Machine learning in numerical simulations

Ivan Oseledets

Skoltech
Skolkovo Institute of Science and Technology

Numerical simulation

- Take a model of a physical/chemical/biological system
- Construct discretization
- Solve resulting system of equations

Numerical simulation

- Model can be inaccurate or unavailable
- Discretization maybe unstable/difficult to construct
- Solver can be slow

Data-driven approaches

- Recover the model from experimental observations
- Approximate output quantities directly: surrogate models
- Optimize the given numerical simulator in a “black-box” way
- Approximate probability distributions instead of single solutions

Classical supervised machine learning

- Given input data, predict output data
- Setup a parametric class of models
- Solve a non-convex optimization problem

$$y_i \approx f(x_i, \theta)$$

Why ML is so easy to use

- Development of special frameworks (Tensorflow, Pytorch)
- Automatic differentiation: you never need to compute gradients (automatic is not symbolic!)
- Stochastic optimization

Deep Learning with PyTorch

Automatic differentiation: metatheorem (Baur-Strassen)

If we can evaluate f(x) in N operations, we can evaluate the gradient in less that cN operations.

You can differentiate any code!

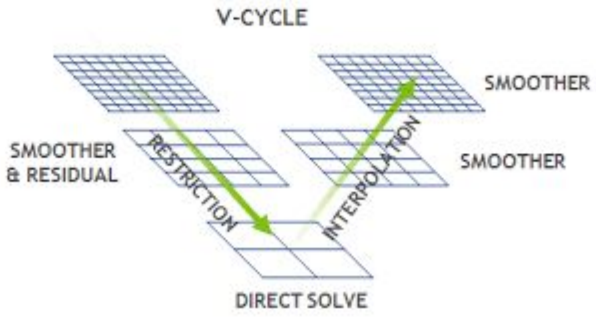
Automatic differentiation frameworks

- They existed for a long time (Fortran, C++), but difficult to use and not so efficient.
- Special frameworks for ML: Tensorflow, Pytorch (“metalanguages”)
- You write a code a little bit differently, but you get free GPU dispatch and gradients for free

What you can do: optimizing preconditioners

- Solve discretized PDE using multigrid method
- Need to define projection/restriction operators
- P, R come from prior knowledge

$$Ax = f$$



What you can do: optimizing preconditioners

- Minimize approximation of spectral radius
- Stochastic gradient method
- Autodiff

Spectral radius ρ

Grid size	Linear	AMG	DMG
7	0.169132	0.194611	**0.079188**
15	0.190049	0.208299	**0.086569**
31	0.195635	0.218042	**0.131717**
63	0.197055	0.259309	**0.143555**
127	0.197412	0.396509	**0.144278**
255	0.197501	0.377769	**0.147190**

Deep Multigrid: learning prolongation and restriction matrices

A Katrutsa, T Daulbaev, **I Oseledets** - arXiv preprint arXiv:1711.03825, 2017 - arxiv.org

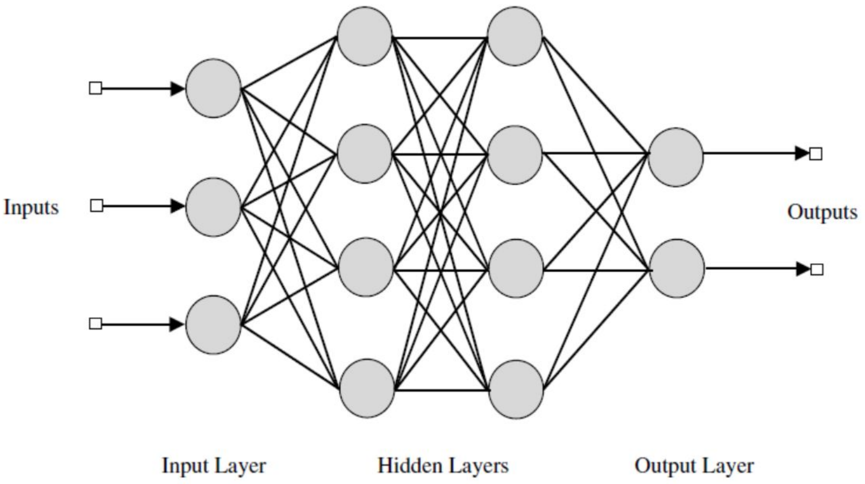
Approximation of multivariate functions

- In the example, the we optimized the known algorithm
- This is called differentiable programming
- What if we do not know the algorithm?
- We need a class of parametrized models

Neural networks

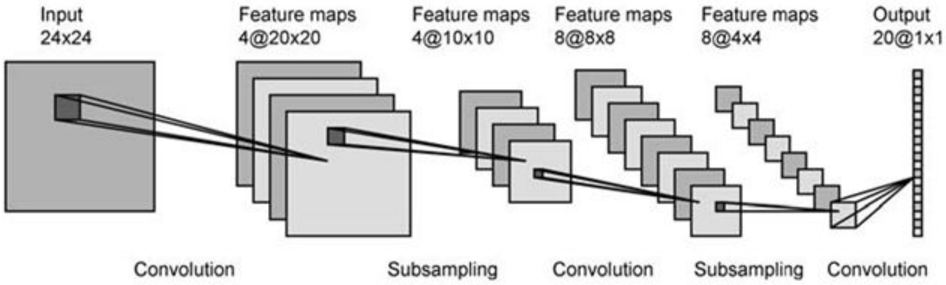
- Deep neural networks are extremely efficient for image, text and audio processing
- Feed-forward: superposition of linear/pointwise-nonlinear functions

$$y_{k+1} = f(W_k x_k + b), \quad y_{out} = y_N.$$



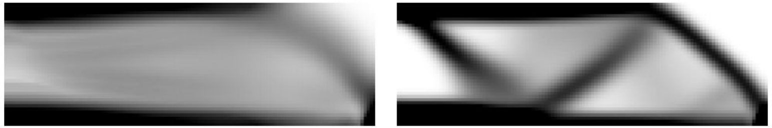
Convolutional neural networks

- Convolutional neural networks: Toeplitz matrices for W
- They are useful to work with piecewise-smooth objects (images)



Example: Topology optimization (Sosnovik, Oseledets)

Figure 1: The design domain, boundary conditions, and external load for the optimization of a half MBB beam.



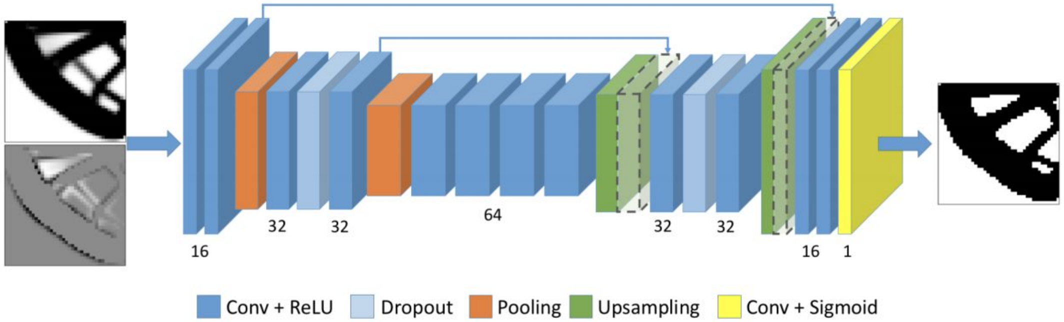
(a) Iteration 3 (b) Iteration 13

(c) Iteration 30 (d) Iteration 80

$$
\begin{cases}
\min\limits_{\boldsymbol{x}} & c(\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{x}) = \sum\limits_{j=1}^{N} E_j(x_j)\boldsymbol{u}_j^T \boldsymbol{k}_0 \boldsymbol{u}_j \\
\text{s.t.} & V(\boldsymbol{x})/V_0 = f_0 \\
& \boldsymbol{KU} = \boldsymbol{F} \\
& x_j \in \{0; 1\}, \quad j = 1 \dots N
\end{cases}
$$

Example: Topology optimization (Sosnovik, Oseledets)

- Idea: learn a mapping from current iterate and its gradient to the final solution
- Similar to image segmentation problem



Neural networks for topology optimization
I Sosnovik, I Oseledets - arXiv preprint arXiv:1709.09578, 2017 - arxiv.org

Example: Topology optimization (Sosnovik, Oseledets)

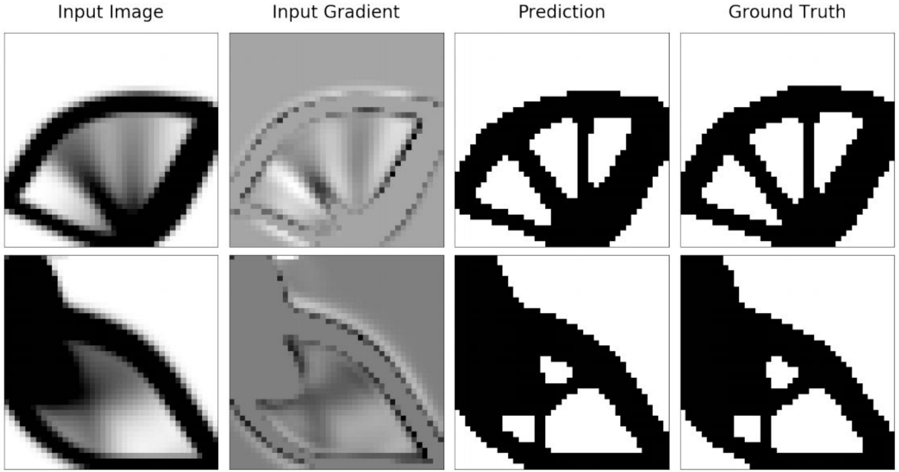


Figure 4: Top: SIMP is stopped after 8 iterations, binary accuracy 0.96, mean IoU 0.92; Bottom: solver is stopped after 5 iterations, binary accuracy 0.98, mean IoU 0.95.

	Iteration								
Method	5	10	15	20	30	40	50	60	80
Thresholding	92.9	95.4	96.5	97.1	97.7	98.1	98.4	98.6	98.9
CNN $P(5)$	**95.8**	97.3	97.7	97.9	98.2	98.4	98.5	98.6	98.7
CNN $P(10)$	95.4	**97.6**	**98.1**	98.4	98.7	98.9	99.0	99.0	99.0
CNN $P(30)$	92.7	96.3	97.8	**98.5**	**99.0**	**99.2**	**99.4**	**99.5**	99.6
CNN $U[1, 100]$	94.7	96.8	97.7	98.2	98.7	99.0	99.3	99.4	**99.6**

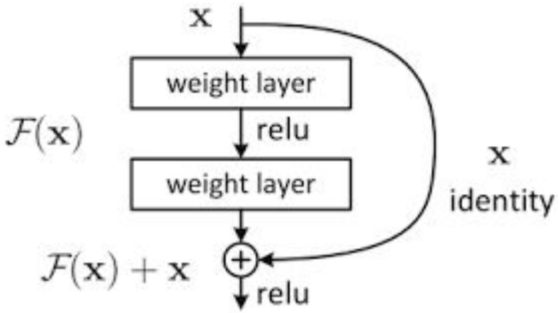
Idea for new architectures

- We can reutilize popular formats (what we have seen in the literature) to create new learnable architectures
- Example: hierarchical matrices for mapping input to the output.

Resnet: popular architecture

- Instead of learning
- We learn residual connection
- Motivated by multigrid!

$$y = f(x)$$
$$y = f(x) + x$$

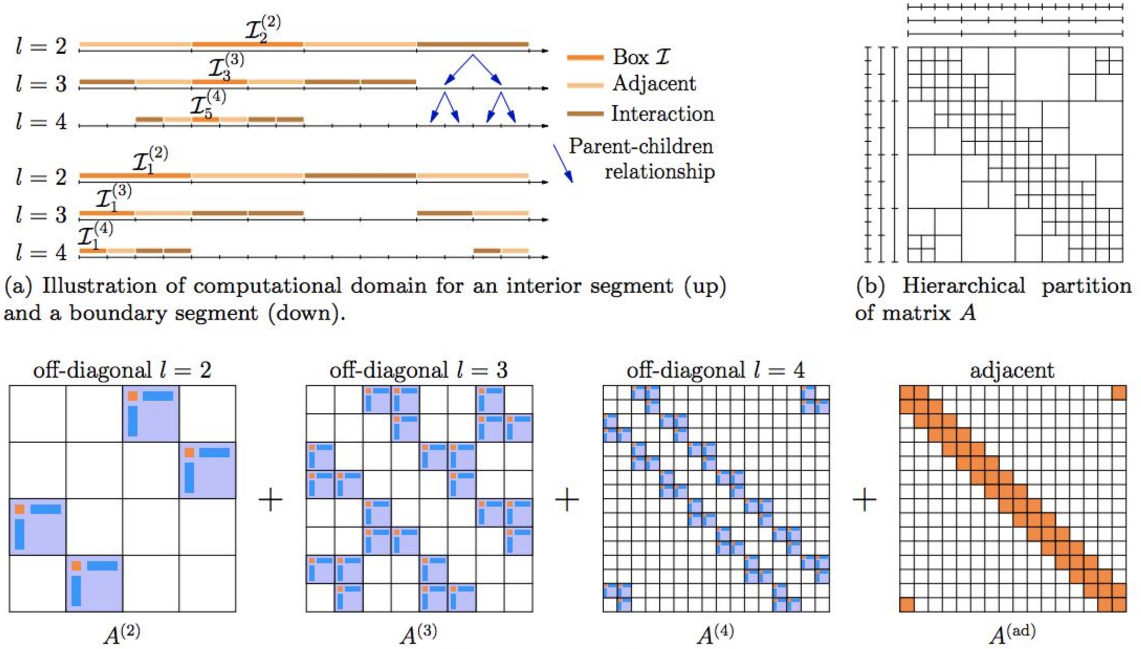


Multiscale neural network

- We can reutilize popular formats (what we have seen in the literature) to create new learnable architectures
- Example: hierarchical matrices for mapping input to the output.

A **multiscale neural network** based on hierarchical matrices
Y Fan, L Lin, L Ying, L Zepeda-Núnez - arXiv preprint arXiv:1807.01883, 2018 - arxiv.org

Multiscale neural network: idea



(a) Illustration of computational domain for an interior segment (up) and a boundary segment (down).

(b) Hierarchical partition of matrix A

(c) Decomposition of matrix A

Figure 1: Hierarchical partition of computational domain, its corresponding partition of matrix A and the decomposition of matrix A.

Multiscale neural network: idea

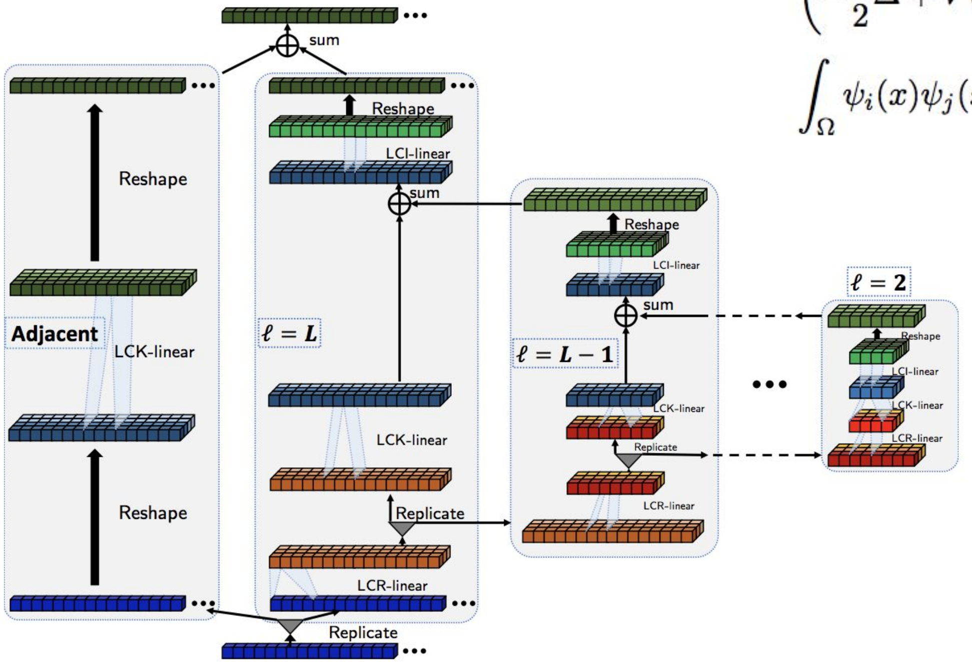


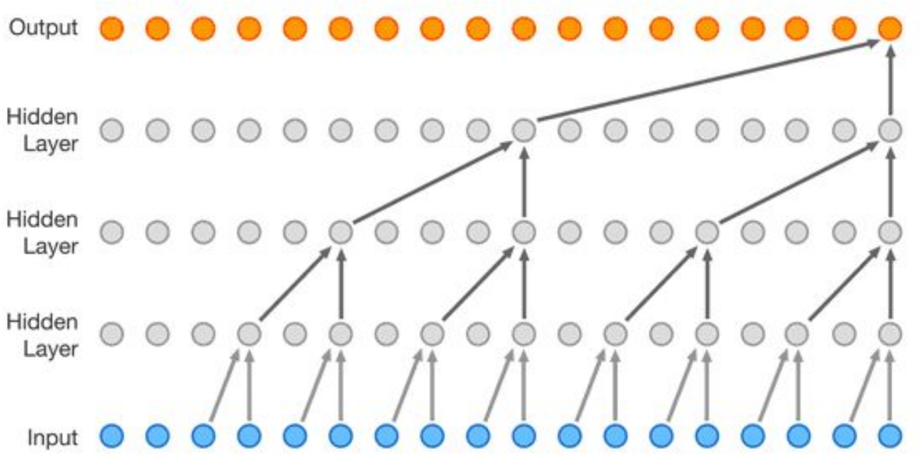
Figure 5: Neural network architecture for the matrix-vector multiplication of $\mathcal{H}^2$-matrices.

$$\left(-\frac{1}{2}\Delta + V(x)\right)\psi_i(x) = \varepsilon_i \psi_i(x), \ \ x \in \Omega = [-1,1)^d$$

$$\int_\Omega \psi_i(x)\psi_j(x)dx = \delta_{ij}, \quad \rho(x) = \sum_{i=1}^{n_e} |\psi_i(x)|^2,$$

Wavenet

- Wavelets as neural network architecture

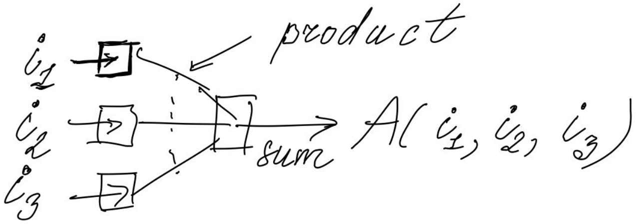


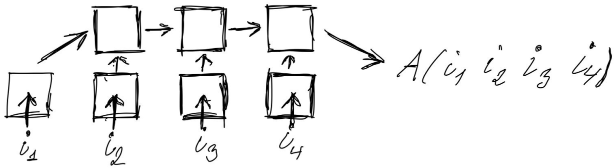
Tensor decompositions and neural networks

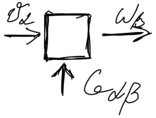
- Connections between tensor decompositions and special neural networks
- We can prove results on deep networks using tensor analysis

Expressive power of recurrent neural networks
V **Khrulkov**, A Novikov, I **Oseledets** - ICLR 2018
On the expressive power of deep learning: A tensor analysis
N **Cohen**, O Sharir, A **Shashua** - Conference on Learning Theory, 2016 - jmlr.org

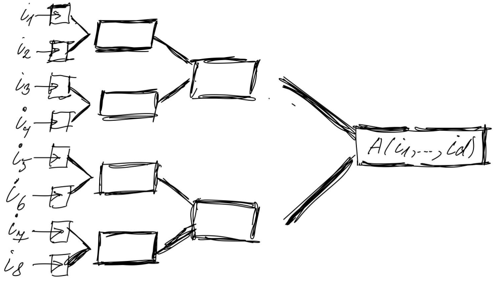
Tensor decompositions and neural networks







$$w_\beta = \sum_\alpha v_\alpha G_{\alpha\beta}$$



$$u_\gamma = \sum_{\alpha\beta} M_{\alpha\beta\gamma} v_\alpha w_\beta$$

Unsupervised learning

- All the previous approaches deal with supervised learning, when the answer is known
- Unsupervised learning is becoming key technique for manifold learning and dimensionality reduction, and also uncertainty quantification

Learning probability distributions

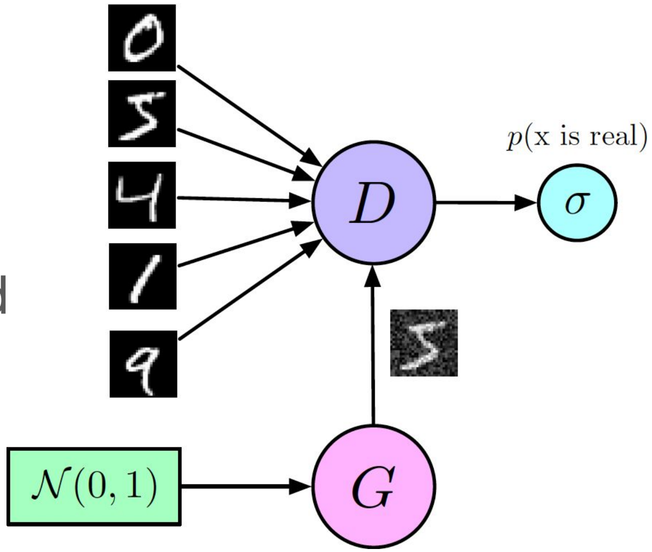
- Given data points $x_1, \ldots, x_N$ that are sampled from a probability distribution p, learn this probability distribution
- Key problem in UQ, inverse problems, data assimilation
- Generative adversarial network (GAN) gaining popularity

Generative adversarial networks

- Proposed by Goodfellow et. al in 2014
- Idea is to approximate this distribution as a parametrized map from a known distribution
- An additional function (discriminator) is used to distinguish between fake and real data
- They learn an adversarial game

Reminder

- Game theoretic approach
- **Generator** learns to mimic the target distribution by generating **samples**
- **Discriminator** learns to distinguish real and fake data



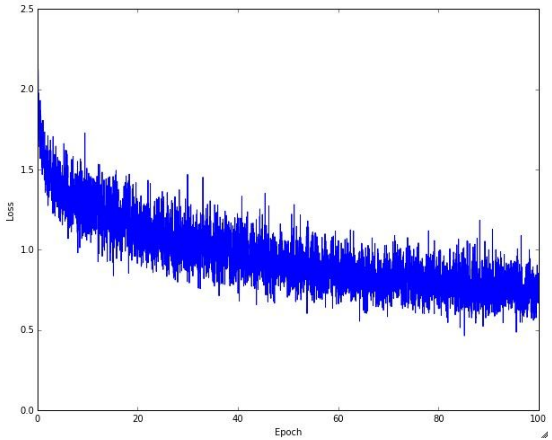
Deep Prior (Ulyanov, Lempitsky, Vedaldi)

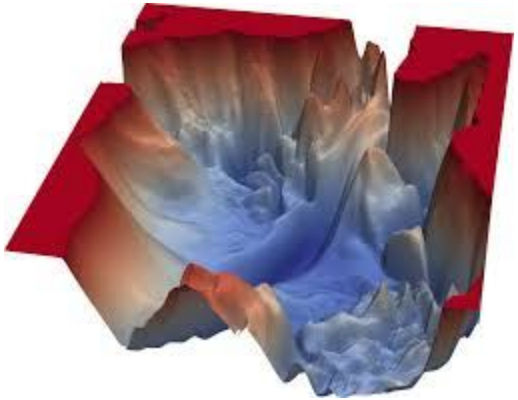
- Overparametrization is not always the issue
- We assume that image is generate by a CNN
- We learn the parameters of CNN by minimizing the cost functional (i.e., for denoising) for 1 image!

$$\min_{\theta} E(f_{\theta}(z), x_0)$$

Problems

- Learning can be slow
- Sensitive to hyperparameters (may not converge in many cases)
- Smoothness, monotonicity, etc. are not guaranteed for neural-network based models
- Loss surfaces can be really complicated





Adversarial perturbations

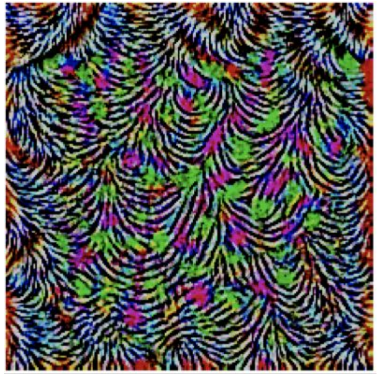
- **Adversarial perturbations** easily fool many state of the art networks
- Adding a perturbation of a small norm can force misclassification

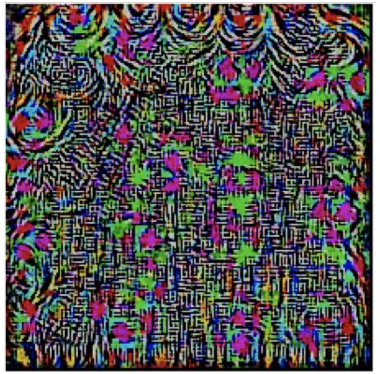
$$\arg\max p(y|x;\theta) \neq \arg\max p(y|x+\varepsilon;\theta)$$

This work is supported by Ministry of Education and Science of the Russian Federation (grant 14.756.31.0001)

Universal adversarial perturbations

- Mosaavi et al (2017) proposed **universal perturbations**: adding a single noise image allows one to fool the network in many (~70%) cases
- They were also shown to generalize **across networks** really well





Universal adversarial perturbations, Moosavi et al, CVPR 2017

Our results (Khrulkov, Oseledets, CVPR 2018)

- Interpretable easy algorithm
- Relatively fast - only few minutes to construct a perturbation
- We attack **low-level features**

	VGG-16	VGG-19	ResNet50
VGG-16	0.52	**0.60**	0.39
VGG-19	0.48	**0.60**	0.38
ResNet50	0.41	**0.47**	0.44