Understanding models via visualizations, attribution, and semantic identification

Dr Andrea Vedaldi

CVPR 2018 Tutorial on
Interpretative Machine Learning for Computer Vision

Salt Lake City, USA
June 18, 2018
https://interpretablevision.github.io
Explaining deep neural networks

Peeking inside the black box

What does a net do?
- What concepts can it recognise?
- Spurious correlations?
- Limitations?

How does it do it?
- Template matching?
- Compositionality?
- Spatial reasoning?

How does it learn it?
- Generalization?
- Optimisation?
Explaining deep neural networks

Peeking inside the black box

What concepts can it recognise?
- Spurious correlations?
- Limitations?

What does a net do?
Deep networks as encoders

Each subnetwork $\Phi$ maps an image $x$ to a code $y$
Each subnetwork $\Phi$ maps an image $x$ to a code $y$.
Generating iconic examples

Attribution

Semantic identification
Generating iconic examples

Attribution

Semantic identification
Find out “how much” of the image $x$ can be reconstructed from the code $y$.

**Images**

$\mathcal{X} = \mathbb{R}^m$

**Codes**

$\mathcal{Y} = \mathbb{R}^n$
How much information about $x$ does $y$ contain?

Find out “how much” of the image $x$ can be **reconstructed from the code $y$**

Images

$\mathcal{X} = \mathbb{R}^m$

Codes

$\mathcal{Y} = \mathbb{R}^n$

Reconstructions are not unique; rather, they form an **equivalence class** of images that **are the same for the network**
Starting from random noise, “match” the code via **direct optimization**

**Images**
\[ \mathcal{X} = \mathbb{R}^m \]

**Codes**
\[ \mathcal{Y} = \mathbb{R}^n \]

\[
\min_x \| \Phi(x) - \Phi(x_0) \|^2
\]

Pre-image

Images

Codes

\[ x_0 \]

\[ x_1 \]

\[ y \]
Neural nets are “meaningless” outside their training domain. Hence, reconstructions should be constrained to be natural images.

\[ \mathcal{X} = \mathbb{R}^m \]
\[ \mathcal{X}_n = \text{natural} \]
\[ \mathcal{X}_{pn} = \text{pseudo-natural} \]

\[ \mathcal{Y} = \mathbb{R}^n \]

\[ \Phi^{-1} \]
Several possible implementations

**Regularized energy**

\[
\min_x \| \Phi(x) - \Phi(x_0) \|^2 + \mathcal{R}(x)
\]

For example TV-norm

**Constrained optimization**

\[
\min_{x \in \mathcal{X}_{pm}} \| \Phi(x) - \Phi(x_0) \|^2
\]

For example Deep Image Prior

**Posterior probability**

\[
p(x \mid y) \sim \delta(\Phi(x) - y) \cdot p(x)
\]

For example Plug & Play gen. nets

**Understanding deep image representations by inverting them**

Mahendran Vedaldi, CVPR, 2015

**Deep image prior**

Ulyanov Vedaldi Lempistky, CVPR, 2018

**Plug & play generative networks: Conditional iterative generation of images in latent space**

Nguyen, Yosinksi, Bengio, Dosovitskiy, Clune, CVPR, 2017
The prior is the **structure** of the CNN

The network provides a **parametrization** of images:

\[
x = \Psi(w; z_0)
\]

**w** is **not learned** but used as a **free image parameter**
A parameterization that offers high-impedance to noise

Consider the reconstruction problem

- $x$ target image
- $w$ net parameters, randomly initialized
- use SGD to minimize the $L^2$ loss

$$\min_w \|x - \Phi(w)\|^2$$

The convergence speed is proportional to how “natural” the image looks

Deep image prior, Ulyanov Lempitsky Vedaldi, CVPR 2018
For **inpainting** we only reconstruct the visible pixels, implicitly infer the others

\[
\min_w \| \mathbf{m} \odot (x - \Phi(w)) \|^2
\]

**Deep image prior**, Ulyanov Lempitsky Vedaldi, CVPR 2018
Deep image prior: inpainting

Masked image
Deep image prior: inpainting

Deep-image prior completion
The inverter is only given

- The code $y_0$
- The network to invert $\Phi$
- The structure (not the parameters) of the generator $\psi$

Inverting codes via the deep image prior

\[
\min_{w} \| \Phi(\Psi(w)) - \Phi(x_0) \|^2
\]
Inverting a Deep CNN

AlexNet
[Krizhevsky et al. 2012]
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN
Inverting a Deep CNN

Original image
Inverting a Deep CNN

Original image
Inverting a Deep CNN

conv 1  conv 2  conv 3  conv 4  conv 5  fc 6  fc 7  fc 8

Original image

39
Inverting a Deep CNN

Original Image

Conv 1

Conv 2

Conv 3

Conv 4

Conv 5

FC 6

FC 7

FC 8
Decoding AlexNet trained on ImageNet

Is the code semantic or visual?

fc8 is a 1000-dimensional **class score vector**... or is it?
4.1 Data Augmentation

Describe the two primary ways in which we combat overfitting. The easiest and most common method to reduce overfitting on image data is to artificially enlarge the dataset using label-preserving transformations (e.g., [25, 4, 5]). We employ two distinct forms of data augmentation, both of which allow transformed images to be produced from the original training set by a factor of 2048, though the resulting training examples are, of course, highly inter-correlated.

To each training image, we add multiples of the found principal components, forcing us to use much smaller networks. At test time, the network makes a prediction by extracting the (flattened) output of the first convolutional layer and filters it with 256 kernels of size \( 3 \times 3 \times 5 \). The fully-connected layers have 4096 neurons each. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the image is divided into ten patches (the four corner patches and the center patch) as well as their horizontal reflections. We do this by extracting random 224–224 patches (and their horizontal reflections) from the ImageNet training set.

The second form of data augmentation consists of altering the intensities of the RGB channels in layer on the ten patches.

The first form of data augmentation consists of generating image translations and horizontal reflections. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers.

As an example of how to maximize specific neuron activation, consider the following equation:

\[
\min_w - \langle e_k, \Phi(\Psi(w)) \rangle
\]
Deep Quiz

https://goo.gl/jURsCP
Visualization via direct optimization

Reading list

**Visualizing higher-layer features of a deep network.** Erhan, Bengio, Courville, U Montreal, 2009

**Visualization via direct optimization**

**Visualizing and understanding convolutional networks**

**Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps**
Simonyan Zisserman Vedaldi, ICLR, 2104

**Understanding deep image representations by inverting them**
Mahendran Vedaldi, CVPR, 2015

**Google “inceptionism”**
Mordvintsev et al. 2015

**Understanding neural networks through deep visualisation**
Yosinksi et al. ICMLW, 2015

**Plug & play generative networks: Conditional iterative generation of images in latent space**
Nguyen, Yosinksi, Bengio, Dosovtskiy, Clune, CVPR, 2017

**Deep image prior**
Ulyanov Vedaldi Lempistky, CVPR, 2018

**Activation maximisation** for class neurons

Activation maximization using **empirical prior, deconvnet**

Activation maximization and **saliency**

**Inversion** at different depths, **natural image prior**

Activation maximisation for **intermediate neurons**

Improved regularizers, artistic applications (deep dreams)

More regularizers, toolbox

Strong learned regularizer, sample **diversity**

Advanced “data agnostic” regularization
The choice of inversion prior has a strong effect

AlexNet Visualizations

Deep Image Prior

TV-Norm Prior
The inverter is only given:

- The code $y_0$
- The network to invert $\Phi$
- The structure (not the parameters) of the generator $\psi$

Using the deep image prior is just one option…

$\min_w \|\Phi(\Psi(w)) - \Phi(x_0)\|^2$
The inverter is now given:

- The code $y_0$
- A large training set of images to learn a generator from

$$\min_{\Psi} \frac{1}{N} \sum_{i=1}^{N} \| \Psi(\Phi(x_i)) - x_i \|^2$$
Train a strong prior from examples

Visualizing via learning a generator

Inverting convolutional networks with convolutional networks
Dosovitskiy Brox, CVPR, 2016

Synthesizing the preferred inputs for neurons in neural networks via deep generator networks
Nguyen, Dosovitskiy, Yosinski, Brox, Clune, NIPS, 2016

Generating images with perceptual similarity metrics based on deep networks
Dosovitskiy Brox, NIPS, 2016

Plug & play generative networks: Conditional iterative generation of images in latent space
Nguyen, Yosinski, Bengio, Dosovitskiy, Clune, CVPR, 2017
**Diagnostic vs aesthetic value**

**Our goal:** diagnose a given **discriminator network** $\Phi$

But inversions **also** reflect the chosen “natural image” **prior** $p(x)$

$p(x) = \begin{align*}
\text{generator net} & \quad \text{structure only} \\
\text{Plug & Play Gen. Net.} & \quad \text{generator net trained GAN-like from ImageNet} \\
\text{Empirical prior} & \quad \text{ImageNet validation set (empirical distribution)}
\end{align*}$

Illustrates the **model** $\Phi$

Illustrates the **prior** $p(x)$
Reviews and interfaces

If you want to dig further

The building blocks of interpretability
Olah, Satyanarayan, Johnson, Carter, Schubert, Ye, Mordvintsev

Understanding neural networks through deep visualisation
Yosinski et al. ICMLW, 2015
Generating iconic examples

Attribution

Semantic identification
Find what parts of an image are salient for a deep network.
Saliency: Backpropagation

Sensitivity analysis of target neuron w.r.t. input pixels

\[ J = \frac{d\Phi(x)}{dx} \]  

The “salient” pixels usually light up

Three popular methods

**Deconvolution**
**Visualizing and understanding convolutional networks**
Zeiler Fergus, ECCV, 2014

**Gradient** (backpropagation)
**Deep inside convolutional networks: Visualising image classification models and saliency maps**
Simonyan, Vedaldi, Zisserman, ICLR, 2014

**Guided backpropagation**
**Striving for simplicity: The all convolutional net**
Springenberg, Dosovitskiy, Brox, Riedmiller, ICLR, 2015
The only difference is in how ReLU is reversed!

Salient deconvolutional networks, Mahendran Vedaldi, ECCV, 2016
... the only difference is in how ReLU is reversed

See comparison in [Salient deconvolutional networks, Mahendran Vedaldi, ECCV, 2016]
The saliency of neurons at the same location is about the same.

Lack of channel specificity

Saliency for:
- maximally activated neuron
- random neuron
- minimally activated neuron
Better **channel specificity** can be achieved by backpropagating only a few layers.

CAM and Grad-CAM

---

**Learning deep features for discriminative localization**
Zhou, Khosla, Lapedriza, Oliva, Torralba, CVPR, 2016

**Grad-CAM: Visual explanations from deep networks via gradient-based localization**
Selvaraju, Cogswell, Das, Vedantam, Parikh, Batra, ICCV, 2017
Define some other rules to back-propagate the “relevance” of activations.

On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation
Bach, Binder, Montavon, Klauschen, Müller. PLOS one, 2015

Top-down neural attention by excitation backprop
Zhang, Lin, Brandt, Shen, Sclaroff, ECCV, 2016
A simple example: gradients modulated by the forward activations

Due to chaining and cancelations we get that at any level $m$ relevance is the modulated gradient

$$r_m^T = \frac{dx_m}{dx_m^T} \cdot \text{diag}(x_m)$$

Actual relevance and excitation backpropagation use more sophisticated rules

See extensive tutorial this morning!
Methods have been defined by specifying a “backpropagation formula”

But what does the result of this computation actually mean?
The meaning of saliency

It “looks good”

Deconvolution
- Sharp
- Poor spatial selectivity

Gradient
- Blurry
- Good spatial selectivity

Guided Backprop
- Sharp
- Good spatial sensitivity

Reminder: they all still have poor channel selectivity
It generates “semantic” heat maps

A good correlation means that:
1) the **diagnosed model** “understand” the location of objects and
2) the **saliency method** can diagnose this fact

**Measure**: degree of correlation between the saliency results and **ground truth semantic labels** (e.g. objects).

Drawbacks:
1) failure of localization **confound** limitations of the model and the saliency method
2) difficult to say which is which since the saliency formulas are largely **heuristics**
The meaning of saliency

Gradients prove a local approximation of the model

\[ \Phi(x) \approx \left( \frac{d\Phi}{dx} , x - x_0 \right) + \Phi(x_0) \]

The gradient can be directly interpreted as a local linear approximation of the model.

However, all other saliency propagation rules do not have simple interpretations such as this.
Towards a formal approach to explanations

Study how $\Phi(x)$ changes up to perturbations $\pi(x)$ of the input $x$

**Perturbation** should be *meaningful* (interpretable). E.g:
- Injecting noise
- Rotating or translating the image
- Erasing parts of the image

The *representation* may
- Be *invariant* (stay the same)
- Be *equivariant* (respond predictably)

The *analysis* may be
- Local around $x$ and $\pi$
- For a distribution $p(x)$ and a fixed $\pi$
- For a distribution $p(\pi)$ and a fixed $x$
- …
Meaningful perturbation analysis: saliency

Saliency via eliding objects

We seek the “smallest elision” that maximally changes the neuron activation.
Meaningful perturbation analysis: saliency

Searching for the smallest elision mask via optimization

\[ E(m) = \Phi_c((1 - m) \odot x + m \odot (g_\sigma * x)) + \lambda \|m\|_1, \quad 0 \leq m \leq 1 \]

The energy rewards lowering the class score while using a small mask

Use SGD to optimize over the mask \( m \)
Looking beyond neural network artifacts

Neural networks are **fragile to adversarial perturbations**

Adversarial perturbations attract gradient descent

Looking beyond neural network artifacts

Adversarial elision

Meaningful elision

improbable in nature

likely in nature

Regularization can help finding more meaningful perturbations

Examples: simplify the mask, look for the average effect of a pool of similar masks
Crisp regions
Similar to gradient, meaning is “obvious” by definition

Interpretable explanations of black boxes by meaningful perturbation, Fong Vedaldi, CVPR, 2017
What is salient may not be meaningful

Example: the hot chocolate is recognized via the spoon and the truck vs the license plate
Let $y = \Phi(x)$ be the label predicted for image $x$ by the deep net.

Empirically, we can find tiny perturbations $x + \delta$ that change $y$ arbitrarily!

\[ \delta^* = \arg\min_\delta \|y_{\text{arbitrary}} - \Phi(x + \delta)\| \]

\[ \|\delta\| < \epsilon \]

**Intriguing properties of neural networks**


---

**CNN fragility**

Easily fooled by adversarial examples
Adversarial examples can be successfully “injected” in real life

Adversarial glasses fooling face recognition

Adversarial stickers fooling sign recognition


Adversarial defence

Method: recognize genuine vs adversarial images by learning a classifier on top of the saliency maps

(Illustrative of properties of saliency, not really a recommended defense strategy!)
How is a representation affected by an image warp?

**Short answer:** warping image usually reduces to sparse linear tf in feature space.

**Long answer:** Understanding image representations by measuring their equivariance and equivalence. Lenc Vedaldi. CVPR 2015 & IJCV 2018
Are different neural networks “the same”?

Short answer: there generally are corresponding features in different networks (up to 1x1 linear tfs).

Long answer: Understanding image representations by measuring their equivariance and equivalence. Lenc Vedaldi. CVPR 2015 & IJCV 2018
Generating iconic examples

Attribution

Semantic identification
Each neuron / concept “activates” for a subset of natural images (patches)

Assume that neurons have binary activation and that concepts apply deterministically

\[ \text{concept}_i = \{ x \in \mathcal{X} : \text{concept}_i(x) = \text{true} \} \]

\[ \text{neuron}_k = \{ x \in \mathcal{X} : \Phi_k(x) = 1 \} \]
Each neuron / concept “activates” for a subset of natural images (patches)

Assume that neurons have binary activation and that concepts apply deterministically

\[ \text{concept}_i = \{ x \in \mathcal{X} : \text{concept}_i(x) = \text{true} \} \]

\[ \text{neuron}_k = \{ x \in \mathcal{X} : \Phi_k(x) = 1 \} \]

Questions:
- Do neurons and concepts correspond one-to-one?
- How many neurons are required to express a concept?
- How many concepts are required to express a neuron?
<table>
<thead>
<tr>
<th>Reading list</th>
<th>Correlates filters with ImageNet patches</th>
<th>Identifies the semantics of some convolutional filters</th>
<th>BRODEN, fine-grained semantic of individual filters</th>
<th>Understand training task performance</th>
<th>Learn linear predictor for diagnostics</th>
<th>Relation between interpretability and classification performance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object-centric representation learning from unlabeled videos.</strong></td>
<td>Gao, Jayaraman, Grauman, ACCV, 2016</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Places: An image database for deep scene understanding</strong></td>
<td>Zhou, Khosla, Lapedriza, Torralba, Oliva. PAMI, 2016</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Understanding intermediate layers using linear probes</strong></td>
<td>Alain Bengio. Proc. ICLR Workshop, 2017</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Revisiting the importance of individual units via ablation</strong></td>
<td>Zhou He Bau Torraba, arXiv 2018</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Concepts as a structured abstraction

Beyond just responding to images

Concepts form a structured space. For example, WordNet induces an is-a hierarchy:

![Diagram of WordNet hierarchy](image)
Net2vec associates a linear concept space to a network

\[ \Phi \]

Train a classifier from a network slice \( \Phi \)

\[ \text{is-a-wing}(x) \approx \langle w_{\text{wing}}, \Phi(x) \rangle \]

Space of natural images

Space of classifier functions

Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. Fong Vedaldi. Proc. CVPR, 2018
Net2vec associates a linear concept space to a network

Concepts as a structured abstraction

Train a classifier from a network slice $\Phi$

$$\text{is-a-wing}(x) \approx \langle w_{\text{wing}}, \Phi(x) \rangle$$

$$\text{is-swirly}(x) \approx \langle w_{\text{swirly}}, \Phi(x) \rangle$$

Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. Fong Vedaldi. Proc. CVPR, 2018
Net2Vec associates a linear concept space to a network

Train a classifier from a network slice $\Phi$

\[
is-a-wing(x) \approx \langle w_{\text{wing}}, \Phi(x) \rangle
\]

\[
is-swirly(x) \approx \langle w_{\text{swirly}}, \Phi(x) \rangle
\]

\[
is-furry(x) \approx \langle w_{\text{furry}}, \Phi(x) \rangle
\]

Net2Vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. Fong Vedaldi. Proc. CVPR, 2018
Thousands of images annotated with hundreds of concepts, often densely

<table>
<thead>
<tr>
<th>Image-level Annotations</th>
<th>Pixel-level Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>street (scene)</td>
<td>flower (object)</td>
</tr>
<tr>
<td>swirly (texture)</td>
<td>headboard (part)</td>
</tr>
<tr>
<td></td>
<td>pink (color)</td>
</tr>
<tr>
<td></td>
<td>metal (material)</td>
</tr>
</tbody>
</table>

Network dissection: Quantifying interpretability of deep visual representations
Net2vec: building concept embeddings

Vecs: pixel-wise linear predictor of a concept

Threshold activations

“Dog” vec embedding

Channel-wise sum

Segmentation mask

99.5% quantile [Bau et al., 2017]

IoU = .77
Net2vec: building concept embeddings

Sparse vets (only a few neurons)

Threshold activations

“Dog” vec embedding

Channel-wise sum

Subset selection follows [Agrawal et al., 2014]

Segmentation mask

IoU = .77
Net2vec: building concept embeddings

Singleton vests (only one neuron)

Threshold activations

“Dog” vec embedding

Channel-wise sum

Segmentation mask

\[ \sum \]

\[ \times w_1 \]

\[ \times w_2 \]

\[ \text{Best filter predictor} \]

~ [Bau et al., 2017]

IoU = .77
**Observation:** using more than one channel performs much better for most concepts

Individual neuron do not “isolate” concepts
Number of channels per concept

Depends strongly on the concept, even at similar level of abstractions

Number of Top Filters Used ($F$)

Set IoU

- **airplane**: 64 channels
- **person**: 8 channels
Concepts may need to be combined to explain neurons

Neuron may correspond to concept combinations, or to “unknown” concepts.

Sheep ($\text{IoU}_{\text{set}} = .21$)

Horse ($\text{IoU}_{\text{set}} = .21$)

Cow ($\text{IoU}_{\text{set}} = .20$)

AlexNet conv5 66 is highly selective for multiple farm animals.
Net2vec associates a linear concept space to a network

The representation induces a similarity between concepts

\[ [K_\Phi]_{ij} = \langle w_{\text{concept}_i}, w_{\text{concept}_j} \rangle \]

We can now compare “conceptualizations”

\[
\text{similarity}(\Phi, \Psi) = \frac{\sum_{ij} [K_\Psi]_{ij} [K_\Phi]_{ij}}{\sqrt{\sum_{ij} [K_\Psi]_{ij}^2} \cdot \sqrt{\sum_{ij} [K_\Phi]_{ij}^2}}
\]
Comparing “conceptualizations”

AlexNet vs VGG16 vs GoogleNet vs Self-supervised vs Word Embeddings

- AlexNet — ImageNet
- AlexNet — Places365
- VGG16 — ImageNet
- VGG16 — Places365
- GoogLeNet — ImageNet
- GoogLeNet — Places365

- Segmentation
- Fully-Supervised
- Self-Supervised
- Embeddings

- Classification
- Fully-Supervised
- Self-Supervised
- Embeddings

- Other Embeddings
- WordNet
- Word2Vec

- Tracking
- Audio
- Object-centric
- Moving
- Egomotion

- less similar
- more similar
Generating iconic examples

Attribution

Semantic identification
## Summary

### Generating conic examples
- Inversion vs activation maximization
- The importance of the prior / regularizer
- Aesthetic vs diagnostic

### Attribution
- (Modified) gradient backpropagation
- Excitation and relevance backpropagation
- Meaningful perturbation analysis
- Understanding via approximating models

### Semantic identification
- Concentrated vs distributed codes
- Comparing learned abstractions