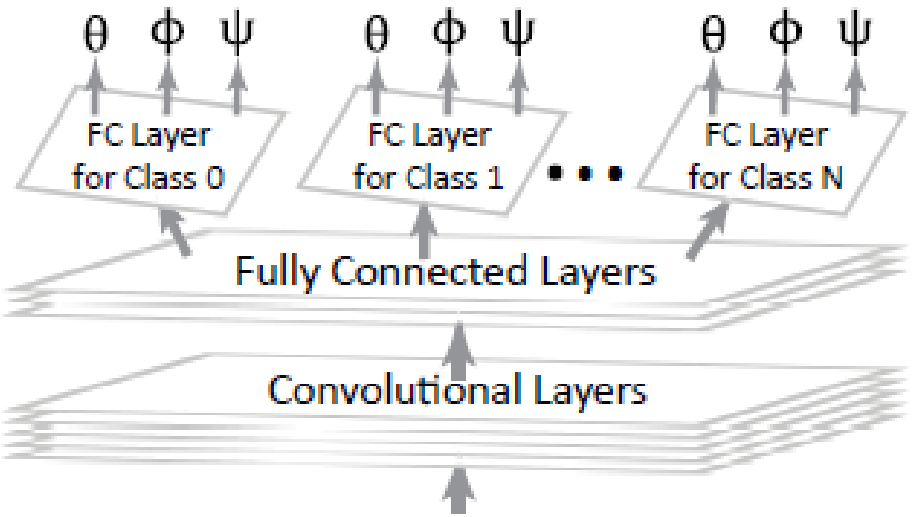


Figure 4. **Network architecture**[2]. Our network includes shared feature layers and class-dependent viewpoint estimation layers.

CNN for 3D Modeling

- Goal: To predict the viewpoint of an object in an image
- Viewpoint is with respect to the camera
- Viewpoint characterized by three parameters (θ, ϕ, ψ) denoting the azimuth, elevation and in-plane rotation angles.
- Opensource ShapeNet† used

† https://www.shapenet.org/

CNN for 3D Modeling

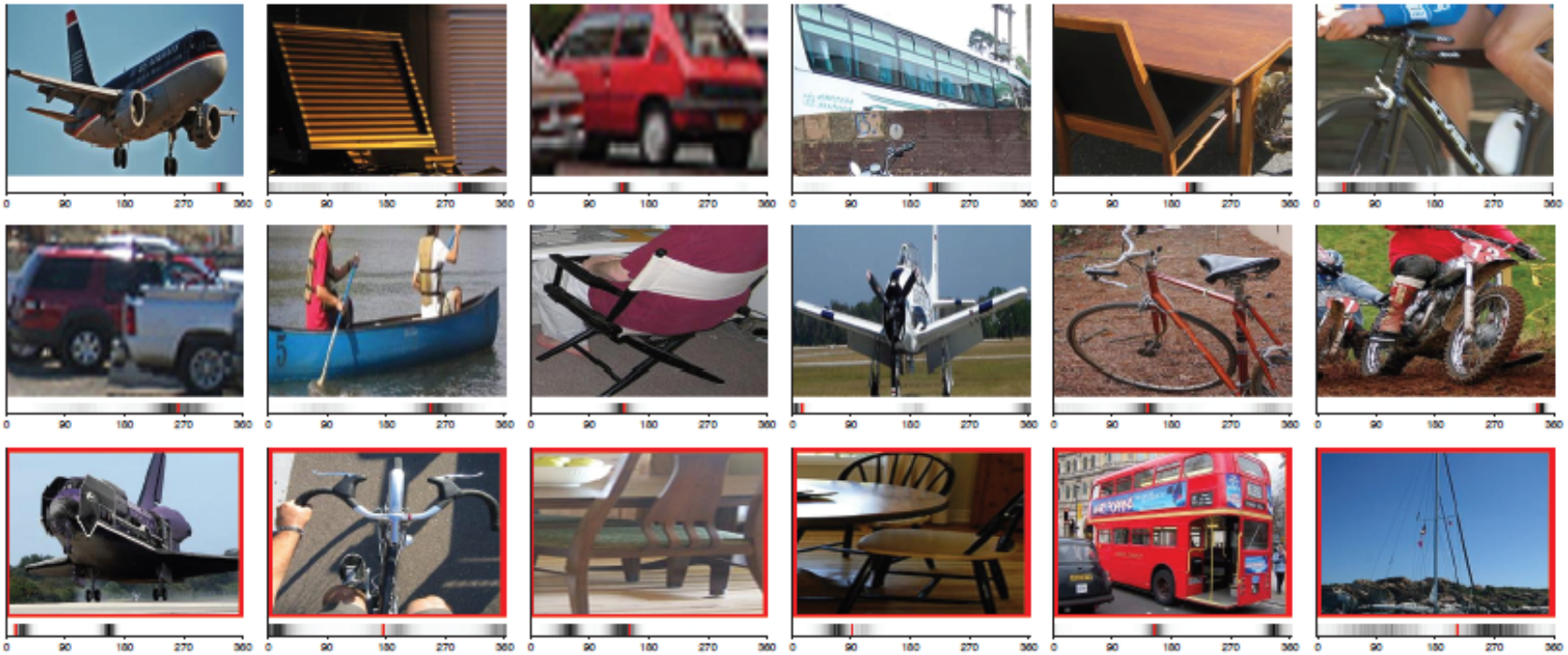


Figure 10. **Viewpoint estimation example results.** The bar under each image indicates the 360-class confidences (black means high confidence) corresponding to $0^\circ \sim 360^\circ$ (with object facing towards us as 0° and rotating clockwise). The red vertical bar indicates the ground truth. The first two rows are positive cases, the third row is negative case (with red box surrounding the image).

Left Ventricle Segmentation

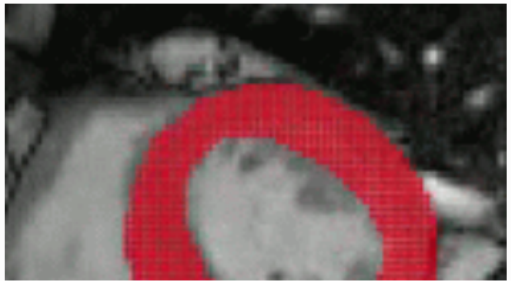
Cardiac Atlas Project

Studies | Challenges | Resources | Publications

LEFT VENTRICULAR SEGMENTATION CHALLENGE

Establishing ground truth consensus segmentation images from automated methods

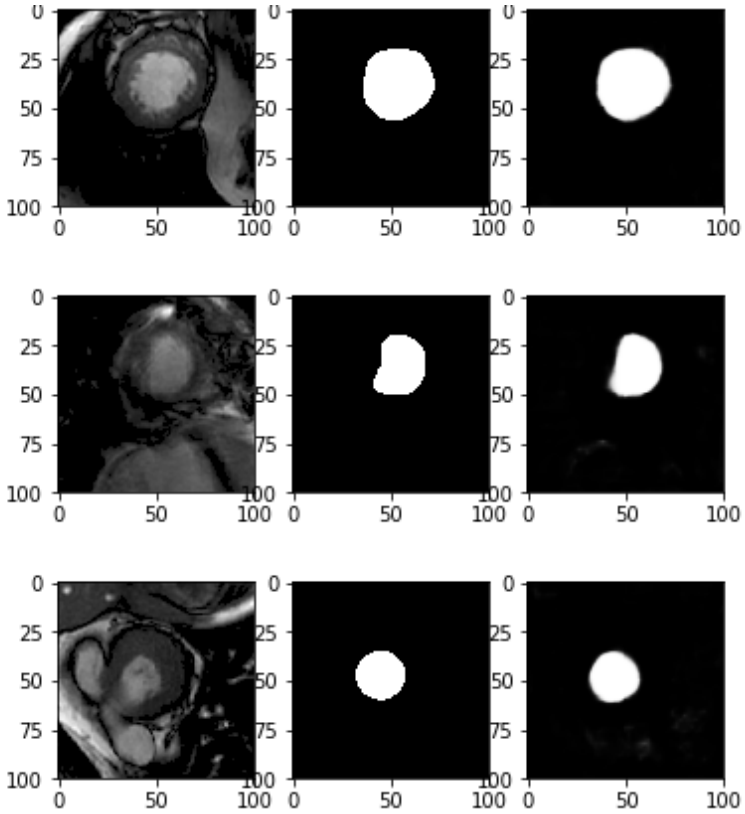
About



One major challenge for developing a 4D segmentation algorithm is the lack of available large set of ground truth that are defined for the whole cardiac frames and slices. Initiated from the 2011 LV Segmentation Challenge that was held for the 2011 STACOM Workshop, we have started up a larger collaborative project to establish the ground truth or the consensus segmentation images for myocardium. The segmentation challenge is therefore set to open for new algorithms to participate. We aim to establish consensus segmentation images from a large set of data. We used modified STAPLE method to generate the consensus images from the contributing participants (raters). However, before the consensus images can be robustly established, we need a lot of participants, which are semi or fully automated segmentation methods. The more people join this work, the better the consensus images. Everybody can participate in this work, particularly for students and researchers in the field of fully automatic

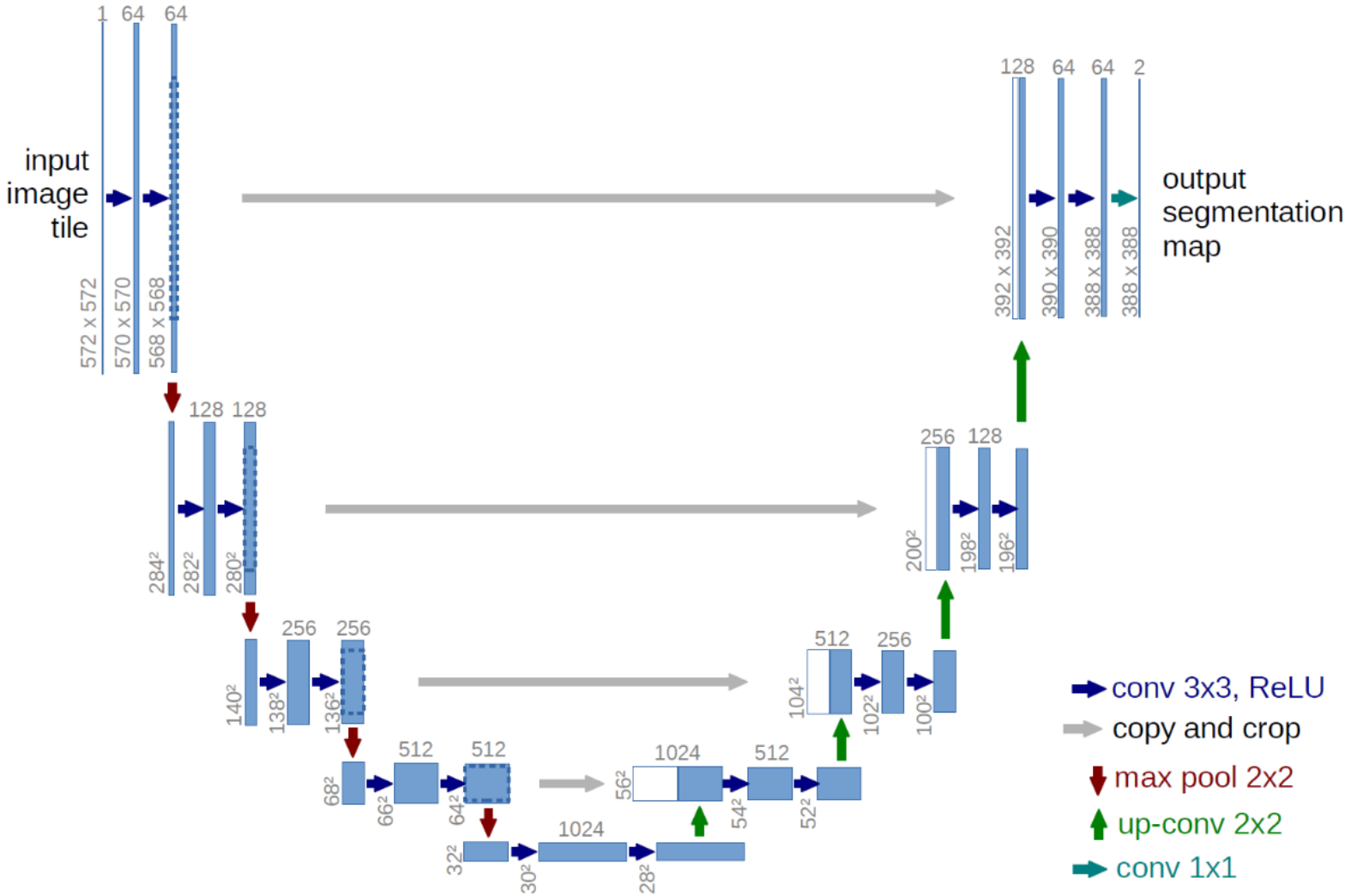
http://www.cardiacatlas.org/challenges/lv-segmentation-challenge/

Left Ventricle Segmentation



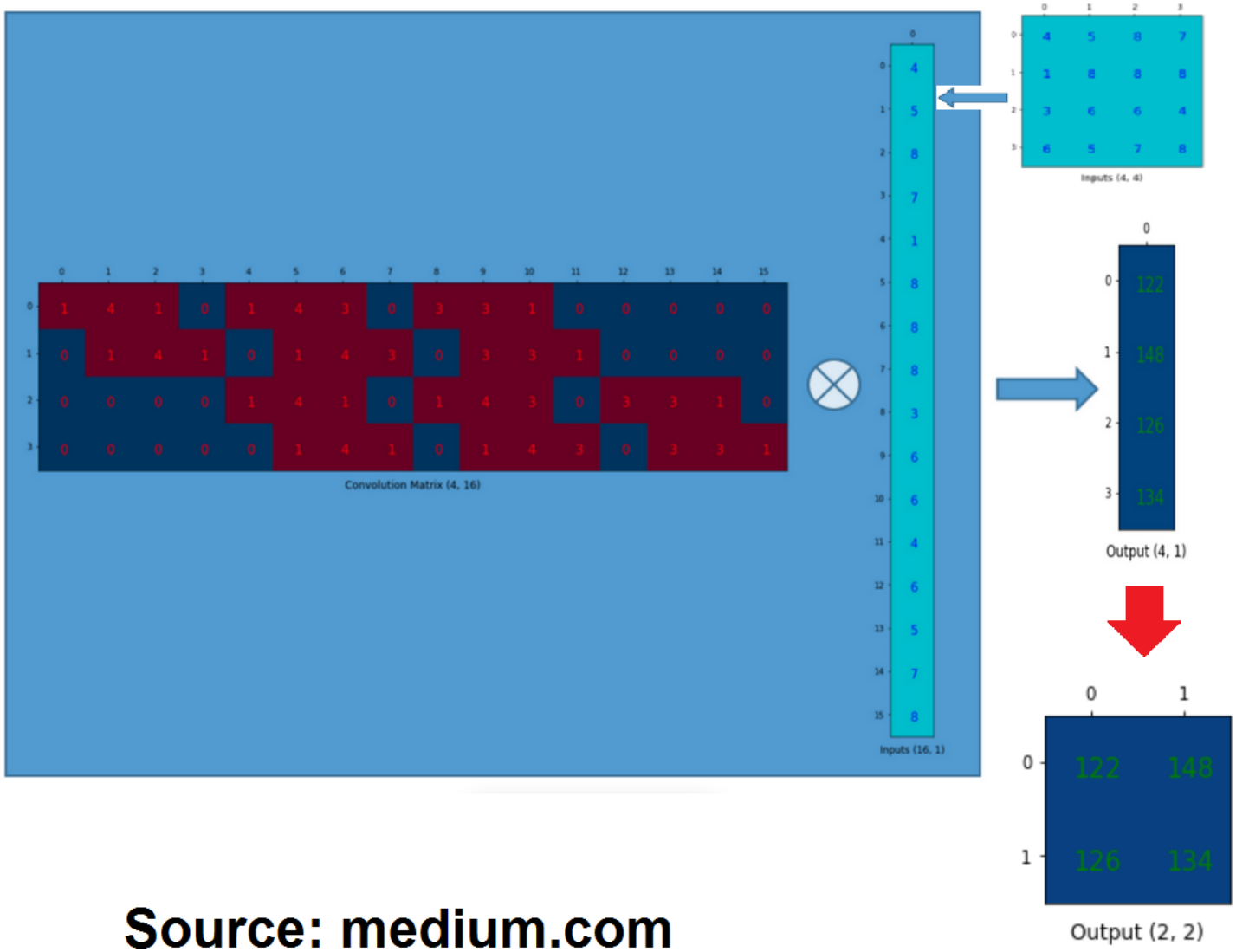
Ventricle of Human Heart | **Mask indicating ventricle location** | **Predicted Mask**

U-Net†



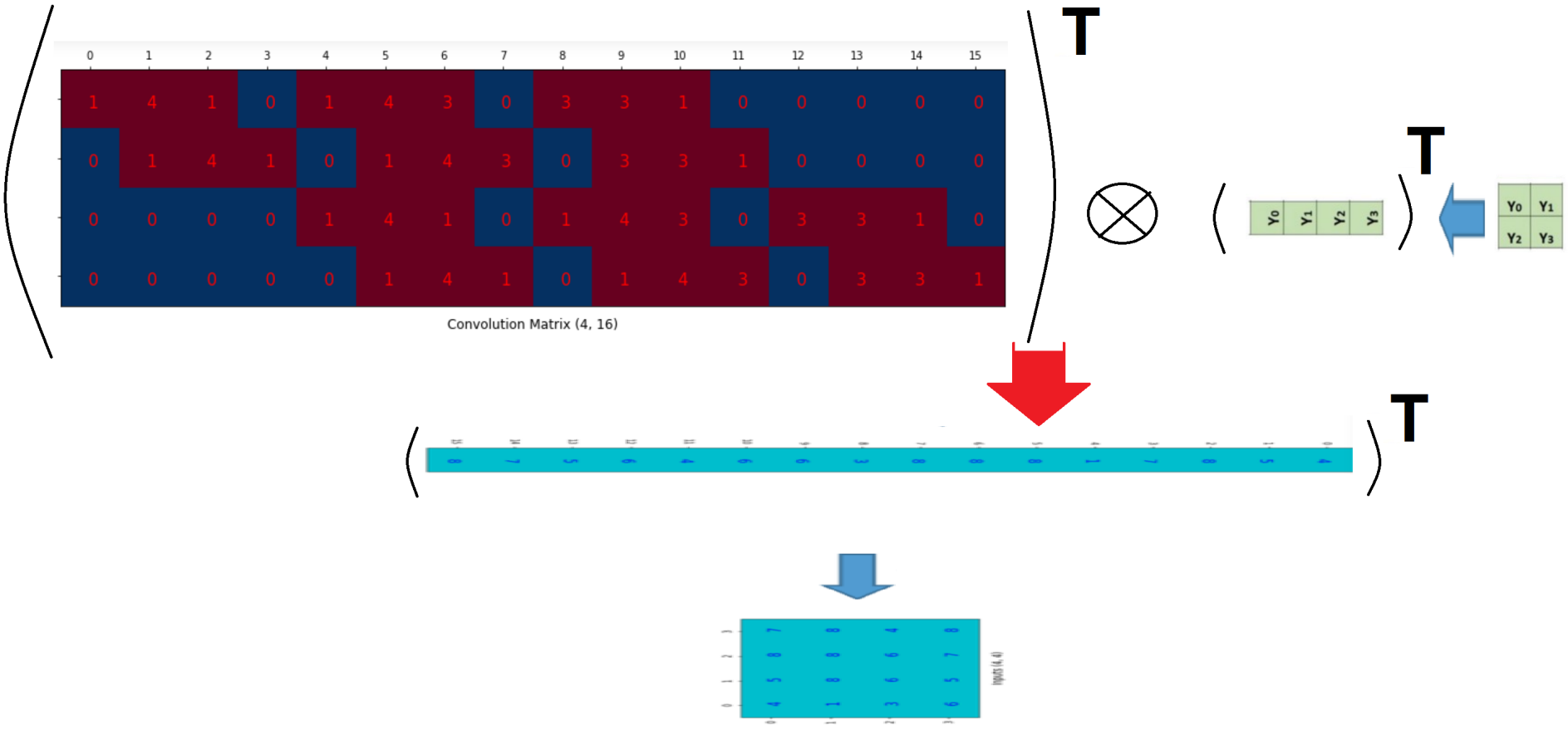
† U-Net: Convolutional Networks for Biomedical Image Segmentation. O. Ronneberger, P. Fischer, T. Brox. MICCAI, 2015.

Downsampling as Matrix Multiplication



Source: medium.com

U-Net: Upsampling



Upsampling operation (or Transpose Convolution)
Source: medium.com. towardsdatascience.com

Probing CNN

Visualizing and Understanding Convolutional Networks

Matthew D. Zeiler and Rob Fergus

Dept. of Computer Science,
New York University, USA
{zeiler,fergus}@cs.nyu.edu

Probing CNN

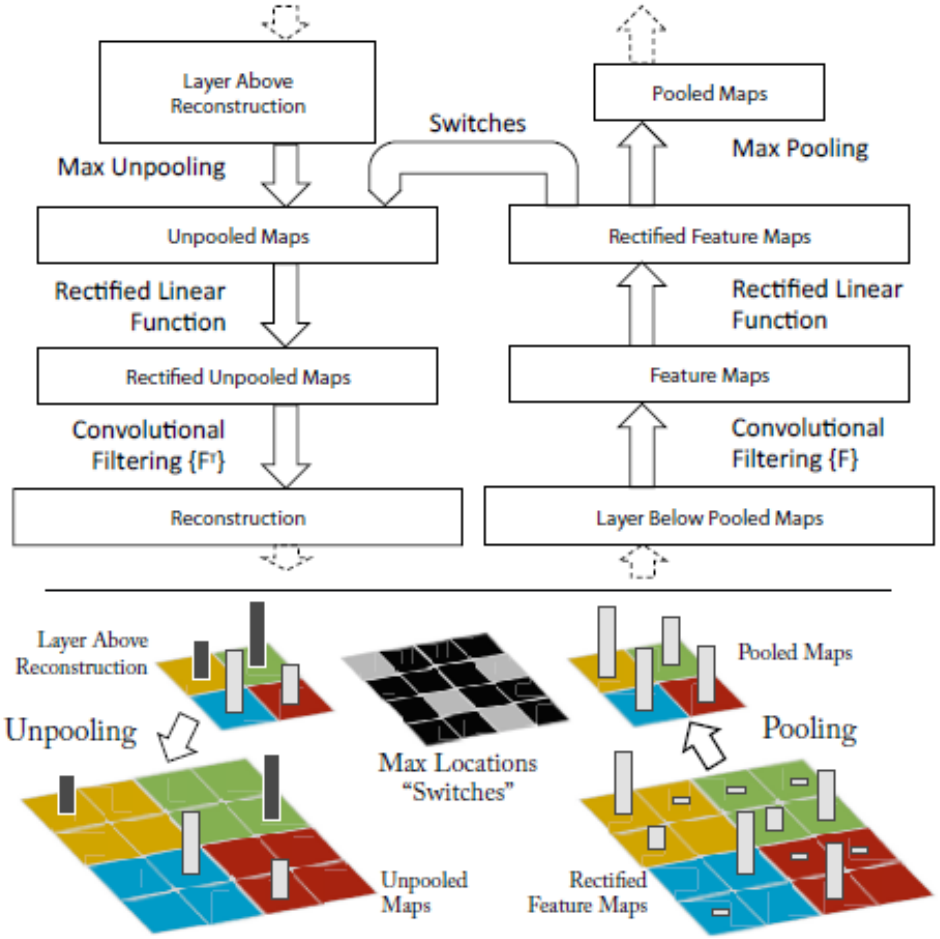


Fig. 1. Top: A deconvnet layer (left) attached to a convnet layer (right). The deconvnet will reconstruct an approximate version of the convnet features from the layer beneath. Bottom: An illustration of the unpooling operation in the deconvnet, using *switches* which record the location of the local max in each pooling region (colored zones) during pooling in the convnet. The black/white bars are negative/positive activations within the feature map.

Probing CNN

Fig. 2. Visualization of features in a fully trained model. For layers 2-5 we show the top 9 activations in a random subset of feature maps across the validation data, projected down to pixel space using our deconvolutional network approach. Our reconstructions are *not* samples from the model: they are reconstructed patterns from the validation set that cause high activations in a given feature map. For each feature map we also show the corresponding image patches. Note: (i) the the strong grouping within each feature map, (ii) greater invariance at higher layers and (iii) exaggeration of discriminative parts of the image, e.g. eyes and noses of dogs (layer 4, row 1, cols 1). Best viewed in electronic form. The compression artifacts are a consequence of the 30Mb submission limit, not the reconstruction algorithm itself.

Probing CNN

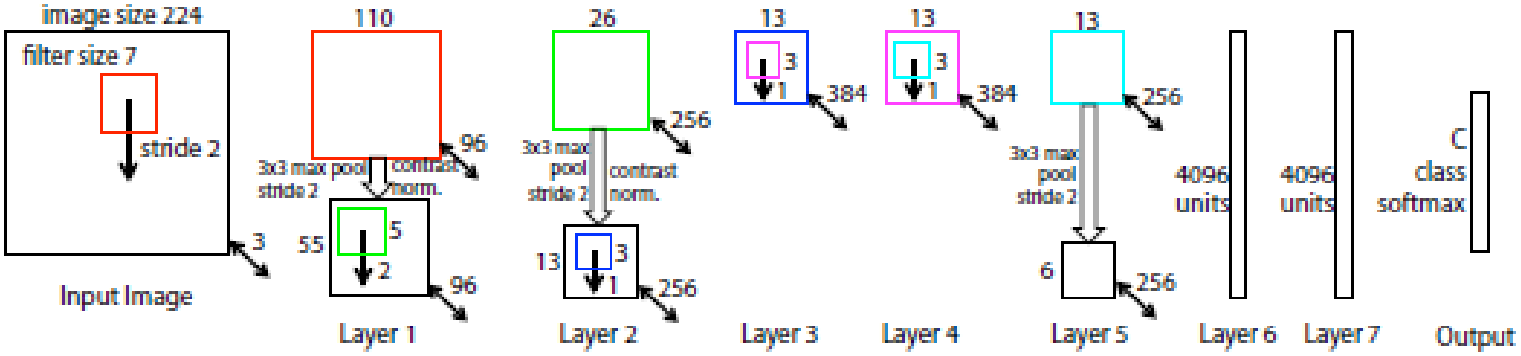


Fig. 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

Probing CNN

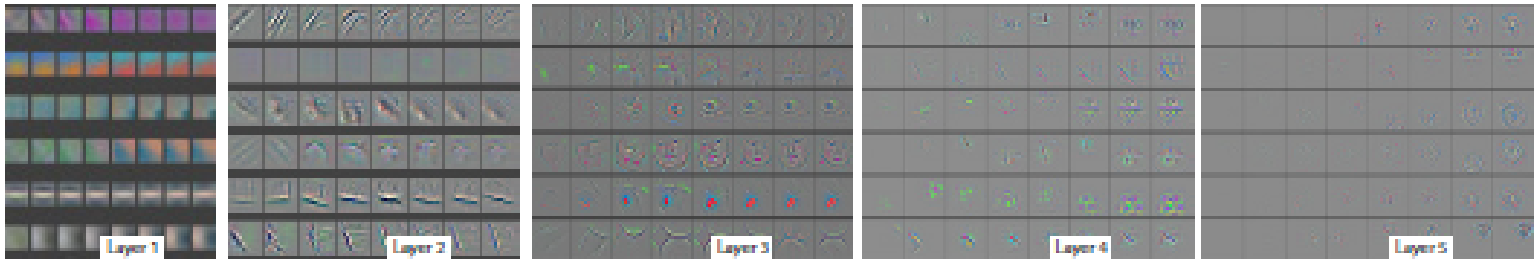


Fig. 4. Evolution of a randomly chosen subset of model features through training. Each layer's features are displayed in a different block. Within each block, we show a randomly chosen subset of features at epochs [1,2,5,10,20,30,40,64]. The visualization shows the strongest activation (across all training examples) for a given feature map, projected down to pixel space using our deconvnet approach. Color contrast is artificially enhanced and the figure is best viewed in electronic form.

Probing CNN

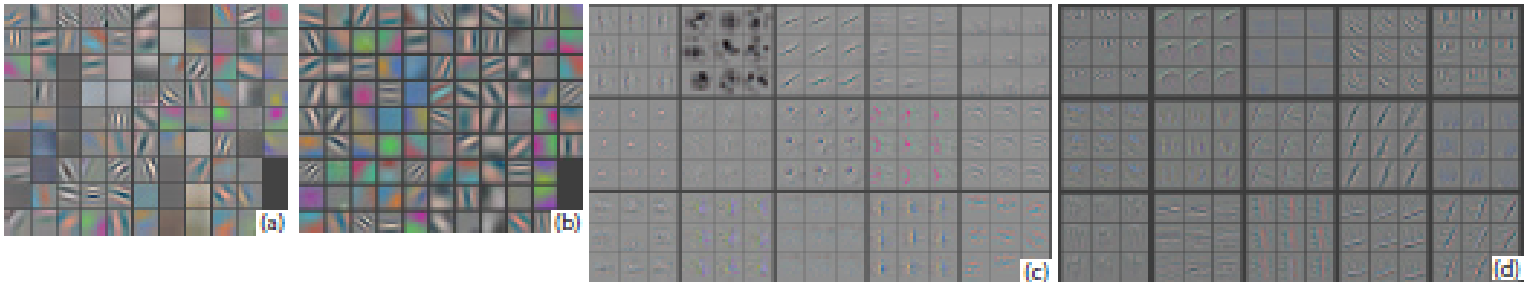


Fig. 5. (a): 1st layer features without feature scale clipping. Note that one feature dominates. (b): 1st layer features from Krizhevsky *et al.* [18]. (c): Our 1st layer features. The smaller stride (2 vs 4) and filter size (7x7 vs 11x11) results in more distinctive features and fewer "dead" features. (d): Visualizations of 2nd layer features from Krizhevsky *et al.* [18]. (e): Visualizations of our 2nd layer features. These are cleaner, with no aliasing artifacts that are visible in (d).

Probing CNN

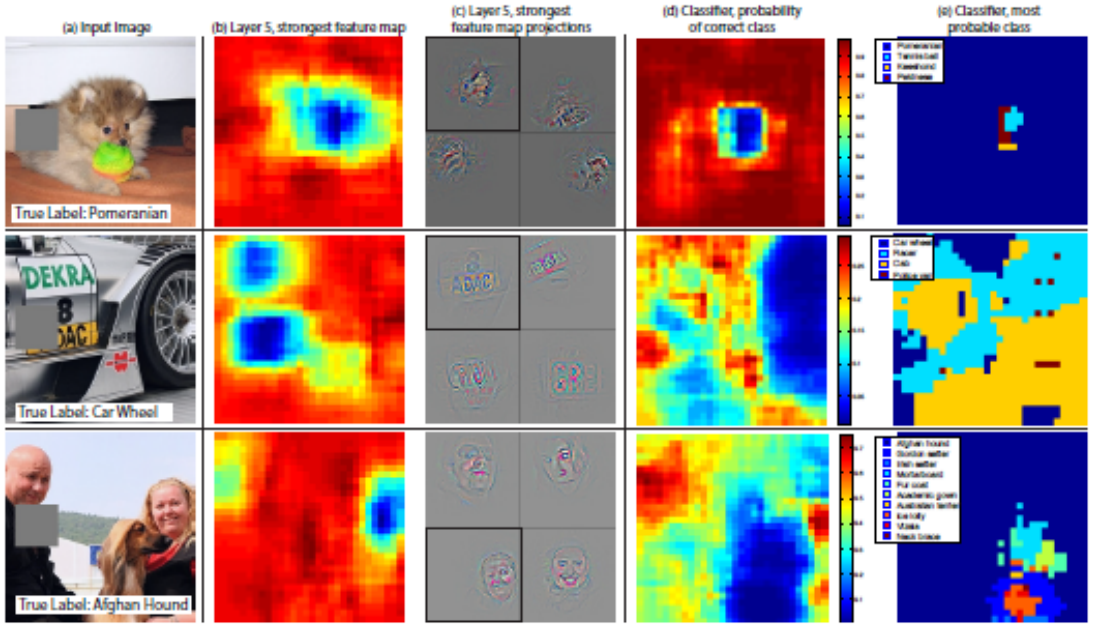


Fig. 6. Three test examples where we systematically cover up different portions of the scene with a gray square (1st column) and see how the top (layer 5) feature maps ((b) & (c)) and classifier output ((d) & (e)) changes. (b): for each position of the gray scale, we record the total activation in one layer 5 feature map (the one with the strongest response in the unoccluded image). (c): a visualization of this feature map projected down into the input image (black square), along with visualizations of this map from other images. The first row example shows the strongest feature to be the dog's face. When this is covered-up the activity in the feature map decreases (blue area in (b)). (d): a map of correct class probability, as a function of the position of the gray square. E.g. when the dog's face is obscured, the probability for "pomeranian" drops significantly. (e): the most probable label as a function of occluder position. E.g. in the 1st row, for most locations it is "pomeranian", but if the dog's face is obscured but not the ball, then it predicts "tennis ball". In the 2nd example, text on the car is the strongest feature in layer 5, but the classifier is most sensitive to the wheel. The 3rd example contains multiple objects. The strongest feature in layer 5 picks out the faces, but the classifier is sensitive to the dog (blue region in (d)), since it uses multiple feature maps.