AIMS Big Data Course
Universal, unsupervised and understandable representations

Dr Andrea Vedaldi
Dr Andrew Zisserman

For lecture notes and updates see
http://www.robots.ox.ac.uk/~vedaldi/teach.html

How can AI impact people?

Horizontal problems
- Autonomous driving
- Internet search
- Social networks
- Medical imaging (?)

Vertical problems
- Measuring plants
- Matching manuscripts
- Matching galaxies
- Counting penguins
- Recognizing flowers
- Measuring condensation
- Tracking crystals
- Measuring astrolabes
- Searching Greek vases
- Comparing 19th century paintings
- Matching Mesopotamian clay rolls
- ... 

Vertical AI problems

Archeology, bibliography, art historians, etc.
Image matching, search, comparison, recognition

Biology, zoology, medical imaging, material science, etc.

Counting penguins
Vertical AI problems

- Biology, zoology, areas of medical imaging, material science, etc.
- Counting crystals in thousands of videos

The need for (too) big data

- One of the most significant bottlenecks of deep learning
- The need for (too) big data

Visual Genome
- 100K images
- 4M annotations

ImageNet ILSVRC
- 1.2M images

ADE 20K
- 20K images
- Dense annotations


Open Images: A public dataset for large-scale multi-label and multi-class image classification. Krasin et al., 2017

Three key challenges of deep learning

- Universal Representations: Fewer models to train
- Unsupervised Representations: Less effort to train new models
- Understandable Representations: Trust, safety, and usability
Three key challenges of deep learning

- Universal Representations
  - Fewer models to train
- Unsupervised Representations
  - Less effort to train new models
- Understandable Representations
  - Trust, safety, and usability

Universal representation

- Object recognition
- Medical diagnosis
- Text spotting

Universal representation family

- Learning multiple visual domains with residual adapters, Rebuffi, Bilen, Vedaldi, NIPS 2017
Universal representation families

Object recognition
Medical diagnosis
Text spotting

Preview
- > 90% of parameters are shared
- Same or better performance than narrow models
- No forgetting

Applications
- Better than standard pre-training, especially for small dataset
- Efficient model storage, transmission, updating


Residual adapters

An efficient parametrization

Residual Adapters
- Shrinkage controls the amount of adaptation
- Better generalization when adapting to small target datasets

Adapters
- Tweak a fixed neural network block
- Interleaved with standard convolutions
- 1 x 1 filter bank
- Only ~10% of the parameters


Visual Domain Decathlon Challenge

A new benchmark

Goal: learn a single model that performs as well as possible on 10 very different visual domains

Universal Families vs UberNets

Complementary problems

Universal family: more domains
- Aircraft, people, flowers, text, glyphs, texture, Internet, ...

Detection, segmentation, boundaries, normals, parts, ...

UberNet: more tasks


Three key challenges of deep learning

Universal Representations
- Fewer models to train

Unsupervised Representations
- Less effort to train new models

Understandable Representations
- Trust, safety, and usability

Can we drop annotations?
Self-supervised learning

Images are cheap and abundant

However, manual annotations are extremely expensive

Self-supervised features

Deep image prior

Self-supervised structure

Deep image prior

Pretext task

Target task

Where to get supervision from

Self-supervised features

Perturbations

Motion

Modalities

Find a pretext task to pre-train a model $\Phi$

The pretext tasks come with cheap supervision

Fine-tune the model for a target task

Far less annotations are now required
Goal: learn a model $\Phi$ that reproduces a human annotator

Standard supervision

Nope: the “auto annotator” is just as complex as the model $\Phi$

Self-supervision from perturbations

Example perturbation = delete half of the image
Concrete learning scheme

Intuition: completing an image may require the network to learn about objects

Perturbations useful for self-supervision

Colorization as a proxy task for visual understanding. Larsson, Maire, Shakhnarovich. CVPR, 2017.


Self-supervision from related images

Look for image pairs that are correlated from the outset

Disadvantage: the network learns to operate on perturbed data

Source of related images

Deep network
Self-supervision using video sequence

Images related through time


Self-supervision using multiple modalities

Data sharing the same root cause

Sensor 1 (camera)
Sensor 2


Self-supervision using multiple modalities

Example: reconstruct sound from images


Self-supervision using multiple modalities

Example: photographer bias

Task: find out which picture is upright

The supervisory signal is in the photographer bias: people take pictures with a specific orientation.

How to setup the learning problem

Self-supervised features
Perturbations
Motion
Modalities
Constraining
Marginalising
Correlating

Probability
Perturbations
Motion
Modalities
Constraining
Marginalising
Correlating

How to setup the learning problem

Many pretext tasks are ill-posed

Approach 1: use a probability distribution

Explicitly model ambiguity by predicting a probability distribution

Approach 2: constrain the prediction task
Addressing ambiguous pretext tasks

E.g.: reorder frames causally instead of generating them.
Reordering is much easier than generation.

Learning and using the arrow of time. Wei, Lim, Zisserman, Freeman. CVPR, 2018.

Approach 3: reduce the amount of predicted information
Addressing ambiguous pretext tasks

Example: only predict what could move in an image.

Learning visual groups from co-occurrences in space and time. Isola, Zoran, Krishnan, Adelson. ICLR Workshop, 2015.
Common fate principle

Group together what moves together

Optical flow

Flow similarity (kernel)

Pairwise flow similarities

Damage: pick one frame

Encoder

Pairwise pixel similarities

Decoder

Loss


Approach 4: learning correspondences

Addressing ambiguous pretext tasks

Randomly extract $x$ and $y$ from either the same or different data

Learn to tell which one is the case

Reduces to predicting a binary switch

Audio-visual correspondences

Switch

visual network $\Phi_1$

audio network $\Phi_2$

Randomly extract $x$ and $y$ from either the same or different data

Learn to tell which one is the case

Reduces to predicting a binary switch


Approach 4: learning correspondences

Randomly extract $x$ and $y$ from either the same or different data

Learn to tell which one is the case

Reduces to predicting a binary switch
Co-occurrences may be better captured by **mutual information**

\[ \Phi_1 \quad \Phi_2 \]

A statistical “distillation” principle

**Learning objective**

\[
\max \; I(\Phi_1(f_2(z)), \Phi_2(f_2(z)))
\]

**Mutual Information**

- **Deep network 1**
- **Deep network 2**

---

**Image clustering and segmentation**

Learn maximally mutually informative classes

- **Sensor 1**
- **Sensor 2**

**Classifier network**

\[
p(c | x) \quad p(c | y)
\]

**Mutual Information**

- **Sensor 1**
- **Sensor 2**

---

**Clustering results**

- **ImageNet fruits clustering**
- **Satellite image segmentation**

- **Self-supervised features**
- **Self-supervised structure**
- **Deep image prior**
Can we learn the structure of visual objects explicitly?

From features to structure

Take I: A 3D surface

\[ S \subset \mathbb{R}^3 \]

Take II: an equivalence class of deformable surfaces

\[ x, x', S \subset \mathbb{R}^3, S' \subset \mathbb{R}^3 \]

Can put in correspondence different instances, categories

\[ x, x', S \subset \mathbb{R}^3, S' \subset \mathbb{R}^3, S'' \subset \mathbb{R}^3 \]
Homeomorphic to a sphere

\[ S \subset \mathbb{R}^3 \]
\[ S' \subset \mathbb{R}^3 \]
\[ Z = S^2 \]

Take III: Sphere

\[ Z = S^2 \]

Mathematical dogs

The Hairy-Ball Theorem

These are a few of the concepts and objects studied by topology: now we’ll look at a theorem.

If you look at the way the hairs lie on a dog, you will find that they have a ‘parting’ down the dog’s back, and another along the stomach. Now topologically a dog is a sphere (assuming it keeps its smooth shot and neglecting internal organs) because all we have to do is stretch its legs and frost it up a bit (Figure 80).

Figure 80

One might wonder whether it is possible to comb the hairs in such a way that all partings were eliminated. This would give a smooth
How can we learn this function without manual supervision?

Geometric consistency of the labelling function

\[ \forall u : \Phi(x; u) = \Phi(gx; gu) \]

Degeneracy: invariance can be trivially satisfied

\[ \forall u, u' : \begin{cases} \Phi(x; u) = \Phi(gx; u') \iff u' = gu \end{cases} \]

Addressing the degeneracy

Invariance & distinctiveness

\[ \forall u, u' : [\Phi(x; u) = \Phi(gx; u') \implies u' = gu] \]
Induces invariance and distinctiveness

A probabilistic formulation

Map pixels to 3D vectors $\Phi(x;u) \in \mathbb{R}^3$

- Vector length codes certainty

Probabilistic vector matching

- Conditional heat map:
  $$S(u' \mid u) = \langle \Phi(x;u), \Phi(gx;u') \rangle$$

- Conditional correspondence probability:
  $$p(u' \mid u) = \frac{e^{S(u' \mid u)}}{\int e^{S(u' \mid u')} \, du'}$$

Loss encourages both precision and accuracy:

$$L = \int \|u' - gu\|^2 p(u' \mid u) \, du \, du'$$

Data: $(x, g)$ where $x$ is a random image and $g$, the estimated flow or a random warp

Unsupervised learning of a dense equivariant labeller

Training data
- Pairs of video frames $x_t$ and $x_{t+\Delta}$
- $g$ is the optical flow

Learning an object from motion

Examples & results
3D deformable object

<table>
<thead>
<tr>
<th>Input Frames</th>
<th>Labelled Pixels</th>
<th>Projected to sphere</th>
<th>Vector Magnitude</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Panda" /></td>
<td><img src="image2" alt="Labelled Pixel" /></td>
<td><img src="image3" alt="Projected Sphere" /></td>
<td><img src="image4" alt="Vector Magnitude" /></td>
</tr>
</tbody>
</table>

Faces

Training data

Pairs of warped images $x$ and $x'$ from a dataset of ~200K celebrity images
$g$ are a synthetic warps

$(x_t, x_{t+1}, g)$

Learned embedding

Different face instances are automatically aligned

Cat faces
Equivariance revisited

Key ideas
1. **Equivariance** learns a data representation from a relationship \((x, x', g)\) between data pairs.
2. It works by trying to map the data into a simpler representation \(y\) where the relationship still holds true.

\[ \Phi(g \cdot x) = g \cdot \Phi(x) \]

Auto-equivariance, factorization, and normalization

A new assumption
The representation space \(Y\) coincides with the transformation group \(G\).

Consequences
1. We can rewrite the constraint as a factorization.
2. The factors provide an absolute reference frame for the data, normalizing it.

\[ \Phi(g \cdot x) \circ \Phi(x^{-1}) = g \]

Learning co-variant feature detectors

Aligning 3D object categories

Learning covariant feature detectors, Lenc Vedaldi, ECCW 2016

Input video 1

Input video 2
**Viewpoint factorization network**

![Diagram of Viewpoint Factorization Network](image1)

- Absolute viewpoint from CNN
- Relative viewpoint from SFM
- Learned absolute viewpoint

**Learned absolute viewpoint**

![Image of Learned Absolute Viewpoint](image2)

Learning 3D object categories by looking around them. Novotny, Larlus, Vedaldi, ICCV, 2017

**Learning to count**

![Diagram of Learning to Count](image3)

- Random partition
- Counting network

Train a network to count the number of "objects" within an image

**Key idea:** the count of the full image must be the sum of the counts of the subparts

Three-ways factorization


Three key challenges of deep learning

- Universal Representations
  - Fewer models to train
- Unsupervised Representations
  - Less effort to train new models
- Understandable Representations
  - Trust, safety, and usability

Kind of explanations

- Analysis
  - Given an off-the-shelf networks, explain what it knows, how it works, and how it learns
- Win an argument
  - The network explains its decision to a user, with the goal of convincing her
- Communicating a skill
  - Explain to a human or machine how to solve a certain class of problems, in general

Analysing deep neural networks

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?

Deep networks as encoders

- Given an input $x$, the network transforms it through a series of layers $c_1, c_2, \ldots, c_5$ and then outputs $f_6, f_7, f_8$.
- The final output is the encoded representation $\Phi(x)$.
- The network aims to minimize the loss between $\Phi(x)$ and the target $\Phi(y)$.
- The network learns the encoding function $\Phi$ to map inputs to useful representations.
Deep networks as encoders

Images
\( \mathcal{X} = \mathbb{R}^m \)

Codes
\( \mathcal{Y} = \mathbb{R}^n \)

Generating iconic examples

\( x \rightarrow \Phi \rightarrow y \)

Attribution

Multiple images map to the same code

How much information about \( x \) does \( y \) contain?

Images
\( \mathcal{X} = \mathbb{R}^m \)

Codes
\( \mathcal{Y} = \mathbb{R}^n \)

\( x_1, x_2, x_3 \rightarrow \Phi \rightarrow y \)
Pre-image

Reconstructions form an equivalence class of images, called a pre-image.

All pre-images that are indistinguishable for the network.

Finding pre-images via optimisation

Natural pre-images

We are interested in pre-images that can realistically be network inputs.

Pseudo-natural pre-images

Regularised energy

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

Constrained optimisation

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

Posterior probability

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

For example TV-norm

Understanding deep image representations by inverting them

Mahendran Vedaldi, CVPR, 2015

For example Deep Image Prior

Deep image prior

Ulyanov Vedaldi Lempitsky, CVPR, 2018

For example Plug & Play gen. nets

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

Nguyen, Yosinksi, Bengio, Dosovitskiy, Clune, CVPR, 2017
Generator nets as image parameterisations

Consider a generator network $\Psi$ with a fixed input $z_0$.

The network parameters $w$ can be thought as image parameters.

$w \mapsto x = \Psi(z_0; w)$

---

Fit a network to a single example

Start randomly-initialised network.

Given an image $x$, its parameter $w$ is recovered by solving the optimisation problem

$$\min_w \| x - \Psi(z_0; w) \|^2$$

This is similar to learning the network from a single image.

---

Deep image prior

For most generator networks fitting naturally-looking images is easier/faster than fitting others.

Deep image prior

Ulyanov Vedaldi Lempitsky, CVPR, 2018

---

Deep image

For inpainting we only reconstruct the visible pixels, implicitly infer the others.

$$\min_w \| m \odot (x - \Phi(w)) \|^2$$
Artificial Intelligence

The inverter is only given the code; it is not learned from data in any way.

Inverting codes via the deep

The inverter is only given the code; it is not learned from data in any way.

\[
\min_w \| \Phi(\Psi(w)) - \Phi(x_0) \|^2
\]

Inversion result

Inverting AlexNet

[Krizhevsky et al. 2012]
Inverting AlexNet
Inverting AlexNet
Inverting AlexNet
4 Reducing Overfitting

4.1 Data Augmentation

We 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 training set by a factor of 2048, though the resulting training examples are, of course, highly inter-dependent. Without this scheme, our network suffers from substantial overfitting, which would have forced us to use much smaller networks. At test time, the network makes a prediction by extracting 10 bits of constraint on the mapping from image to label, this making each training example impose 10 bits of constraint on the mapping from image to label.

The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 	imes 3 	imes 192$, and the fifth convolutional layer has 256 kernels of size $3 	imes 3 	imes 256$.

The second convolutional layer takes as input the (response-normalized pooling or normalization layers) output of the first convolutional layer and filters it with 256 kernels of size $3 	imes 3 	imes 256$. This increases the size of our network’s remaining layers.

The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network’s remaining layers is given by $253,440–186,624–64,896–64,896–43,264–48$.

In our implementation, the transformed images are generated in Python code on the CPU while the images with very little computation, so the transformed images do not need to be stored on disk.

Of data augmentation, both of which allow transformed images to be produced from the original training images. Specifically, we perform PCA on the set of RGB pixel values throughout the training set.

The second form of data augmentation consists of altering the intensities of the RGB channels in the images. This is the reason why the input images in Figure 2 are $224 	imes 224 	imes 3$-dimensional.

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities in the network:

- **Conv 1**: First convolutional layer.
- **Conv 2**: Second convolutional layer.
- **Conv 3**: Third convolutional layer.
- **Conv 4**: Fourth convolutional layer.
- **Conv 5**: Fifth convolutional layer.
- **FC 6**: First fully connected layer.
- **FC 7**: Second fully connected layer.
- **FC 8**: Third fully connected layer.

Is the code semantic or visual?

Activation maximization

$$\min_w \langle e_z, \Phi(\Psi(w)) \rangle$$
Artificial Intelligence

References

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

Visualizing and understanding convolutional networks
Zeiler Fergus. Proc. ECCV, 2014

Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Simonyan Zisserman Vedaldi, ICLR, 2014

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

Google “imagenet”

Understanding neural networks through deep visualisation
Yosinski et al. ICMW, 2015

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

Deep image prior
Ulyanov Vedaldi Lampelzky, 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 maximization for intermediate neurons
- Improved regularizers, artistic applications (deep dreams)
- Activation maximization using empirical prior, deconvnet
- More regularizers, toolbox
- Strong learned regularizer, sample diversity
- Advanced “data agnostic” regularization

Effect of the prior

Deep Image Prior
TV-Norm Prior
Artificial Intelligence

The inverter is only given the code; it is not learned from data in any way.

Inverting codes via the deep

Inversion result

Inverter

Minimised

model \( \Phi \)

Learning the inverter

Popular methods combine:

- perceptual loss \( x_0 \approx x \)
- feature rec. loss \( \Phi(x_0) \approx \Phi(x) \)
- adversarial loss (GAN) \( p(x_0) \approx p(x) \)

Diagnostic vs

Our goal: diagnose a given network \( \Phi \)

But inversions also reflect the chosen "natural image" prior \( p(x) \)

\[
p(x) = \begin{cases} 
\text{only prior is the structure of the gener.} & \text{Deep Image Prior} \\
\text{prior comes from training a GAN on ImageNet} & \text{Plug & Play Gen. Net.} \\
\text{ImageNet empirical distribution} & \text{Empirical prior} 
\end{cases}
\]

Illustrates the model \( \Phi \)

Illustrates the prior \( p(x) \)

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

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
Reviews and interfaces

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

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

Definitely check out Distill!

Attribution

Where is the model looking?

Backprop methods: grad

The “salient” pixels usually light up
Early backprop 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

Backprop: deconv, grad, guided grad


Comparisons

- **DeconvNet**
  - Sharp
  - Poor spatial selectivity

- **Gradient**
  - Blurry
  - OK spatial selectivity

- **Guided Backprop**
  - Sharp
  - OK spatial sensitivity

Warning: they all still have poor channel selectivity
Smoother grads

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

Gradient × input
\[ x \odot \frac{d\Phi(x)}{dx} \]

Integrated Gradients
\[ (x - \bar{x}) \otimes \int_0^1 \frac{d\Phi(x - \alpha(x - \bar{x}))}{dx} d\alpha \]

SmoothGrads
\[ E \left[ \frac{d\Phi(x + \epsilon)}{dx} \right], \epsilon \sim \mathcal{N} \]


Comparisons

Backprop: CAM and

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

Lack of channel

Visualising any output results in about the same result
The meaning of attribution maps

For most methods, attribution is defined algorithmically.

Hence, the meaning of the output is not so clear.

Grad method = sensitivity analysis

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

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

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

Perturbations 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 $\pi(x)$
- For a distribution $p(x)$ and a fixed $\pi(x)$
- For a distribution $p(\pi)$ and a fixed $x$

... $\Phi(\pi)$

Extremal Perturbations

Change the input and observe the effect on the output.

Input Occlusion RISE

[Zeiler and Fergus, ECCV 2014; Petsiuk et al., BMVC 2018]

Blur everywhere $\Rightarrow$ response suppressed

TODO: Regenerate animations using latest code base
Preserve 10% \Rightarrow \text{response preserved}

Meaningful perturbations

We seek the "smallest elision" that maximally changes the neuron activation.

"cat" probability

1.00

0.01

0.5

Original

Redact-out

Blur-out

Adversarial perturbations

Neural networks are fragile to adversarial perturbations.

Adversarial perturbations attract gradient descent.


Extremal perturbations

A mask is optimized to

\[
\text{argmax}_m \Phi(m \otimes x)
\]

subject to \(\text{area}(m) = a\).
Artificial Intelligence

Optimizing w.r.t. to an area constraint is challenging.

Here we re-formulate it as matching a rank statistics.

\[ \text{Area constraint subject to } \text{area}(m) = a \]

\[ \text{vectorize sort} \]

\[ L_{\text{area}} = \| \text{vecsort}(m) - r_a \|_2 \]

\[ m_r \alpha \text{vecsort}(m) \]

Smooth masks

\[ \text{conv}(u; m; k) = \frac{1}{Z} \sum_{v \in \Omega} k(u - v)m(v) \]

\[ \text{maxconv}(u; m; k) = \max_{v \in \Omega} k(u - v)m(v) \]

\[ \text{smmax}_{u \in \Omega; T} f(u) = \frac{\sum_{u} f(u) \exp(f(u)/T)}{\sum_{u} \exp(f(u)/T)} \]

\[ \text{smoothconv}(u; m; k; T) = \text{smmax}_{v \in \Omega; T} k(u - v)m(v) \]

m(v) : mask

Comparison with prior work on "meaningful perturbations"

Compared to Fong and Vedaldi, 2017, we remove all regularization terms in the energy term.

Our innovations result in a method that’s more principled, stable, and sensitive.
Algorithm

1. Pick an area $\alpha$
2. Use SGD to solve the optimization problem for a large $\lambda$:
   \[ \arg\max_m \Phi(\text{smooth}(m) \otimes x) - \lambda \| \text{vecsort}(\text{smooth}(m)) - r_a \|_2^2 \]
3. If needed, sweep $\alpha$ and repeat

Foreground evidence is usually sufficient

Large objects are recognised by their details
Small objects contribute cumulatively

Suppressing the background may overdrive the network

Diagnosing networks
Example: the hot chocolate is recognized via the spoon and the truck vs the license plate

CNN fragility
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

$\delta^* = \arg\min_{\|\delta\|} \|y_{\text{arbitrary}} - \Phi(x + \delta)\|_{\|\delta\|}$
Dangerous adversaries


Adversarial defence

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

(Illustrative of attribution, not really a recommended defence strategy)

Assessing attribution: pointing game & weak localisation

Goal: measure the spatial correlation between attribution maps and object occurrences

If the correlation is strong:
- the diagnosed model "understand" the object an
- the attribution method can tell

However, if the correlation is poor, either:
- the diagnoses model does not understand the object
- the attribution method fails to tell
Assessing attribution: neuron sensitivity

Attribution should generally result in a different output depending on which neuron one wishes to visualise.

Golden Tiger

Gradient DeConvNet Guided BP Grad × Input Excit. BP Contrastive

Assessing attribution: parameter sensitivity

Attribution should also produce a different output if the model weights are different — e.g. random


Assessing attribution: shift invariance


Network 1

[0 1]

Network 2

[-1 0]

Input

Gradients

Signal Methods

GB PN

Assessing attribution: perturbation analysis

Display

elephant
Attributing channels at intermediate layers

Spatial attribution

Channel attribution

Channel attribution

\[ \Phi_{b}(m \odot \Phi_{a}(x)) \leq \Phi_{a}(x) \]

Subject to area(m) = a
Activation “diffing”

\[ \sum m \otimes \Phi_a(x) \]

Original
\( \Phi_a(x) \)

Perturbed
\( m \otimes \Phi_a(x) \)

Equivariance

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

Equivalence

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.

Collected references

Software

Captum
https://pytorch.org/captum/
More than just vision

TorchRay
https://github.com/facebookresearch/TorchRay
Attribution, reproducibility, benchmarks

Summary

Universal Representation
- Compact representation families

Unsupervised Representation
- Self-supervision for learning features
- Self-supervision for learning structure
- What’s in the prior

Understandable Representations
- Iconic visualizations
- Attribution
- Semantic identification