Skoltech Computer Vision

The Third Christmas Colloquium on Computer Vision

Сколково, 26.12.2017

Web: http://sites.skoltech.ru/compvision/cccv17/

1. Computer Vision In the Wild
2. Generating Images
3. Efficient Convolutional Networks
4. Beyond Computer Vision

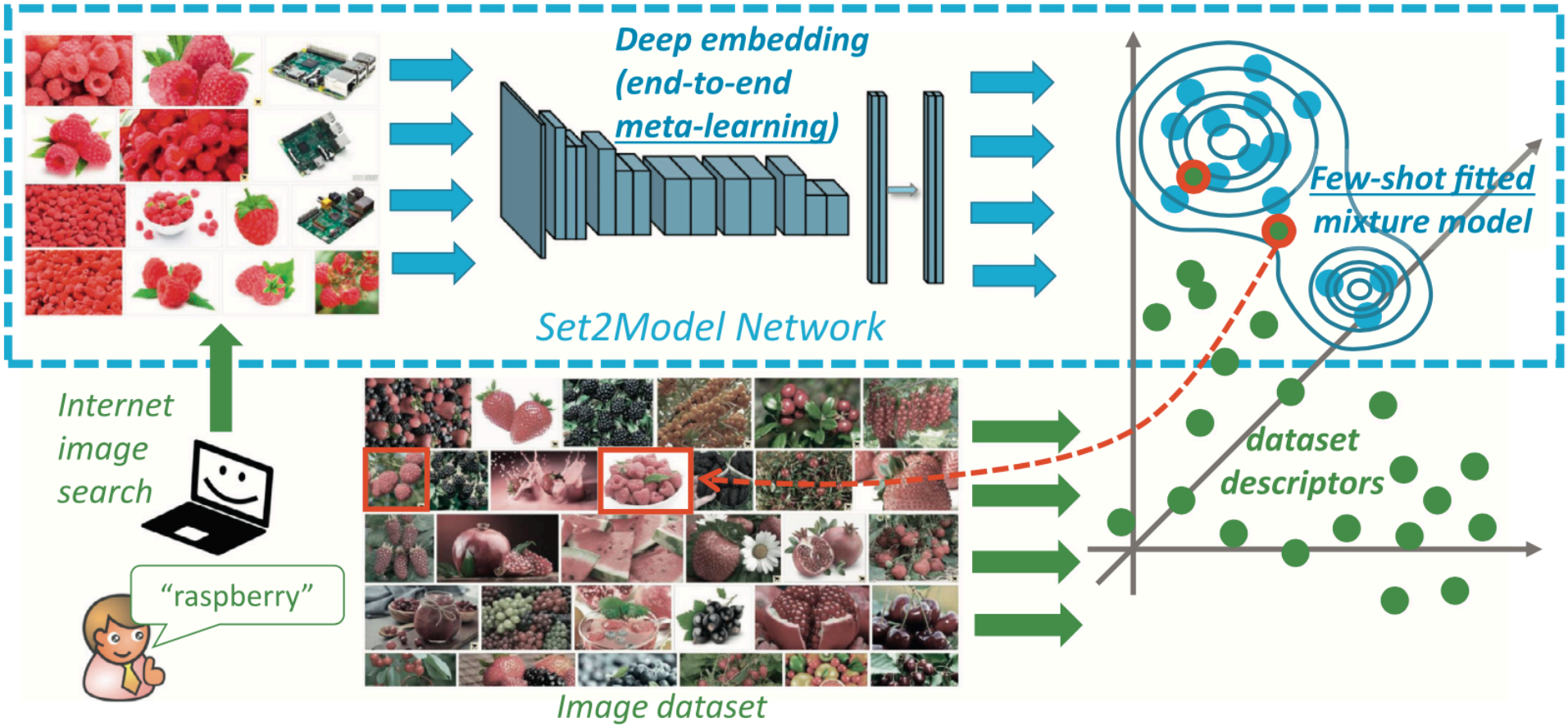
Set2Model Networks: Learning Discriminatively To Learn Generative Visual Models (CVPR 2017, CVIU)

Alexander Vakhitov (Skoltech)

Задача few-shot learning concepts/meta-learning (learning-to-learn):

- Metric-learning (Siamese net)
- Nearest Class Mean (NCM) classifier as a distance to a mean of image class descriptors
- Matching net

Пример. Поиск по содержанию



Meta learning (1)

Training set (one concept) → Descriptor set

$$X = \{x^1, x^2, \ldots, x^N\} \qquad D = \{d^1, d^2, \ldots, d^N\} \quad d^i = f(x^i; w)$$

Задача – оценить параметры **w**, которые описывают различные понятия.

Обучающие выборки – примеры различных понятий $\{\mathcal{T}_i = (X_i, Z_{+,i}, Z_{-,i})\}$

$X_i = \{x_i^1, x_i^2, \ldots, x_i^{N_i}\}$ Concept describing set (web images)

$Z_{+,i} = \{z_{+,i}^1, z_{+,i}^2, \ldots, z_{+,i}^{M_{+,i}}\}$ Relevant examples set

$Z_{-,i} = \{z_{-,i}^1, z_{-,i}^2, \ldots, z_{-,i}^{M_{-,i}}\}$ Irrelevant examples set

Простые решения

mean-based system (**AVG**)

$$r_{AVG}(z|X; w) = \left(\frac{1}{N}\sum f(x^i, w)\right)^T z.$$

nearest neighbor (**NN**) ranker

$$r_{NN}(z|X; w) = \max_i \left(f(x^i, w)\right)^T z.$$

SVM

$$r_{SVM}(z|X; w) = z^T u(\{f(x^i, w)\}_{i=1}^N)$$

Предложенный подход

Training set
(one concept)

$$X = \{x^1, x^2, \ldots, x^N\}$$

Descriptor set

$$D = \{d^1, d^2, \ldots, d^N\} \; d^i = f(x^i; w)$$

Probabilistic generative model (GMM)

$$p_{GM}(d;\ \theta)$$

Learning

$$F : X \to \theta^*$$

$$\theta^* = \arg\max_{\theta} l(\theta|D), \quad l(\theta|D) = \sum_{i=1}^{N} \log p_{GM}(d^i, \theta).$$

Inference

$$p(z|X; w) = p_{GM}(f(z; w), F(X; w)).$$

В идеале

$$p(z^k_{+,i}|X_i; w) > p(z^l_{-,i}|X_i; w)$$

S2M network

Функция потерь (не дифференцируемая)

$$L(w) = \sum_i \frac{1}{M_{+,i} M_{-,i}} \sum_{k=1}^{M_{+,i}} \sum_{l=1}^{M_{-,i}} \chi\left[R^k_{+,i} < R^l_{-,i}\right]$$

$$R_{+,i} = \{p(z^1_{+,i}|X_i; w),\ p(z^2_{+,i}|X_i; w) \dots p(z^{M_{+,i}}_{+,i}|X_i; w)\}$$

Кусочно-дифференцируемая аппроксимация – **histogram loss** (Ustinova, Lempitsky, NIPS 2016)

$$L(w) = \sum_i \sum_{k=1}^{B} h^k_{-,i} \sum_{l=1}^{B} h^l_{+,i}$$

Обучение – SGD. Для реализации нужно найти частные производные $\frac{\partial \theta^*}{\partial d^i_{(j)}}$

$$p_{GMM}(d, \theta) = \sum_{i=1}^{k} v_i \mathcal{N}(d, \mu_i, \Sigma_i)$$

$$c(\theta) = \sum_{i=1}^{k} v_i - 1 = 0$$

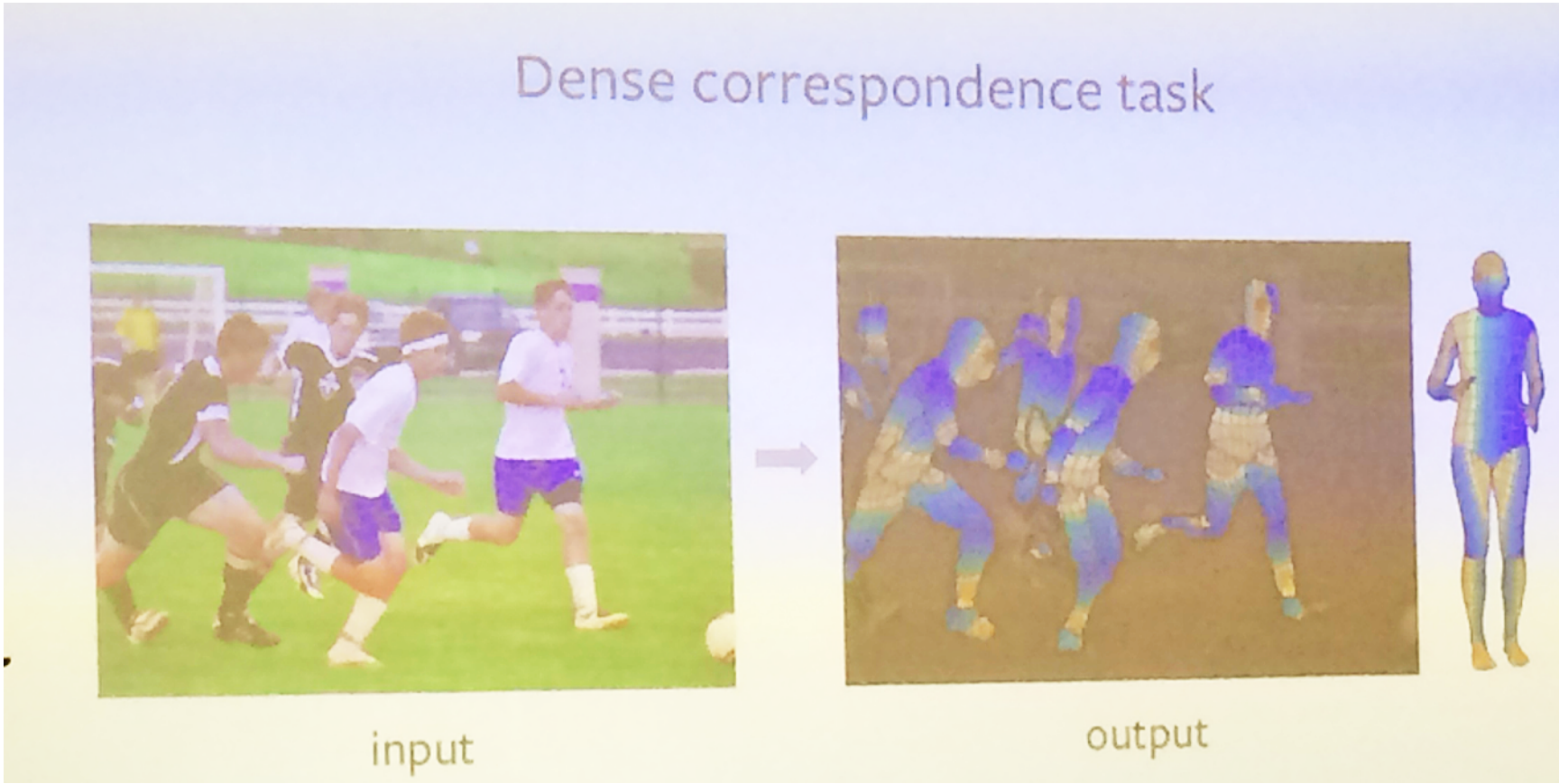
$$\begin{cases} \frac{\partial^2}{\partial \theta^2}\mathcal{L}(\theta^*, \lambda^*)\nabla^T\theta^* + \frac{\partial^2}{\partial \lambda \partial \theta}\mathcal{L}(\theta^*, \lambda^*)\nabla^T\lambda^* \\ \quad + \frac{\partial^2}{\partial d^i \partial \theta}\mathcal{L}(\theta^*, \lambda^*) = 0, \\ \frac{\partial^2}{\partial \theta \partial \lambda}\mathcal{L}(\theta^*, \lambda^*)\nabla^T\theta^* + \frac{\partial^2}{\partial \lambda^2}\mathcal{L}(\theta^*, \lambda^*)\nabla^T\lambda^* \\ \quad + \frac{\partial^2}{\partial d^i \partial \lambda}\mathcal{L}(\theta^*, \lambda^*) = 0, \end{cases}$$

Эксперименты: Omniglot, ImageNet(-ILSVRC), RGBD-Object, RGBD-Scenes, Oxford Flowers

Multi-Person Human Pose Estimation In The Wild

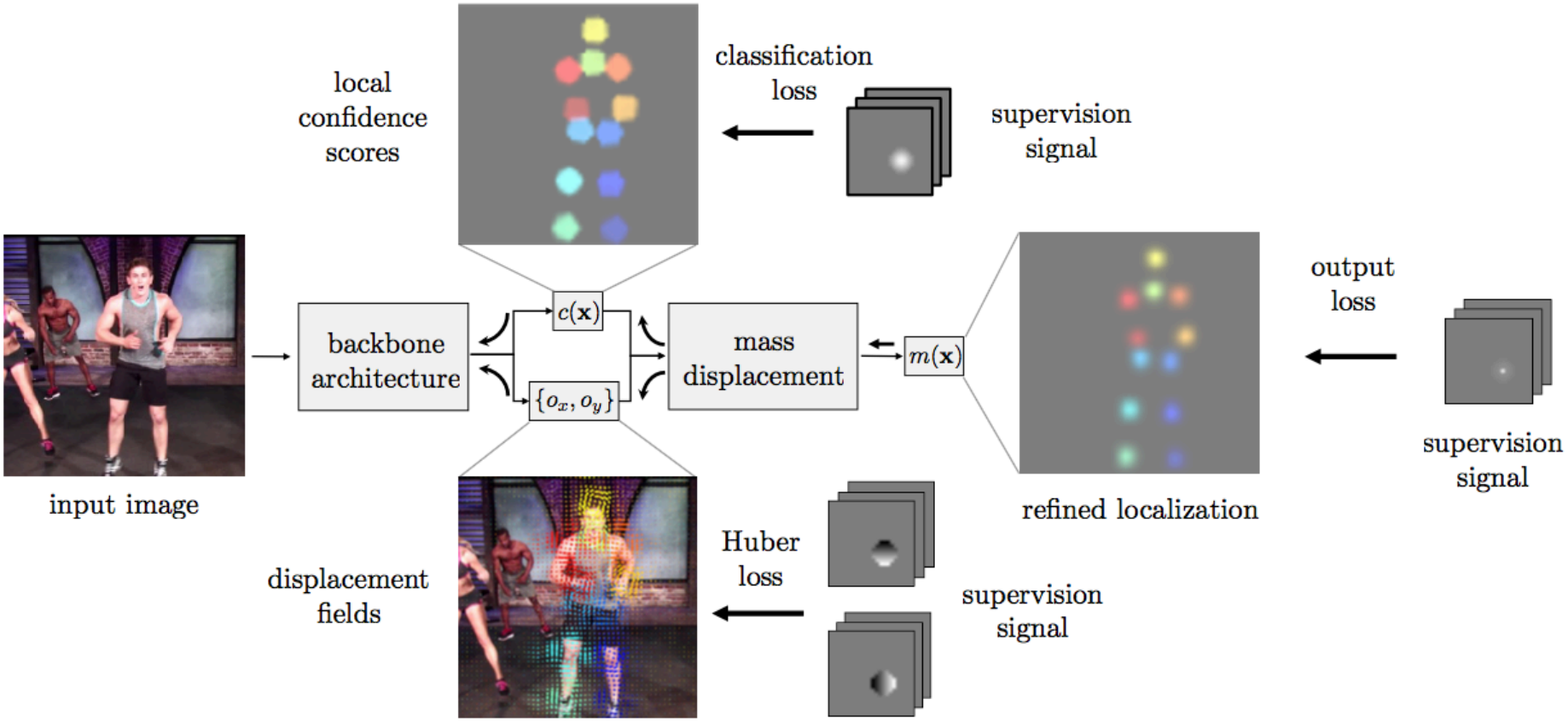
Natalia Neverova (Facebook AI Research)

В основном – по материалам препринта Mass Displacement Networks (arXiv: 1708.03816)



Mass displacement network

CNN's outputs for any position **x**:
1) local confidence for the presence of a feature c(**x**)
2) predicted structure's position – horizontal/vertical displacement **o**(**x**)



A **voting** operation combines these and collapses the smooth CNN predictions into sharp landmarks.

$$m(\mathbf{x}_o) = \sum_{\mathbf{x}} K(\mathbf{x}_o - [\mathbf{x} + \mathbf{o}(\mathbf{x})])c(\mathbf{x}) \quad \Rightarrow \quad m(\mathbf{x}_o) = 1 - \prod_{\mathbf{x}} [1 - K(\mathbf{x}_o - [\mathbf{x} + \mathbf{o}(\mathbf{x})])c(\mathbf{x})]$$

1. Computer Vision In the Wild
2. Generating Images
3. Efficient Convolutional Networks
4. Beyond Computer Vision

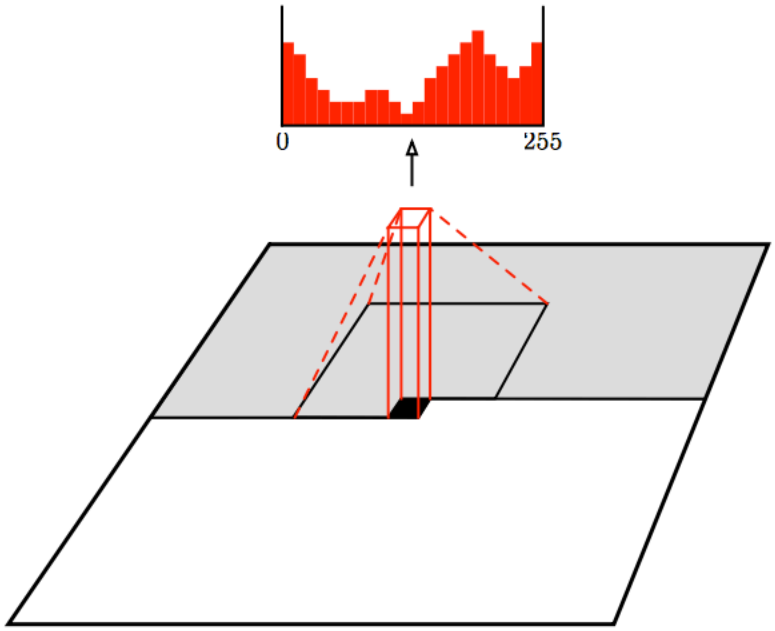
PixelCNN models with Auxiliary Variables for Natural Image Modeling (ICML 2017)

Alexander Kolesnikov (IST Austria)

Семплирование изображений - Pixel CNN

Google DeepMind (van den Oord et al, NIPS 2016)

$$p(X) = \prod_{j=1}^{n} p\left(x_j | x_1, \ldots, x_{j-1}\right)$$



1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

Auxiliary Variables

The PixelCNN's loss function measures the amount of uncertainty of the color value of each pixel, which is dominated by hard-to-predict low-level texture patterns. The model is encouraged to represent such textures well, while more essential image details (object shapes) are neglected

$$p(X, \widehat{X}) = p_{\hat{\theta}}(\widehat{X}) p_{\theta}(X|\widehat{X})$$

$\widehat{X}$ - auxiliary variable (4-bit per pixel quantized grayscale version of the original 24-bits color image X)

PixelCNN++ (low resolution)

upscaling a low-res into a high-res image

$$p_{\theta}(X|\widehat{X}) = \prod_{j=1}^{n} p_{\theta}\left(x_j | x_1, \ldots, x_{j-1}; f_w(\widehat{X})\right)$$

f_w – embedding function with weights **w**

PixelCNN+bias the convolutions of every residual block by adding the computed embedding

Learning of parameters $\Theta = (\hat{\theta}, \theta, w)$

$$\Theta^* = \underset{\Theta}{\operatorname{argmax}} \sum_{X \in \mathcal{D}} \log p(X, \widehat{X}) = \underset{\Theta}{\operatorname{argmax}} \sum_{X \in \mathcal{D}} \log p_{\theta}(X|\psi(X)) + \sum_{X \in \mathcal{D}} \log p_{\hat{\theta}}(\psi(X))$$

Aux var is an image-> Pyramid PixelCNN (breaks the image model into a series of simpler sub-models)

Примеры - CIFAR

GANs for Biological Image Synthesis (ICCV 2017)

Anton Osokin (Higher School of Economics)

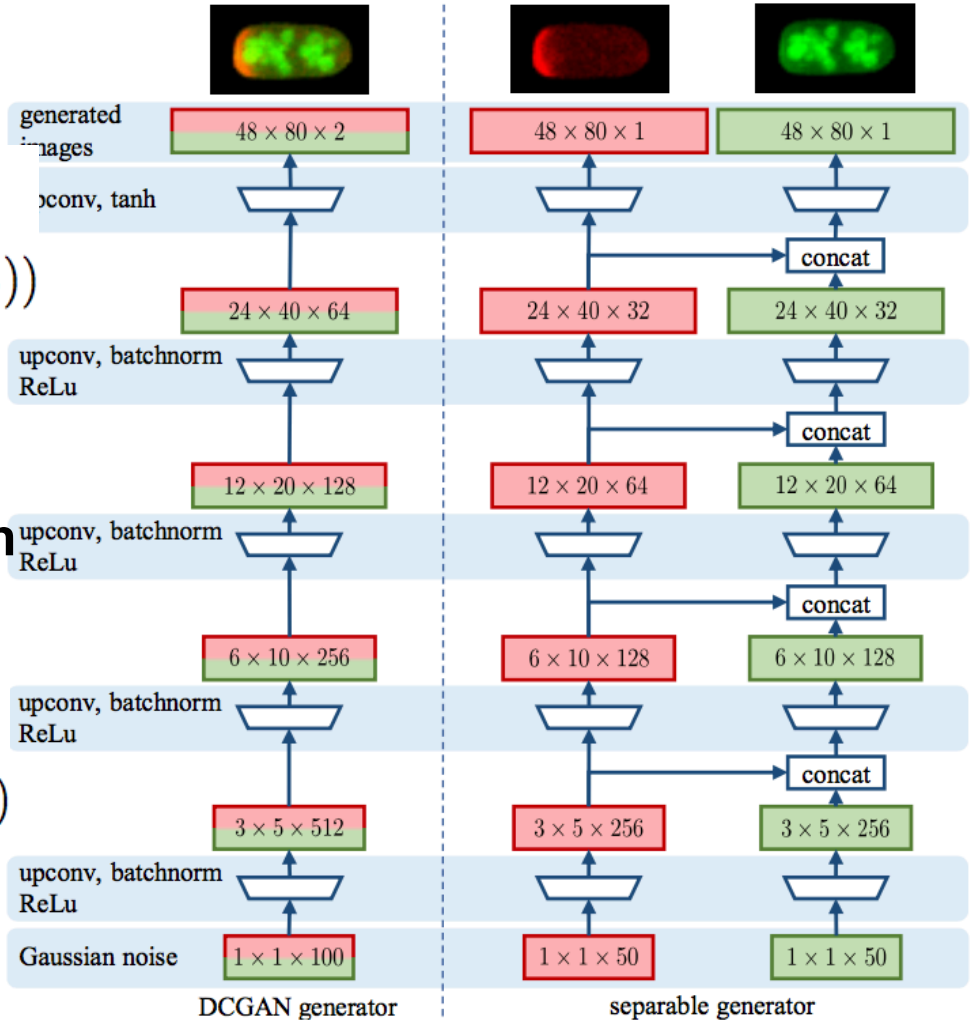
GAN loss (equivalent to Jensen-Shannon divergence)

$$L(\theta_D, \theta_G) = \mathbf{E}_{\boldsymbol{x} \sim \mathrm{P}_{\text{data}}} \log D(\boldsymbol{x}; \theta_D) + \mathbf{E}_{\boldsymbol{z} \sim \mathrm{P}_{\boldsymbol{z}}} \log(1 - D(G(\boldsymbol{z}; \theta_G); \theta_D))$$

Approximation of Wasserstein (earth distance) loss (*arXiv:1704.00028*)

$$W(\theta_D, \theta_G) = \mathbf{E}_{\boldsymbol{z} \sim \mathrm{P}_{\boldsymbol{z}}} D(G(\boldsymbol{z}; \theta_G); \theta_D) - \mathbf{E}_{\boldsymbol{x} \sim \mathrm{P}_{\text{data}}} D(\boldsymbol{x}; \theta_D) + R(\theta_D)$$

Лучше, если данные лежат на low-dimensional manifolds



Deep Image Prior

Dmitry Ulyanov (Yandex/Skoltech)

По материалам препринта 1711.10925

z – random code vector

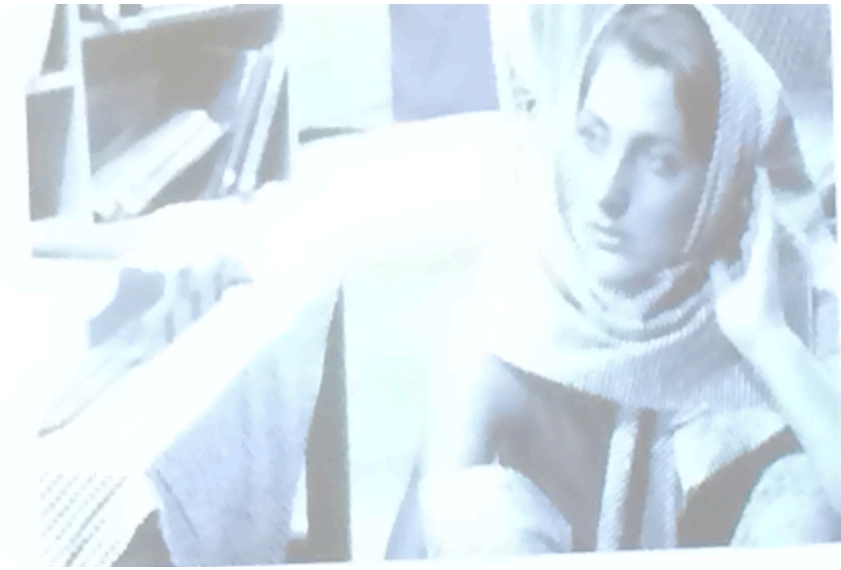
$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, E(f_\theta(z); x_0) \qquad x^* = f_{\theta^*}(z)$$

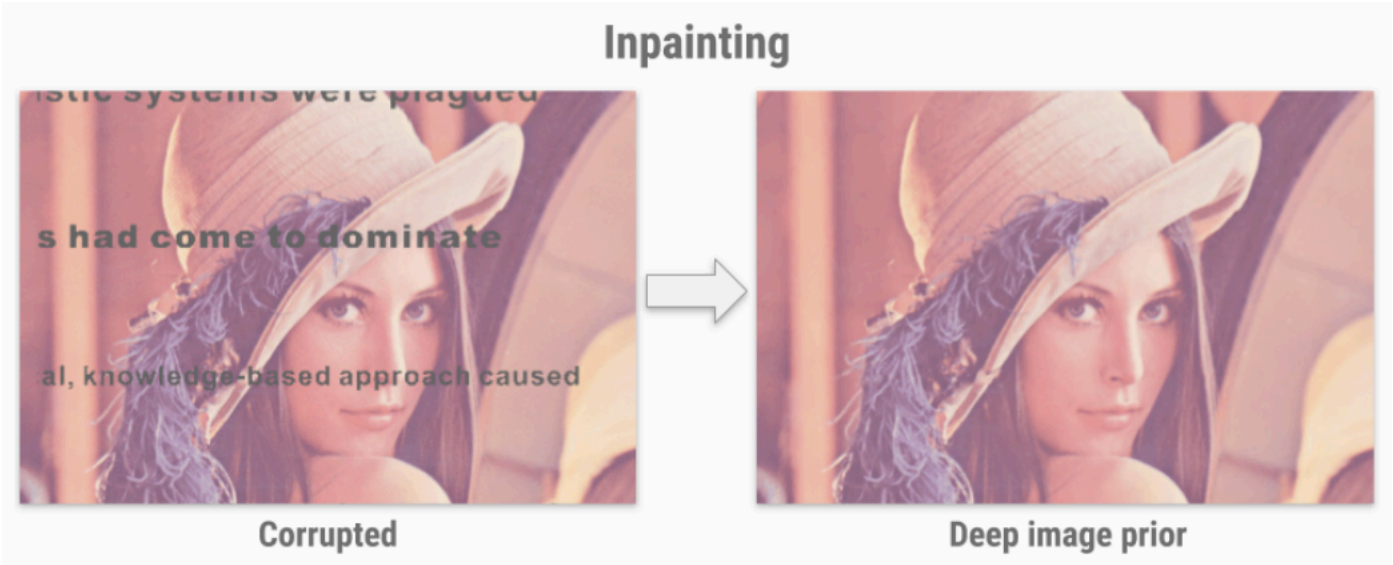
ConvNet обучается аппроксимировать 1 входное изображение

- x – Clean image
- x_0 – Corrupted image (observed)
- m – Binary mask

- **Denoising:** $E(x; x_0) = \|x - x_0\|^2$ **Needs early stopping!**
- **Inpainting:** $E(x; x_0) = \|(x - x_0) \cdot m\|^2$
- **Super-Res.:** $E(x; x_0) = \|d(x) - x_0\|^2$
- **Feature inv.:** $E(x; x_0) = \|\Phi(x) - \Phi(x_0)\|^2$

Примеры





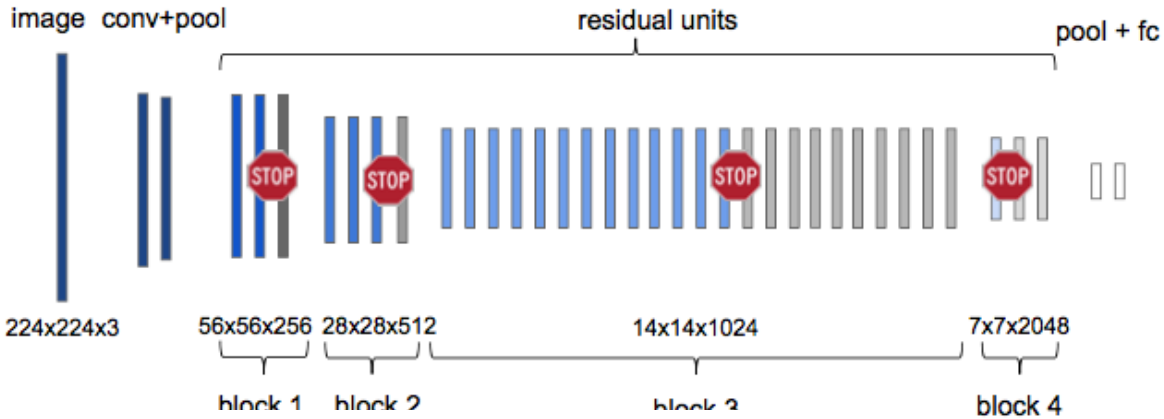
https://dmitryulyanov.github.io/deep_image_prior

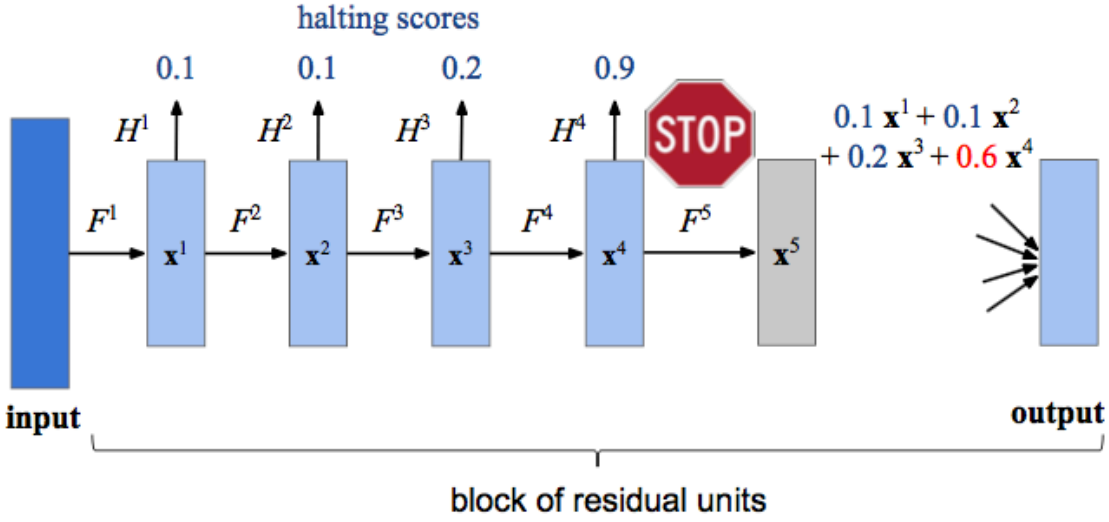
1. Computer Vision In the Wild
2. Generating Images
3. Efficient Convolutional Networks
4. Beyond Computer Vision

Spatially Adaptive Computation Time for Residual Networks (CVPR 2017)

Michael Figurnov (Higher School of Economics)

Adaptive Computation Time





$$\mathbf{x}^0 = \textbf{input},$$

$$\mathbf{x}^l = F^l(\mathbf{x}^{l-1}) = \mathbf{x}^{l-1} + f^l(\mathbf{x}^{l-1})$$

$$\textbf{output} = \mathbf{x}^L.$$

$$h^l = H^l(\mathbf{x}^l) = \sigma(W^l\,\mathrm{pool}(\mathbf{x}^l) + b^l)$$

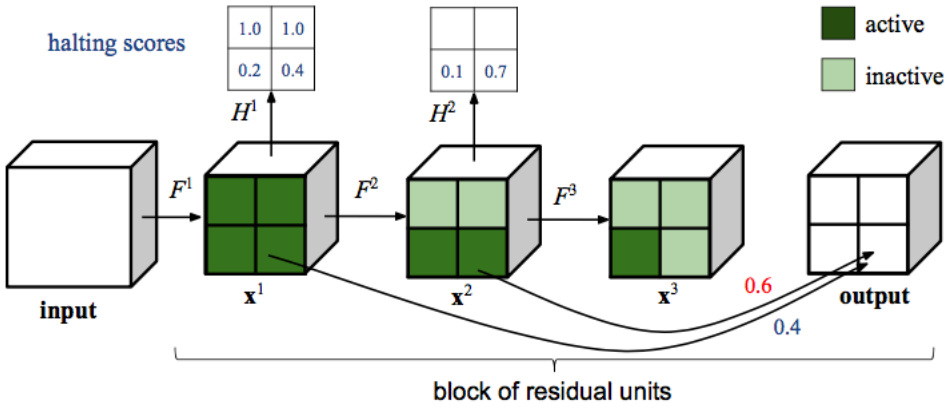
No. of residual units to evaluate:

$$N = \min\left\{n \in \{1 \dots L\} : \sum_{l=1}^{n} h^l \geq 1 - \varepsilon\right\}$$

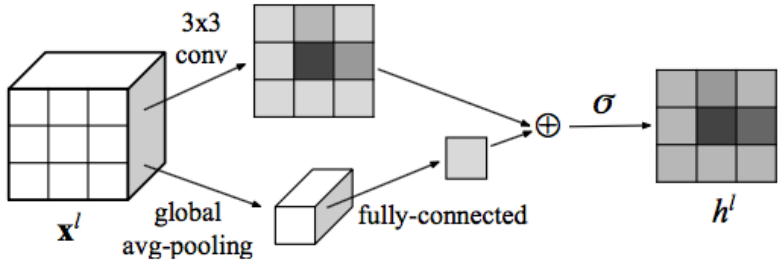
$$\text{output} = \sum_{l=1}^{L} p^l \mathbf{x}^l = \sum_{l=1}^{N} p^l \mathbf{x}^l$$

Spatially Adaptive Computation Time

Apply ACT to each spatial position of the block



Halting score

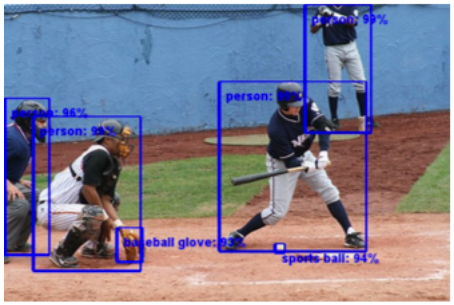


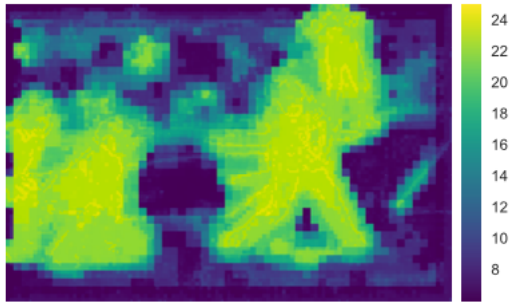
$$H^l(\mathbf{x}) = \sigma(\widetilde{W}^l * \mathbf{x} + W^l \,\mathrm{pool}(\mathbf{x}) + b^l)$$

COCO val set. Faster R-CNN with SACT

Feature extractor	FLOPs (%)	mAP @ [.5, .95] (%)
ResNet-101 [15]	100	27.2
ResNet-50 (our impl.)	46.6	25.56
SACT $\tau = 0.005$	**56.0** $\pm$ 8.5	**27.61**
SACT $\tau = 0.001$	72.4 $\pm$ 8.4	29.04
ResNet-101 (our impl.)	100	29.24

Heat map of computation time





Weight Parameterizations in Deep Neural Networks

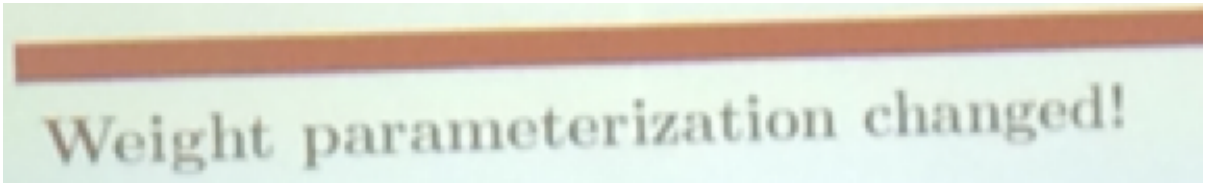
Sergey Zagoruyko (ENPC Paris)

По материалам статей:

- Wide Residual Network (arXiv:1605.07146)
- DiracNets: Training Very Deep Neural Networks Without Skip-Connections (arXiv: 1706.00388)
- Scaling the Scattering Transform: Deep Hybrid Networks, ICCV 2017

Что изменилось с начала эры deep learning?

- Оптимизация: все еще SGD with momentum
- Регуляризация: все еще L2 (weight decay)
- Функция потерь: все еще softmax для классификации
- **Архитектуры**: batch-normalization и skip-connections



DiracNet

parametrize conv weights as a residual of Dirac function, instead of adding explicit skip connection

In two-dimensional case, when convolution might be expressed as matrix multiplication, δ is simply an identity matrix I (or a Kronecker delta). We generalize this operator to the case of a convolutional layer, where an input tensor $x \in \mathbb{R}^{M,N_1,N_2,\ldots,N_L}$ (that consists of M channels of spatial dimensions $(N_1, N_2, \ldots, N_L)$) is convolved with a weight tensor $\hat{W} \in \mathbb{R}^{M,M,K_1,K_2,\ldots,K_L}$

$$y = \hat{W} \odot x,$$
$$\hat{W} = a\delta + W,$$

$$\delta(i, j, l_1, l_2, \ldots, l_L) = \begin{cases} 1 & \text{if } i = j \text{ and } l_m = \left\lfloor \frac{K_m}{2} \right\rfloor \text{ for } m = 1..L, \\ 0 & \text{otherwise;} \end{cases}$$

ResNet

$$y = x + \sigma(W \odot x)$$

DiracNet

$$y = \sigma\left((a\delta + W) \odot x\right) = \sigma\left(ax + W \odot x\right)$$

Loss $$L(W, a, b, x, z) = L_{orig}(W, a, b, x, z) + \gamma_W \|W\|_2^2 + \gamma_b \|b\|_2^2 + \gamma_a \|a\|_2^2$$

- Plain Dirac parameterized networks are able to train end-to-end with hundreds of layers, and achieve nearly state-of-the-art Wide ResNet (WRN) accuracy (CIFAR-10, CIFAR-100, ILSVRC2012);
- Plain deeper DiracNets with less than 50 layers need much less parameters than corresponding wider networks with the same accuracy; performance of deeper than 50 layer networks slowly decreases.

Кроме того, были исследованы симметричные матрицы весов

1. Computer Vision In the Wild
2. Generating Images
3. Efficient Convolutional Networks
4. Beyond Computer Vision

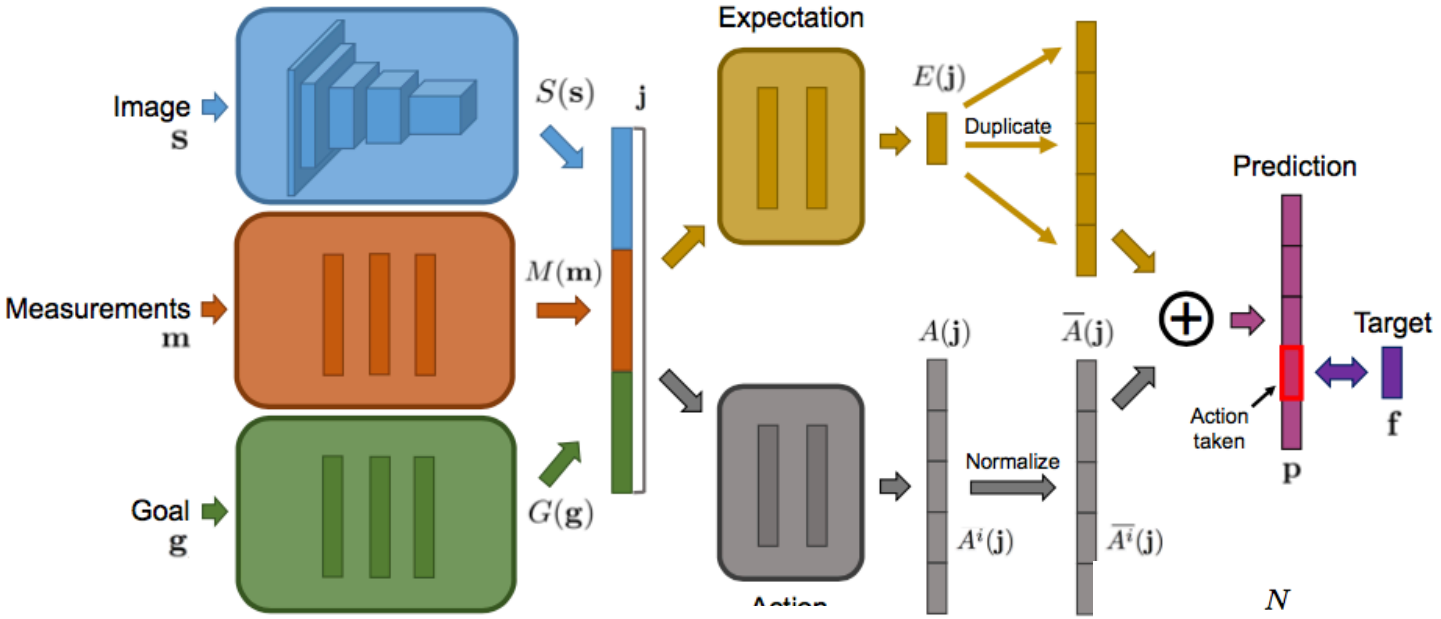
Issues with Deep Reinforcement Learning

Alexey Dosovitskiy (Intel Labs)

По материалам:

- Learning to Act by Predicting the Future, ICLR 2017

Predict future measurements $\mathbf{m}_t$ instead of future images $\mathbf{s}_t$.
The measurements can indicate the attitude, supply levels, and structural integrity in a physical system, or health, ammunition, and score in a computer game



$\mathbf{f} = \langle \mathbf{m}_{t+\tau_1} - \mathbf{m}_t, \ldots, \mathbf{m}_{t+\tau_n} - \mathbf{m}_t \rangle$ $u(\mathbf{f}; \mathbf{g}) = \mathbf{g}^\top \mathbf{f}$ - goal $\mathcal{L}(\theta) = \sum_{i=1}^{N} \|F(\mathbf{o}_i, a_i, \mathbf{g}_i; \theta) - \mathbf{f}_i\|^2$

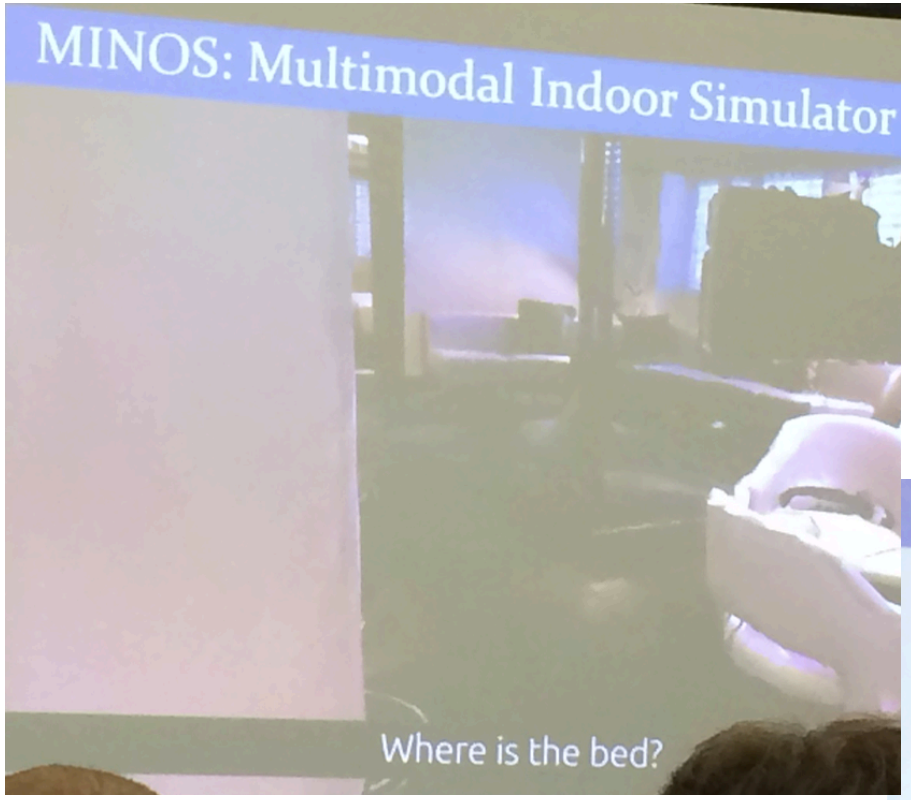
$a_t = \arg\max_{a \in \mathcal{A}} \mathbf{g}^\top F(\mathbf{o}_t, a, \mathbf{g}; \theta)$ - action selection at test time

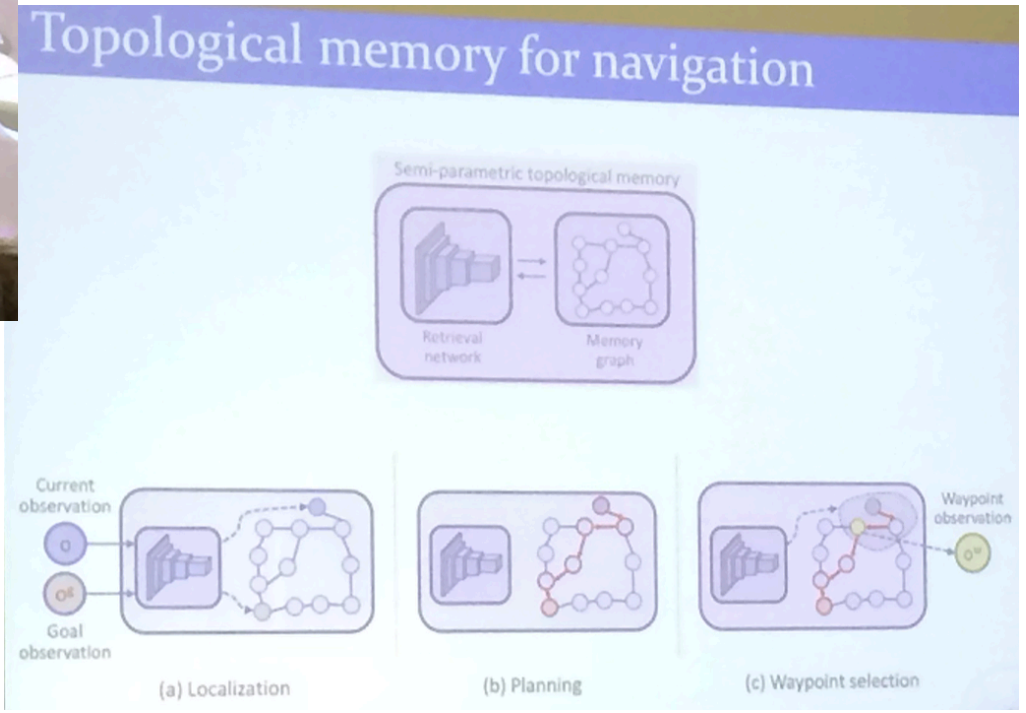
Примеры - VizDoom

Навигация в лабиринте

По материалам:

- MINOS: Multimodal Indoor Simulator (arXiv:1712.03931)



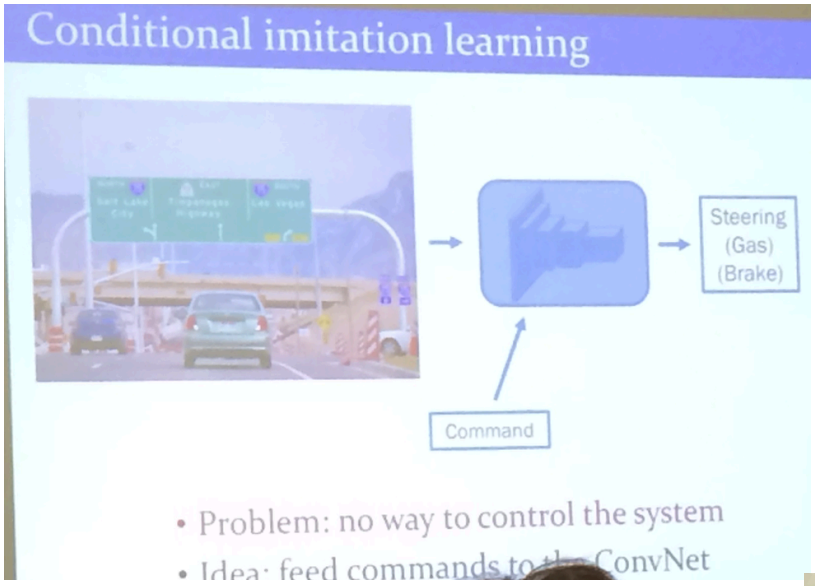


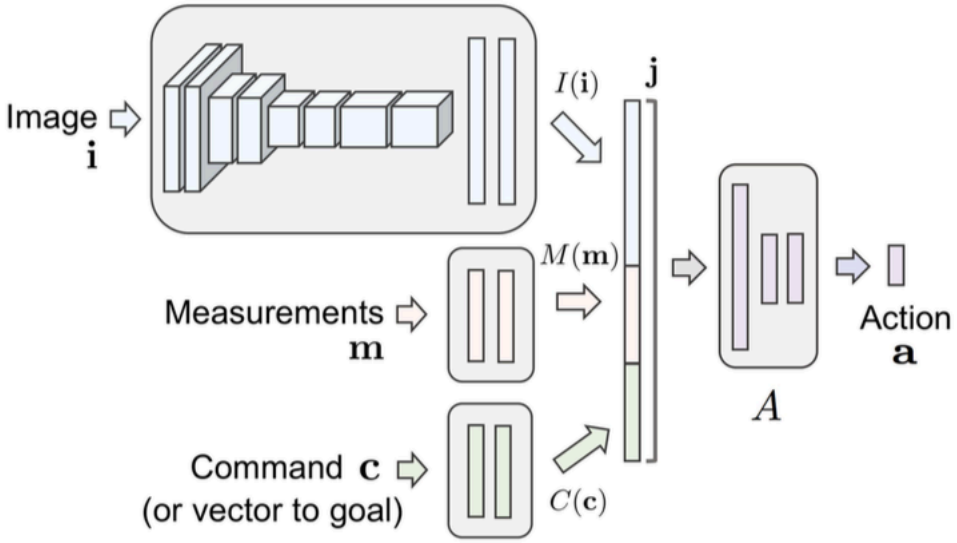
https://github.com/minosworld/minos

Управление авто/роботом

По материалам:

- CARLA: An open urban driving simulator (arXiv:1711.03938)
- End-to-end driving via conditional imitation learning (arXiv:1710.02410)





Simulator - http://carla.org

Fast Adaptation in Generative Models with Generative Matching Networks (Workshop track - ICLR 2017)

Sergey Bartunov (DeepMind)

$$p(\mathbf{x}|\mathbf{X},\boldsymbol{\theta}) = \int p(\mathbf{z}|\mathbf{X},\boldsymbol{\theta})p(\mathbf{x}|\mathbf{z},\mathbf{X},\boldsymbol{\theta})d\mathbf{z},$$

X – additional dataset used in testing (few-shot learning). **No labels!**

$$p(\mathbf{z}) = \mathcal{N}(z|0,I)$$

$$\log p(\mathbf{x}|\boldsymbol{\theta}) \geq \mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi}) = \mathbb{E}_q\left[\log p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta}) - \log q(\mathbf{z}|\mathbf{x},\boldsymbol{\phi})\right]$$

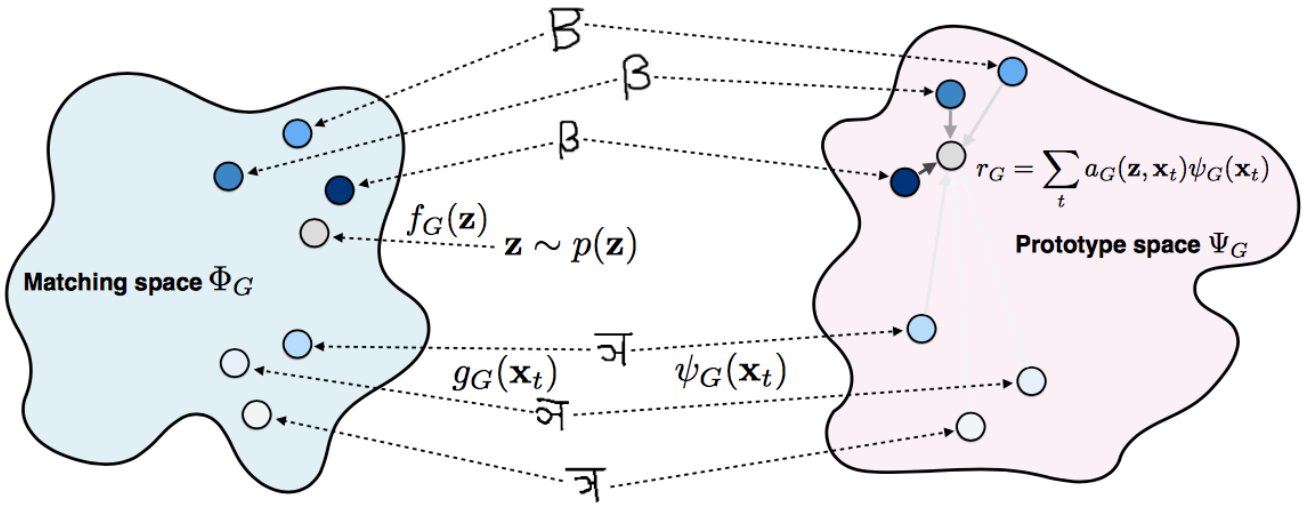
↑ **Lower bound** ↑ **Recognition model**

Training

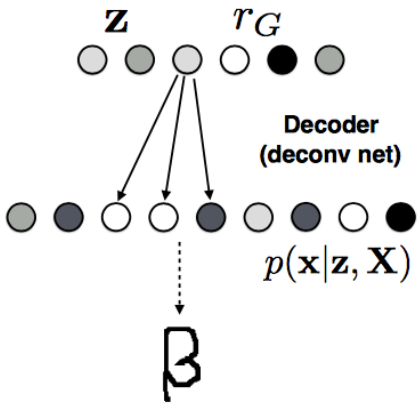
$$p(\mathbf{X}|\boldsymbol{\theta}) = \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{X}_{<t},\boldsymbol{\theta}), \quad \mathbf{X}_{<t} = \{\mathbf{x}_s\}_{s=1}^{t-1}$$

$$\mathcal{L}(\mathbf{X},\boldsymbol{\theta},\boldsymbol{\phi}) = \sum_{t=1}^{T} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{x}_t,\mathbf{X}_{<t},\boldsymbol{\phi})}[\log p(\mathbf{x}_t,\mathbf{z}_t|\mathbf{X}_{<t},\boldsymbol{\theta}) - \log q(\mathbf{z}_t|\mathbf{x}_t,\mathbf{X}_{<t},\boldsymbol{\phi})].$$

Generative Matching Networks



1. Computing attention kernel a_G



2. Aggregating prototypes

3. Decoding an observation

$$a_G(\mathbf{z}, \mathbf{x}_t) = \frac{\exp(\text{sim}(f_G(\mathbf{z}), g_G(\mathbf{x}_t)))}{\sum_{t'=1}^{T} \exp(\text{sim}(f_G(\mathbf{z}), g_G(\mathbf{x}_{t'})))}, \quad r_G = \sum_{t=1}^{T} a_G(\mathbf{z}, \mathbf{x}_t)\psi_G(\mathbf{x}_t)$$

Recognition model

$$q(\mathbf{z}|\mathbf{X}, \mathbf{x}, \boldsymbol{\phi}) = \mathcal{N}(\mathbf{z}|\mu(r_R), \Sigma(r_R))$$

$$a_R(\mathbf{x}, \mathbf{x}_t) = \frac{\exp(\text{sim}(f_R(\mathbf{x}), g_R(\mathbf{x}_t)))}{\sum_{t'=1}^{T} \exp(\text{sim}(f_R(\mathbf{x}), g_R(\mathbf{x}_{t'})))}, \quad r_R = \sum_{t=1}^{T} a_R(\mathbf{x}, \mathbf{x}_t)\psi_R(\mathbf{x}_t)$$

Примеры - Omniglot