Računalni vid

uvod u generativno modeliranje

Siniša Šegvić
UniZg-FER D307

Agenda

Part 1: recent advances in generative recognition

□ algorithms for generating complex data

□ applications of generative models

Part 2: normalizing flows and energy-based models

□ energy-based models

□ generating complex data with bijective models

□ affine coupling and other formulations of normalizing flows

Part 3: generative recognition for dense anomaly detection

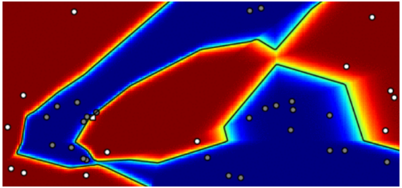
□ generating synthetic training data

□ finding anomalies according to likelihood

Intro: recognition

Discriminative recognition recovers $P(Y=y|X=x)$

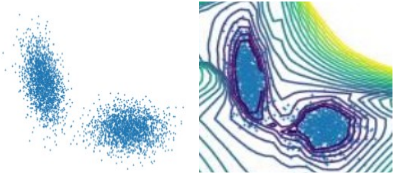
- default flavour of machine learning
- established technology, exciting applications



[unizg-fer-dl1]

Generative recognition recovers $p(X=x, Y=y)$

- or, equally interesting, $p(X=x|Y=y)$ or $p(X=x)$
- called *generative* since sampling from them generates **synthetic data** in the input space
- somewhat eclipsed by the success of discriminative approaches
- rapidly developing, exciting applications

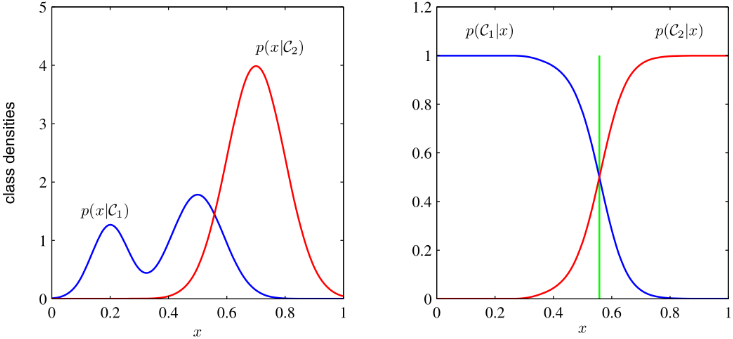


[delic21sem]

Intro: generative vs discriminative

Both approaches produce **probabilistic** output in each input datum, however:

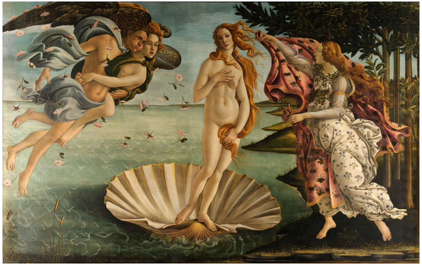
- □ discriminative models produce distributions over **targets**
- □ generative models produce distributions over **inputs**
  - □ how to normalize the distribution, i.e. ensure that it sums (integrates) to 1?
- □ generative recognition is tough!



[bishop06book]

Intro: generative vs discriminative (2)

Intuitively, it is easier to tell the painter than to actually do the painting:



[public domain]

It is easier to distinguish composers than to develop musical ideas.

In words of the infamous food critic Anton Ego:

> *... in the grand scheme of things, the average piece of (bad food) is probably more meaningful than our criticism designating it so.*

Intro: generative vs discriminative (3)

Things get especially tough when the data is complex:

- language: 15-20 words per sentence, 170000 total words (English)
- vision: at least 3×64×64 components, 256 values per component
- generative recognition has to consider integrals over thousands of dimensions in order to normalize $p(\mathbf{x})$

Close encounters with the curse of dimensionality!

Intro: tasks

Generative approaches aim at density of the training data

Three main tasks of a generative model:

□ generate synthetic data points by sampling $p(\mathbf{x})$

  □ trade off: quality vs coverage

  □ useful for **content creation** and **improving** discriminative models

□ transform data-points to a factorized latent representation

  □ useful for **content editing**

□ evaluate density $p(\mathbf{x})$

  □ only generative models with explicit density can do that

  □ useful for **anomaly detection**, **compression**

Intro: approaches

Generative algorithms come in many flavours and many ways to appreciate them:

Algorithm	latent	bias	sampling	density
Energy	unable	coverage	slow	unnormalized
VAE	easy	coverage	fast	ELBO
Autoreg.	unable	coverage	slow	exact
GAN	unable	quality	fast	unable
NFlow	easy	coverage	fast	exact
Diffusion	easy	(coverage)	slow	ELBO
Score	unable	coverage	slow	(unable)

Intro: density estimation

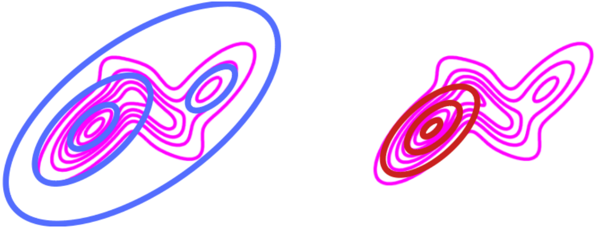
Comparison of generative algorithms with respect to efficient density estimation:

algorithm / density formulation	tractability	efficiency
$p_{EBM}(\mathbf{x}) = e^{-E(\mathbf{x})} / \int_{\mathbf{x}} e^{-E(\mathbf{x})}$	**tractable inference**	**fast**
$p_{VAE}(\mathbf{x}) = \int_{\mathbf{z}} p(\mathbf{x}\|\mathbf{z})p(\mathbf{z})d\mathbf{z}$	intractable (ELBO)	fast
$p_{\mathrm{ar}}(\mathbf{x}) = p(x_1) \prod_{i=2}^{HW} p(x_i\|\mathbf{x}_{<i})$	tractable	slow
$p_{GAN}(\mathbf{x}) =?$ (implicit density)	unavailable	
$p_{flow}(\mathbf{x}) = p_z(\mathbf{f}(\mathbf{x})) \cdot \|\det(\frac{\partial \mathbf{f}}{\partial \mathbf{x}})\|$	**tractable**	**fast**
$p_{diff}(\mathbf{x}) = \int p(\mathbf{x}\|\mathbf{x}^{(1)})$ $\prod_{t=1}^{T-1} p(\mathbf{x}^{(t)}\|\mathbf{x}^{(t+1)})\pi(\mathbf{x}^T)d\mathbf{x}^{(1...T)}$	intractable (ELBO)	slow

Applications: content creation

Most generative algorithms are able to generate data.

However, some optimize for coverage while others optimize for quality [lucas19neurips].



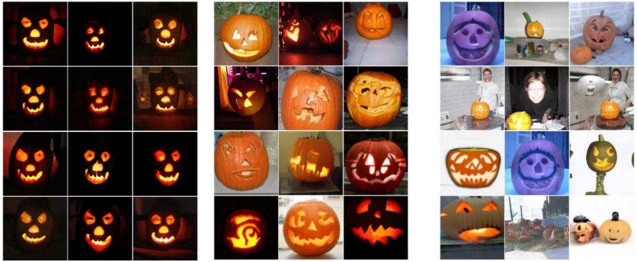
Normalizing flows - high coverage (left); GANs - high quality (right).

[grcic21neurips]

[sehwag22cvpr]

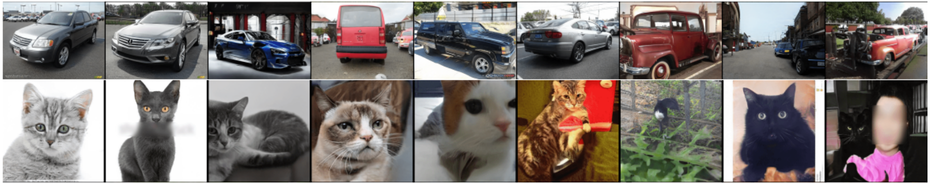
Applications: boosting diversity

Recent work leverages computational power to boost diversity without sacrificing quality:



[sehwag22cvpr]

Other recent work allows to favour either quality (left) or diversity (right):



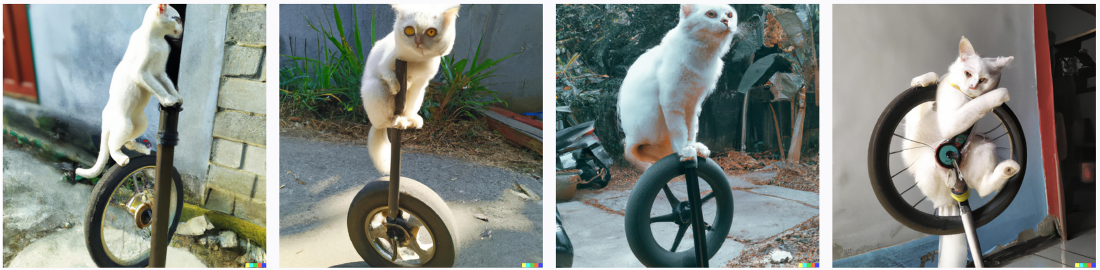
[humayun22cvpr]

Applications: conditional generation

For practical puprposes, we are mostly interested in conditional generation

A popular recent approach connects language embeddings with generative vision.

This is what I got by feeding "a photo of a white cat on a unicycle" to DALL-e:



[DALL-E]

Applications: conditional generation (2)

It also works the other way round (from images to text):

GT: A young boy in the park throwing a frisbee.

L-Verse: A young boy throwing a green frisbee in a lush green park.

GT: A laptop and a cell phone on a table.

L-Verse: A collection of electronic devices and cords sitting on top of a table.

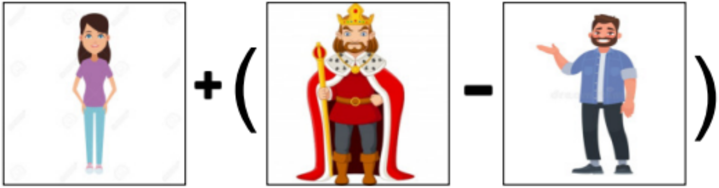
GT: A small bathroom is shown from a door.

L-Verse: A bathroom with a shower curtain over the bathtub next to a toilet.

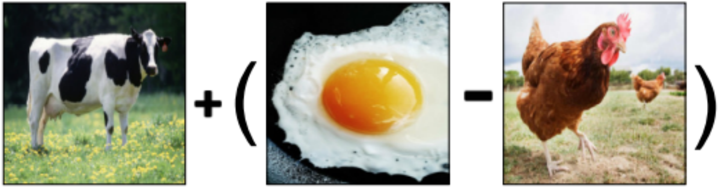
[kim22cvpr]

Applications: conditional generation (3)

One can also perform arithmetic operations on visual semantics and display results in text:



A young queen.

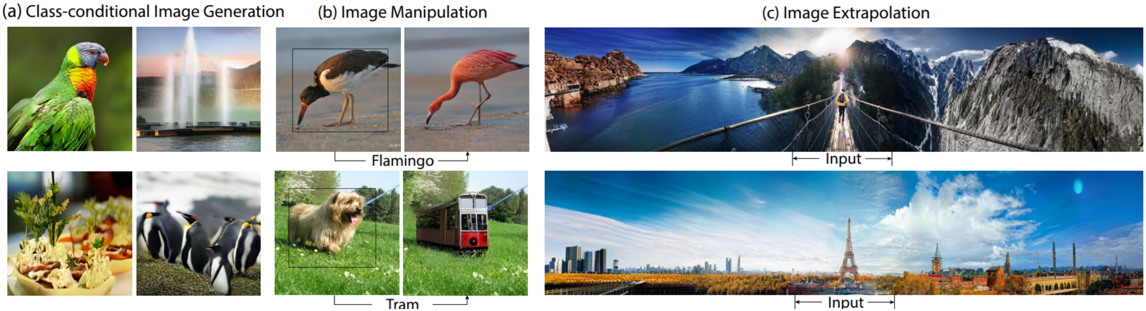


A cow's milk.

[tewel22cvpr]

Applications: content editing

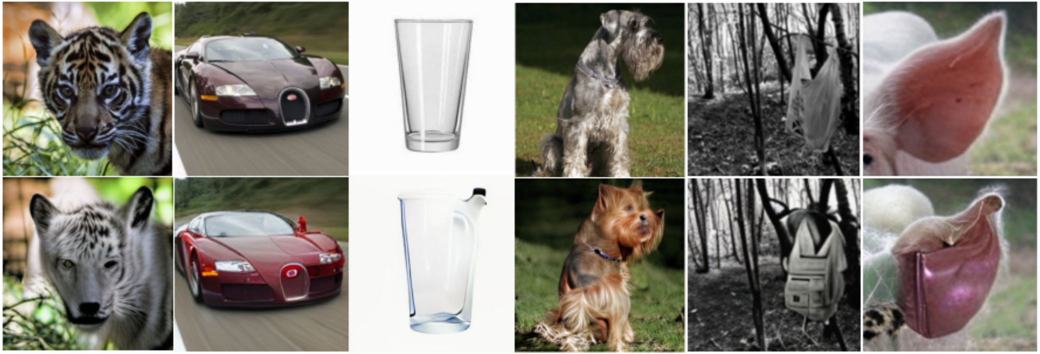
Conditional generation, visual replace, extrapolation by editing the VQ VAE latent:



[chang22cvpr]

Applications: content editing (2)

Another instance of visual replace:



Tiger → White wolf

Black → Red

Glass → Water jug

Standard schnauzer → Yorkshire terrier

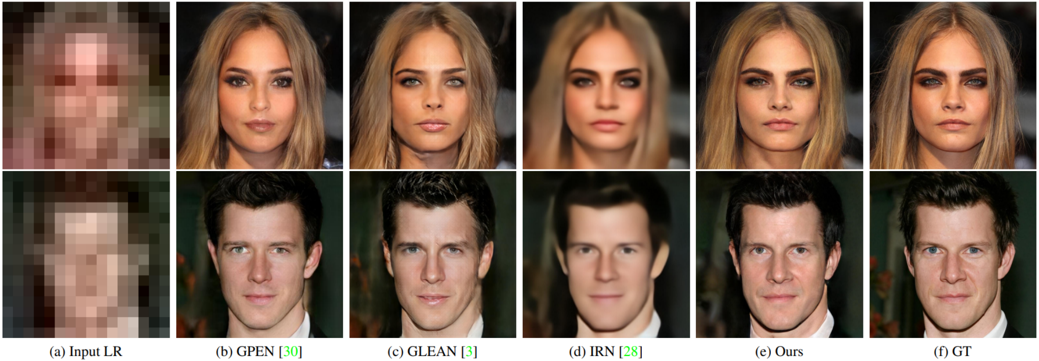
Plastic bag → Backpack

Sow's ear → Silk purse

[couairon22cvpr]

Applications: content editing (3)

Super resolution by leveraging Style GAN latent:

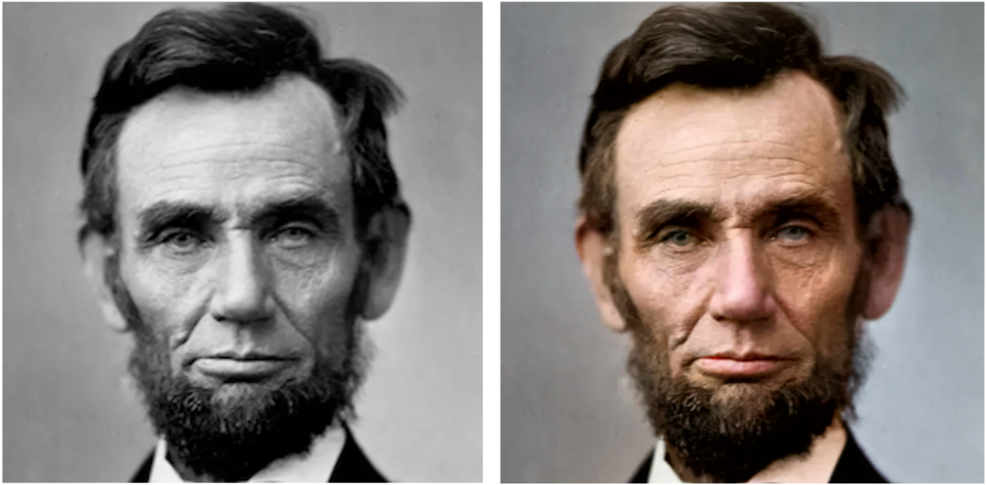


(a) Input LR (b) GPEN [30] (c) GLEAN [3] (d) IRN [28] (e) Ours (f) GT

[zhong22cvpr]

Applications: content editing (4)

Colorization and other inverse problems (inpainting, medical image reconstruction):



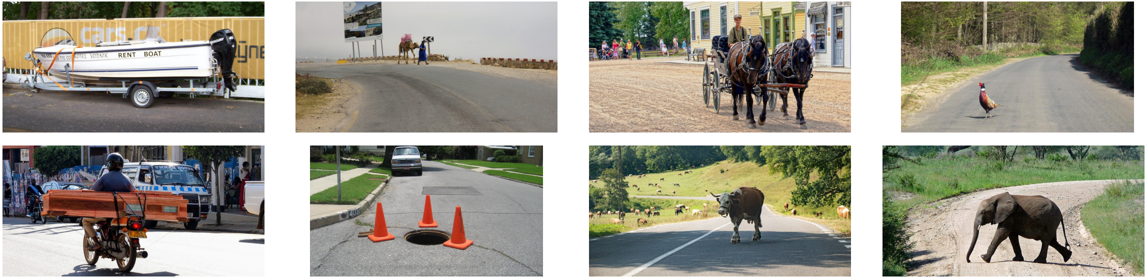
[song22blog]

Applications: anomaly detection

Detect images (or pixels) that are unrelated to the training data

Popular benchmark: Segment Me If You Can [chan21neurips]

The task is to detect pixels that do not belong to any of the 19 road-driving classes:



[https://segmentmeifyoucan.com/]

EBMs: likelihood

Consider $\mathrm{E}_\theta(\mathbf{x})$ as a differentiable scalar function of the input:

- □ we define energy-based density as $p_\theta(\mathbf{x}) = \frac{\exp(-\mathrm{E}_\theta(\mathbf{x}))}{Z(\theta)}$
- □ the normalizing constant is: $Z(\theta) = \int_{\mathbf{x}} \exp(-\mathrm{E}_\theta(\mathbf{x}))\, d\mathbf{x}$
- □ the density p_θ often appears in physics and is also known as Boltzmann distribution.

We can train such model by optimizing negative log likelihood on training data:

- □ $\mathcal{L}(\theta) = -\frac{1}{N}\sum_i \log p_\theta(\mathbf{x_i})$
- □ this loss is equivalent to KL divergence between the data distribution and $p_\theta(\mathbf{x})$
- □ many generative models use this loss (those who do can **recover the likelihood**!)

Conceptually, this is the simplest formulation of a generative model!

EBMs: LEARNING

We usually optimize $-\log p_\theta(\mathbf{x})$ due to being additive wrt iid data:

$$\begin{aligned}-\log p_\theta(\mathbf{x}) = -\log \frac{\exp(-\mathrm{E}_\theta(\mathbf{x}))}{Z_\theta} &= \mathrm{E}_\theta(\mathbf{x}) + \log Z_\theta \\ &= \mathrm{E}_\theta(\mathbf{x}) + \log \int_{\mathbf{x}'} \exp(-\mathrm{E}_\theta(\mathbf{x}'))d\mathbf{x}'\end{aligned}$$

The above criterion is minimized when we decrease the energy of the current datum and increase the energy in **all other data.**

The **all other data** part is problematic:

□ we are supposed to tweak the model in all possible images in each iteration

□ there are $256^{32 \cdot 32 \cdot 3}$ images in CIFAR 10 (a number with thousands of zeros...)

□ we refer to such situations as intractable.

EBMs: gradients

A closer look at the gradients (note the Leibniz rule) confirms the fears:

$$\begin{aligned}
\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} &= -\frac{\partial \mathrm{E}_\theta(\mathbf{x})}{\partial \theta} - \frac{1}{Z_\theta} \int_{\mathbf{x}'} \exp(-\mathrm{E}_\theta(\mathbf{x})) \frac{\partial(-\mathrm{E}_\theta(\mathbf{x}'))}{\partial \theta} d\mathbf{x}' \\
&= -\frac{\partial \mathrm{E}_\theta(\mathbf{x})}{\partial \theta} + \int_{\mathbf{x}'} p_\theta(\mathbf{x}) \frac{\partial \mathrm{E}_\theta(\mathbf{x}')}{\partial \theta} \\
&= \mathbb{E}_{p_\theta(\mathbf{x}')}\left[\frac{\partial \mathrm{E}_\theta(\mathbf{x}')}{\partial \theta}\right] - \frac{\partial \mathrm{E}_\theta(\mathbf{x})}{\partial \theta}
\end{aligned}$$

Note that these are gradients of the positive log-likelihood.

Applying them will decrease the energy of the current datum and increase the energy in **all other data**.

EBMs: PRACTICE

Practical implementations approximate intractable expectation $\mathbb{E}$ with MCMC (n=1!):

$$-\frac{\partial \log p_\theta(\mathbf{x})}{\partial \theta} = \frac{\partial \mathrm{E}_\theta(\mathbf{x})}{\partial \theta} - \frac{\partial \mathrm{E}_\theta(\mathbf{x}^s)}{\partial \theta}$$

We have seen that conceptual simplicity does not imply convenient implementation:

- □ hard learning: the gradients of the loss involve differentiation of an intractable integral
  - □ you can't iterate over all possible images!
  - □ even if you could, you couldn't keep all activations that are required for efficient backprop!
- □ MCMC approximation requires slow sampling through random walks:
  - □ per-dimension (Gibbs)
  - □ hill-climbing (Langevin)
  - □ this appears feasible only for small images

EBMs: hybridization [Gratwohl20iclr]

Idea: reinterpret discriminative logits $\mathbf{s} = f_\theta(\mathbf{x})$ as logarithm of unnormalized joint density:

- □ this is an energy-based model since we can consider s_y as $-\mathrm{E}(\mathbf{x}, y)$

$$p_\theta(\mathbf{x}, y) = \frac{\exp(s_y)}{Z(\theta)}$$

We can easily recover unnormalized density of the data by marginalizing out y:

$$p_\theta(\mathbf{x}) = \sum_y p_\theta(\mathbf{x}, y) = \frac{\sum_y \exp(s_y)}{Z(\theta)}$$

This is again energy-based density since we can say $-\mathrm{E}(\mathbf{x}) = \log \sum \exp s_y$

EBMs: hybridization (2)

This view delivers the same class posteriors as in the standard case:

$$\mathsf{P}^{\mathrm{std}}_{\theta}(y \mid \mathbf{x}) := \mathsf{softmax}(\mathbf{s}) = \frac{\exp(s_y)}{\sum_k \exp(s_k)}$$

$$\mathsf{P}^{\mathrm{hybrid}}_{\theta}(y \mid \mathbf{x}) = \frac{p_\theta(\mathbf{x}, y)}{p_\theta(\mathbf{x})} = \frac{\exp(s_y)/Z(\theta)}{\sum_k \exp(s_k)/Z(\theta)} = \frac{\exp(s_y)}{\sum_k \exp(s_k)}$$

Both frameworks introduce assumptions to be confirmed through training

- □ softmax vs log-unnormalized-joint

The hybrid formulation must be trained with a compound loss

- □ generative: $-\log p_\theta(\mathbf{x_i}) = \log Z(\theta) - \log \sum_y \exp(s_y)$
- □ discriminative: $-\log \mathrm{P}_\theta(y_i \mid \mathbf{x_i})$
- □ this can succeed since softmax has a spare degree of freedom

EBMs: summary

A probability density function must integrate to 1

- □ this innocent fact causes much pain!

When the model sees new data, the training should raise its likelihood

- □ however, distribution learning is a zero-sum game
- □ if you give some probability to $\mathbf{x}_{127}$, you must take the same amount from the others!

Energy-based models address this by directly optimizing $E_\theta(\mathbf{x}) + \log Z_\theta$

- □ in practice this involves approximation $Z_\theta = \exp(-E_\theta(\mathbf{x}^s))$ (MCMC with n=1)

This can also be addressed by embedding the unit-integral constraint into the model itself

- □ this is the point where we meet normalizing flows!

NFs: from EBMs to flows

An EBM model $\mathrm{f}_{\theta_{\mathrm{EBM}}}$ maps high-dimensional inputs $\mathbf{x}$ into scalar energy z

- □ training EBMs is hard since the predicted energy gives unnormalized density.

A normalizing flow $\mathrm{f}_{\theta_{\mathrm{NF}}}^{-1}$ maps high-dimensional inputs $\mathbf{x}$ into decorelated vectors $\mathbf{z}$ from which a normalized density is easily recovered.

The learned distribution can be easily sampled by evaluating random noise $\mathbf{z}$ in the forward direction: $\hat{\mathbf{x}} = \mathrm{f}_{\theta_{\mathrm{NF}}}(\mathbf{z})$,

Requirements (no free lunch):

- □ dim($\mathbf{z}$) must equal dim($\mathbf{x}$),
- □ $\mathbf{f}_\theta$ must be bijective,
- □ $\det \nabla \mathbf{f}_\theta^{-1}$ must be easy to compute.

NFs: definition

A normalizing flow $\mathbf{f}_\theta$ **bijectively** maps high-dimensional $\mathbf{z}$ into high-dimensional $\mathbf{x}$

- it recovers $p(\mathbf{x})$ by leveraging **change of variables**:

$$p_x(\mathbf{x}) = \left|\det \frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right| \cdot p_z(\mathbf{z}) = |\det \nabla \mathbf{f}_\theta^{-1}(\mathbf{x})| \cdot p_z(\mathbf{f}_\theta^{-1}(\mathbf{x}))$$

- distribution of the latents is hardwired: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  - fully factorized latents represent independent factors of variation
- $p(\mathbf{x})$ integrates to 1 **by design** (even with random weights!)

We usually use the absolute derivative so we can neglect the direction of integration.

Non-bijective flows are also feasible (however, the learning gets complicated).

NFs: change of variables vs substitution (1D)

Let us apply the substitution rule for integration of $p_z(z)$ while assuming $x = f(z)$:

$$\mathrm{P}_z(z \in S) = \int_{z \in S} p_z(z) dz = \int_{x \in f(S)} p_z(f^{-1}(x)) \cdot |\frac{dz}{dx}| \cdot dx = \int_{x \in f(S)} p_z(f^{-1}(x)) \cdot |\nabla f^{-1}(x)| \cdot dx$$

However, we can also express $P(z \in S)$ directly in terms of $p_x(x)$:

$$\mathrm{P}_z(z \in S) = \mathrm{P}_x(x \in f(S)) = \int_{x \in f(S)} p_x(x) dx$$

The two equations hold for any S. Hence, the comparison of the two right-hand sides shows that the change of variables formula follows from **integration by substitution**:

$$p_x(x) = p_z(f^{-1}(x)) \cdot |\nabla f^{-1}(x)|$$

NFs: intuition

Assume we have a random variable $Z \sim U(0, 1)$.

Assume we have a dependent random variable $X = f(Z) = 2 \cdot Z$.

Obviously, $p_Z(z) = 1$ for all $z \in [0, 1]$. But how much is $p_X(0.4)$?

Let us find out by changing the variable:

$$p_X(x) = p_Z(f^{-1}(x)) \cdot |\nabla f^{-1}(x)|$$

We recover $p_x(0.4)$ as $p_z(f^{-1}(0.4)) \cdot |\nabla f^{-1}(0.4)| = 0.5$.

□ $z(X = 0.4) = f^{-1}(0.4) = 0.2$, $p_z(Z = 0.2) = 1$

□ $\nabla f^{-1}(0.4) = 0.5$

□ $p_x(0.4) = 0.5$ fits since $P(Z < 1) = P(X < 2) = 1$.

NFs: change of variables (N-D)

The relations remain very similar even in the multivariate case:

$$\begin{aligned} p_{\mathbf{x}}(\mathbf{x}) &= p_z(\mathbf{z}) \mid \det \nabla \mathbf{f}_\theta(\mathbf{z}) \mid \\ &= p_{\mathbf{z}}(\mathbf{f}_\theta^{-1}(\mathbf{x})) \mid \det \nabla \mathbf{f}_\theta^{-1}(\mathbf{x}) \mid \end{aligned}$$

Instead of the derivative, now we have a determinant of the Jacobian:

- determinant is a product of eigenvalues: it reflects the local volumetric stretching of the linear transformation implied by the matrix

If the transformation is composite ($\mathbf{f}_\theta = \mathbf{f}_1 \circ \mathbf{f}_2 \circ ... \circ \mathbf{f}_L$), then the composite determinant is a product of determinants of the individual Jacobians:

$$|\det \nabla \mathbf{f}_\theta^{-1}(\mathbf{x})| = \prod_\ell |\det \nabla \mathbf{f}_\ell^{-1}(\mathbf{z}_\ell)|, \quad \text{where } \mathbf{z}_1 = \mathbf{x}.$$

NFs: model

A normalizing flow $\mathbf{f}_\theta$ bijectively maps the latent $\mathbf{z}$ to the input $\mathbf{x}$

- □ "normalizing": a complex distribution $p_\mathbf{x}(\mathbf{x})$ is transformed into a normalized one $p_\mathbf{z}(\mathbf{z})$:

$$p_\mathbf{x}(\mathbf{x}) = p_\mathbf{z}(\mathbf{f}_\theta^{-1}(\mathbf{x})) \mid \det \nabla \mathbf{f}_\theta^{-1}(\mathbf{x}) \mid$$

- □ "flow": the transformation is performed through a number of differentiable steps:

$$\mathbf{f}_\theta = \mathbf{f}_1 \circ \mathbf{f}_2 \circ \ldots \circ \mathbf{f}_L$$

It is a generative model since it can:

- □ evaluate the data likelihood $p_\mathbf{x}(\mathbf{x})$
- □ generate new data by sampling from $\mathbf{z}$: $\mathbf{x} = \mathbf{f}(\mathbf{z})$
- □ perform mapping to the latent space: $\mathbf{z} = \mathbf{f}^{-1}(\mathbf{x})$

NFs: loss

Normalizing flows are trained to maximize the likelihood of the training data:

$$\begin{aligned}\mathcal{L}(\theta|\mathbf{x}_i) &= -\log p(\mathbf{x}_i) = -\log p_{\mathbf{z}}(\mathbf{f}_\theta^{-1}(\mathbf{x}_i)) - \log|\det \nabla \mathbf{f}_\theta^{-1}(\mathbf{x}_i)| \\ &= -\log p_{\mathbf{z}}(\mathbf{f}_\theta^{-1}(\mathbf{x}_i)) - \sum_\ell \log|\det \nabla \mathbf{f}_\ell^{-1}(\mathbf{z}_{i\ell})|\end{aligned}$$

- □ the term with prior likelihood pushes the latents to $\mathbf{0}$
  - □ we usually have $\mathbf{z} \sim \mathcal{N}(0,1)$
- □ the term with the determinants opposes and prevents the collapse

This objective can be optimized with the usual variants of SGD

NFs: affine coupling

The most popular computational layer for normalizing flows:

- □ the input representation $\mathbf{z}$ is split (e.g. mapwise) into two subsets $\mathbf{z}_1$ and $\mathbf{z}_2$
- □ the two subsets are separately processed as follows [dinh17iclr]:

$$\begin{aligned}\mathbf{z}_1' &= \mathbf{z}_1\\ \mathbf{z}_2' &= \mathbf{z}_2 \odot \exp s_{\theta_s}(\mathbf{z}_1) + t_{\theta_t}(\mathbf{z}_1)\end{aligned}$$

This formulation supports the inverse (generative) pass:

$$\begin{aligned}\mathbf{z}_1 &= \mathbf{z}_1'\\ \mathbf{z}_2 &= [\mathbf{z}_2' - t_{\theta_t}(\mathbf{z}_1')] \oslash \exp s_{\theta_s}(\mathbf{z}_1')\end{aligned}$$

The coupling network $(s_{\theta_s}, t_{\theta_t})$ can have arbitrary structure and complexity.

NFs: affine coupling (2)

Recall the forward pass through the affine coupling module:

$$\begin{aligned}\mathbf{z}_1' &= \mathbf{z}_1\\ \mathbf{z}_2' &= \mathbf{z}_2 \odot \exp s_{\theta_s}(\mathbf{z}_1) + t_{\theta_t}(\mathbf{z}_1)\end{aligned}$$

The Jacobian is triangular; its determinant is the product along the diagonal:

$$\det \frac{\partial z_1' z_2'}{\partial z_1 z_2} = \prod_j \exp s_{\theta_s}(\mathbf{z}_{1j}) \ .$$

The subsequent coupling module will revert the splits:

- □ this time Z_2 will pass through unchanged

NFs: more details

Normalizing flow with 1x1 convolutions [kingma18neurips]:

- 1x1 convolution corresponds to a matrix multiplication
- inverse is easy after LU decomposition
- Jacobian determinant is a product of diagonal elements of U ($L_{ii} = 1$)

Normalizing flows require careful implementation in order to avoid numerical instability.

Normalizing flows can also be built on top of standard residual architectures (non-bijective flows) [behrmann19icml]:

- iterative inverse with fixed-point algorithm (price: 5-10 forward passes)
  - key condition: Lipschitz condition of residual blocks $Lip(g) < 1$
- Jacobian determinant also iteratively approximated
  - Skilling-Hutchinson trace estimator

NFs: SUMMARY

Properties:

- □ exact inference: $p(\mathbf{x}) = p(\mathbf{z}) \prod_\ell |\det \nabla \mathbf{f}_\ell^{-1}(\mathbf{z}_\ell)|$
- □ fast generation $\mathbf{x} = \mathbf{f}(\mathbf{z})$
- □ fast mapping to the latent space $\mathbf{z} = \mathbf{f}^{-1}(\mathbf{x})$

Requirements on computational units ("layers"):

- □ bijective, support for forward and inverse transformations
- □ efficient evaluation of the Jacobian determinant

Rationale:

- □ when the model sees news data, it increases their likelihood
- □ this rearranges the entire surface of the predicted likelihood (everybody gets affected!)
- □ the likelihood always integrates to 1 by construction

NFs: pros and cons

Advantages of normalizing flows with respect to other generative models:

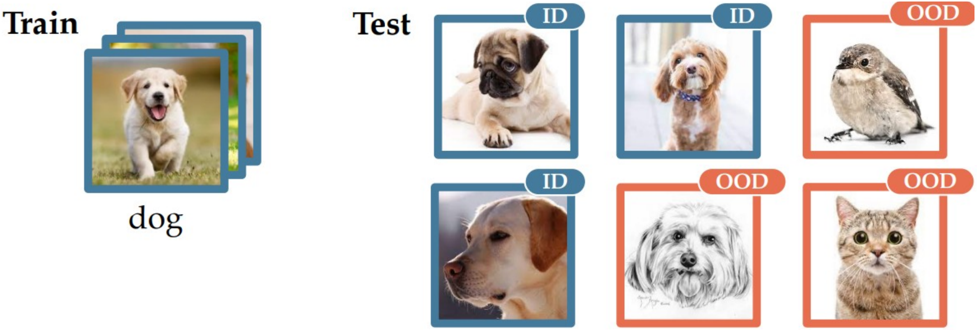
- □ training without MCMC sampling (vs EBM, diffusion)
- □ no mode collapse (vs GAN)
- □ exact likelihood (vs VAE, diffusion)
- □ fast generation (vs autoregressive, diffusion)
- □ moderate computational complexity (vs diffusion)

Weaknesses:

- □ inefficient use of capacity (vs EBM)
- □ large training footprint (vs EBM)
- □ no translational equivariance (vs EBM)

Anomaly detection: about anomalies

Definition: an observation which arouses suspicions of being unrelated to the process that generates training data [hawkins80book].



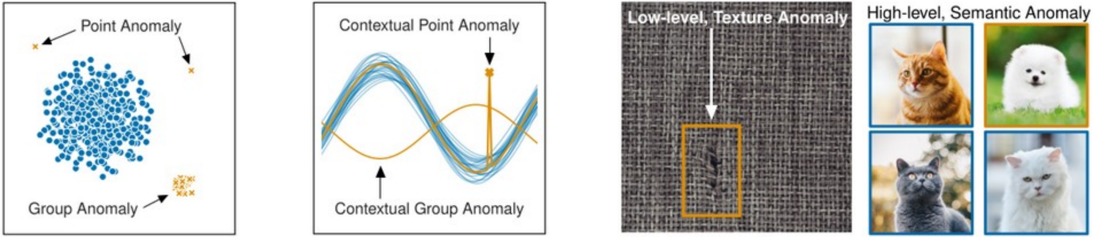
[yang21arxiv]

Anomalous data points are related (or also known as) outliers, out-of-distribution samples or novelties [ruff21pieee].

Anomaly detection: about anomalies (2)

There are several flavours of anomaly detection:

- □ pointwise vs groupwise
- □ contextual pointwise vs contextual groupwise
- □ low level (texture) vs high level (semantics)



[ruff21pieee]

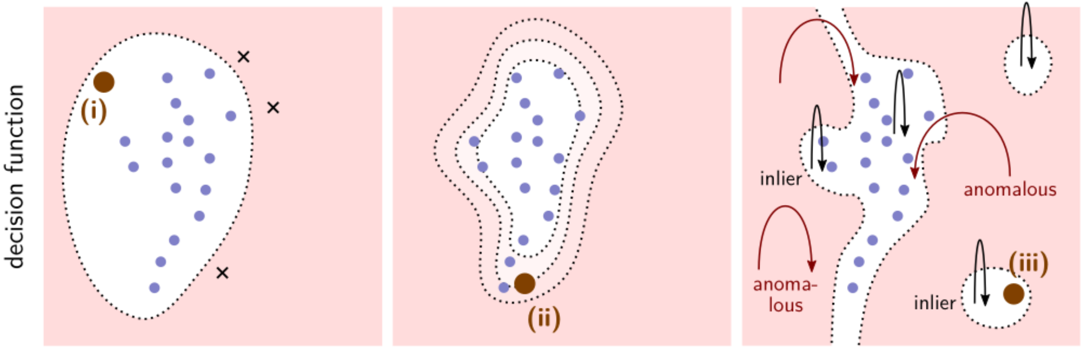
Anomaly detection: overview

Three main approaches to express anomaly score s of the sample $\mathbf{x}$

i) discriminative: arbitrary scalar $s_{\mathrm{cla}} = f(\mathbf{x})$

ii) generative: probability density $s_{\mathrm{pdf}} = p(\mathbf{x})$

iii) reconstructive: $s_{\mathrm{rec}} = \|\mathbf{x} - f_{\mathrm{dec}}(f_{\mathrm{enc}}(\mathbf{x}))\|$



[ruff21pieee]

Anomaly detection: overview (2)

Three main approaches to express anomaly score s of the sample $\mathbf{x}$

i) discriminative: arbitrary scalar $s_{\mathrm{cla}} = f(\mathbf{x})$

- □ problems: feature collapse, negative training data
- □ related to dataset posterior $P(\mathcal{D}_{\mathrm{in}}|\mathbf{x}) = \sigma(s_{\mathrm{cla}})$

ii) generative: probability density $s_{\mathrm{pdf}} = p(\mathbf{x})$

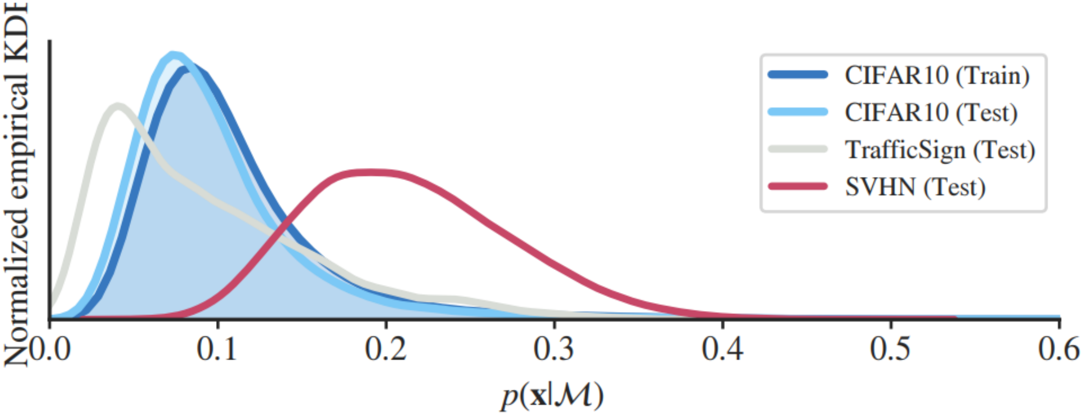
- □ problem: semantic anomalies (if applied to the data)
- □ problems: feature collapse (if applied to semantic features)

iii) reconstructive: $s_{\mathrm{rec}} = \|\mathbf{x} - f_{\mathrm{dec}}(f_{\mathrm{enc}}(\mathbf{x})\|$

- □ problems: generalization in inliers, "identity" shortcut
- □ related to dataset likelihood: $P(s_{\mathrm{rec}}|\mathcal{D}_{\mathrm{in}}, \mathbf{x}) \sim e^{-s_{\mathrm{rec}}^2}$

Anomaly detection: reality check

Direct application of estimated density (either flows or pixel-cnn) to detection of semantic outliers fails miserably:

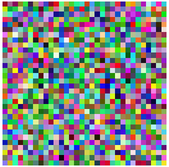


[serra20iclr]

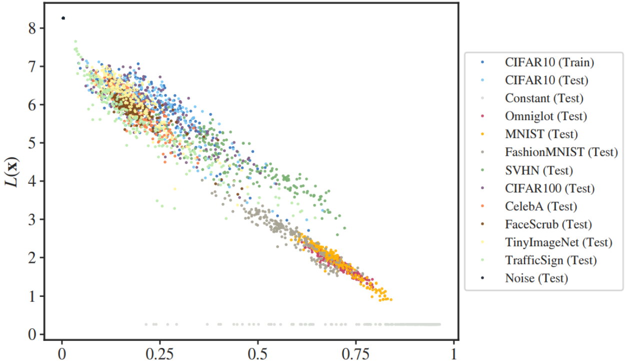
Anomaly detection: role of complexity

Recovered densities wildly depend on image complexity:

- □ consider images with lower compressed lengths $L(\mathrm{x})$
- □ e.g. MNIST, poliglot, constant (the simple ones)
- □ these images score higher in spite of being outliers



[krizhevsky09tr]

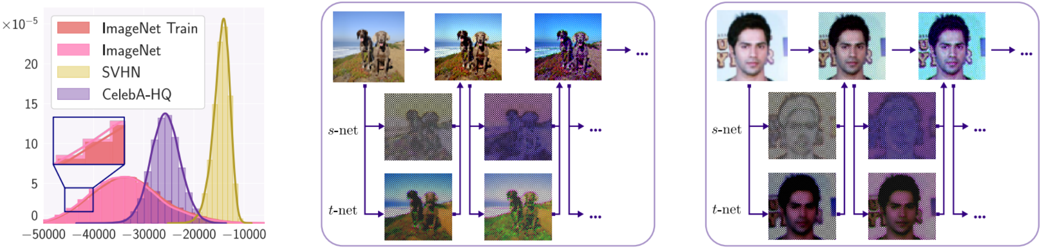


[serra20iclr]

Anomaly detection: inappropriate bias

Possible explanation: maximum likelihood training is unable to recover semantic anomalies since generative models know nothing about semantics.

- □ visualization of internal activations suggest a similar reaction for inliers and outliers
- □ however, they train only with generative loss $L = -\log p(\mathbf{x})$
- □ chances improve when sharing features with a discriminative task [zhang00eccv]



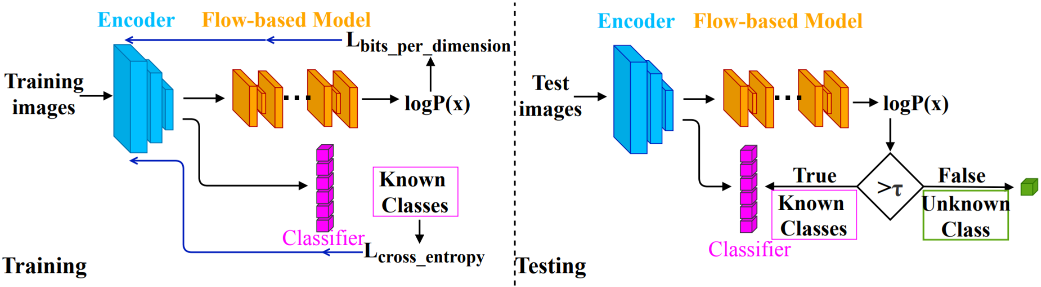
(a) Log-likelihoods (b) ImageNet input, in-distribution (c) CelebA input, OOD

[kirichenko20neurips]

Anomaly detection: hybrid open-set recognition

Open-set performance can be improved by evaluating likelihood of semantic features

- □ discriminative and generative predictions share the latent features
- □ the anomaly score corresponds to inverse density
- □ both losses affect the shared features (hybrid recognition)



[zhang20eccv]

Dense anomalies: generative aproaches

Detect regions with low pixel-level density:

- □ apply any generative model to 1x1 feature windows [blum21ijcv]
- □ find a way to train EBM without sampling - DenseHybrid [grcic22eccv]

Leverage generative modeling in non-density based approaches:

- □ detect reconstruction errors in the resynthesized image [lis19iccv]
- □ discriminative training with jointly generated synthetic negatives - NFlowJS [grcic21visapp, grcic21arxiv]

[https://segmentmeifyoucan.com/]

Dense anomalies: pixel-level density

Dense density estimation: recover probability density as if in a sliding window

Desireable properties: efficient inference, equivariance to translation

- □ VAE - **fast**, can be equivariant
- □ EBM - **fast**, **equivariant** (but very hard training)
- □ pixel-cnn - slow, not equivariant ("linear" factorization)
- □ flow - **fast**, not equivariant
- □ diffusion, score-based - slow, not equivariant
- □ GAN - fast, can be equivariant (but no explicit density)

Dense anomalies: sliding 1×1 window

Apply per-layer flows to frozen features of a standard semantic segmentation model

- □ embedding density [blum21ijcv]

Inference normalizes the feature likelihood with respect to the average layer likelihood:

$$\overline{N}(z_\ell^{(i)}) = \log p(z_\ell^{(i)}) - \frac{1}{N}\sum_k \log p(z_\ell^{(k)})$$

- □ normalized contributions are suitable for ensembling
- □ ℓ denotes the layer, i denotes the pixel:

Strength: combines principled density estimation with feature semantics

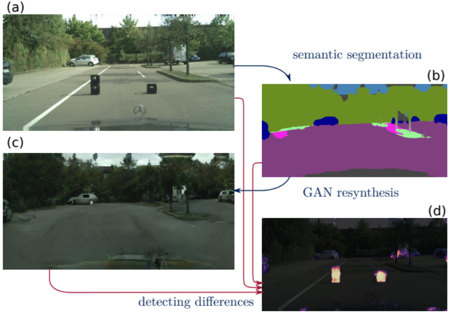
Weakness 1: vulnerability to feature collapse due to frozen features

Weakness 2: does not exploit negative training data

Dense anomalies: image resynthesis

Approach [lis19iccv, vojir21iccv, dibiase21cvpr]:

1. perform standard semantic segmentation
2. resynthesize input by generative image-to-image translation
3. detect anomalous pixels as reconstructions errors



[lis19iccv]

Strength: rather principled, can detect all kinds of anomalies

Weakness 1: diverse inliers and wrong predictions lead to false positive anomalies

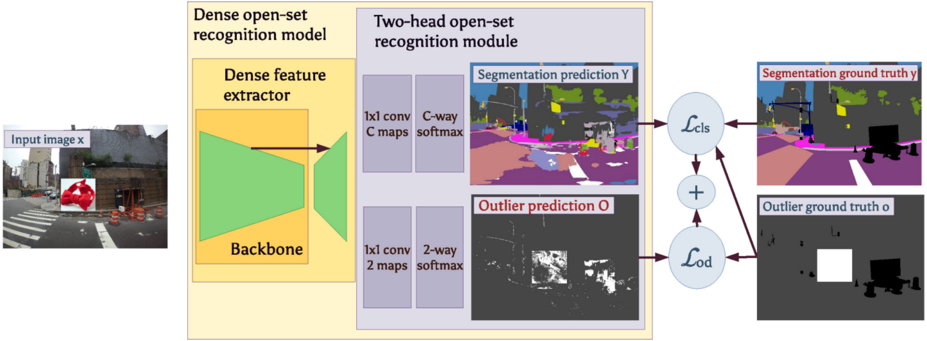
Weakness 2: works only on the road, fails in non-standard road pixels

Weakness 3: rather slow, unsuitable for real-time

Dense anomalies: two heads (both discriminative)

Idea: detect outliers with a discriminative prediction head

- craft mixed-content images by pasting noisy negatives into regular training images
- leverage negative images from the ImageNet dataset
- train the dense closed-set classifier only on positive pixels

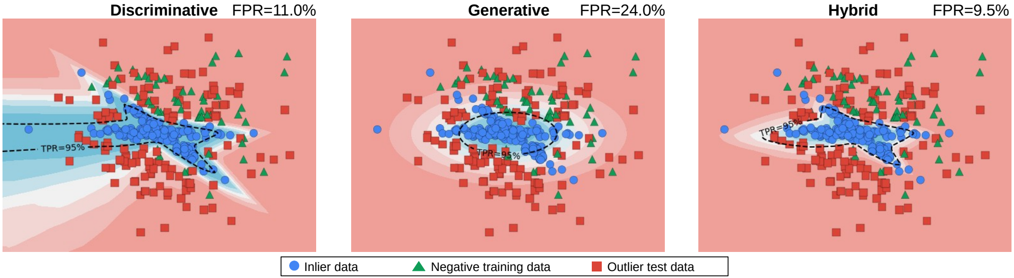


[bevandic22ivc]

Dense anomalies: dense hybrid: idea

Idea: improve the two-head approach with dense likelihood estimates

- □ pro: generative and discriminative anomaly detection exhibit different failure modes
- □ con: requires negative data (use case for sythetic negatives)
- □ con: most generative models are not translation equivariant
  - □ GANs and VAEs can not deliver exact density
  - □ hybrid EBM is a method of choice for this task



[grcic22eccv]

Dense anomalies: synthetic negatives

Training with pasted noisy negative samples produces great outlier detection performance

- □ hard to evaluate the performance
- □ some test anomalies may have been seen during training...

We wish to address this issue by replacing real negative samples with synthetic ones

Question: how to co-train a generative model in order to produce negative examples which could teach the discriminative model to better recognize anomalies?

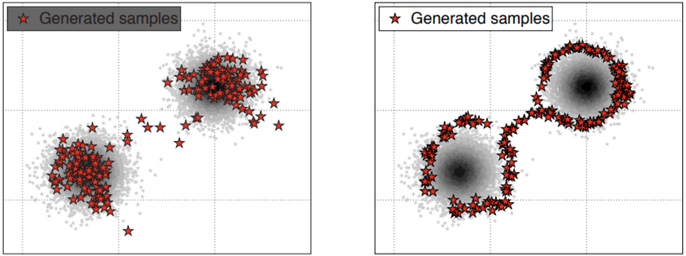
Dense anomalies: synthetic negatives (2)

We pose the following requirements on model parameters θ:

□ high data likelihood in inliers $p_\theta(\mathrm{x})$

□ high discriminative entropy in generated data $P(y|\mathbf{f}_\theta(\mathrm{z}))$

Such learning generates samples at the border of the training distribution [lee18iclr]



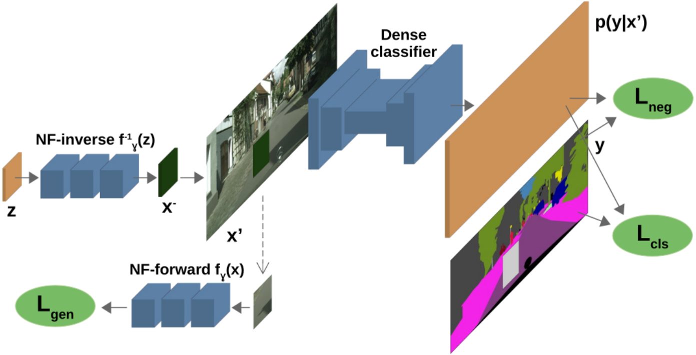
[lee18iclr]

The dicriminative model can be trained to predict high uncertainty in these samples!

Dense anomalies: synthetic negatives (3)

We adapt the joint learning scheme for dense prediction:

- □ we use a normalizing flow instead of GAN (arbitrary resolution, better coverage)
- □ we contribute a robust loss that accounts for generative noise
- □ we propose a suitable optimization procedure for joint learning



[grcic21visapp]

Conclusion

- Generative recognition has experienced a lot of exciting recent progress
  - we are proud of our systems although they are not intelligent in the strong sense.
- Image is a collection of easily counterfeited pixels
  - digital photographs should not be treated as a reflection of the reality
  - verification of integrity possible only in presence of cryptographic signatures
  - important implications for our society
- Semantic anomaly detection can not be properly addressed in absence of semantic supervision.
- Open-set recognition appears easier than four years ago
  - it is not unlikely that soon it will be considered as solved.

Thank you for your attention!

Questions?

This presentation would not have been possible without insightful ideas and hard work of Matej Grcić, Jakob Verbeek, Ivan Krešo, Marin Oršić, Petra Bevandić, Josip Šarić, Ivan Grubišić, Marin Kačan, Iva Sović, Nenad Markuš and Jelena Bratulić.

This research has been supported by Croatian Science Foundation (MULTICLOD, ADEPT), ERDF (DATACROSS, A-UNIT, SAFETRAM), NVidia Academic Hardware Grant Program, Rimac automobili, Microblink, Gideon brothers, Romb technologies, Promet i prostor, Končar, UniZg-FPZ, and VSITE.