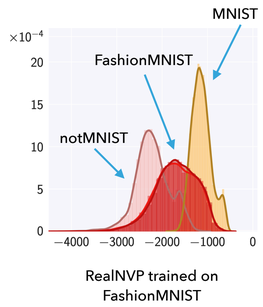
Why Normalizing Flows Fail to Detect Out-of-Distribution Data

Polina Kirichenko @polkirichenko
Pavel Izmailov @Pavel_Izmailov
Andrew Gordon Wilson @andrewgwils

- Normalizing flows often surprisingly fail to distinguish between in- and out-of-distribution data: *why*?
- Flows learn local pixel correlations and generic image-to-latent-space transformations not specific to the target image dataset
- Modifying the architecture of flow coupling layers, we can bias the flow towards learning the semantic structure of the target data, improving OOD detection



RealNVP trained on FashionMNIST

Out-of-distribution detection

- Goal: detect inputs that are different from the training data
- Important in many safety-critical applications
- Likelihood-based generative models are attractive for OOD detection: we can reject inputs that get low likelihood

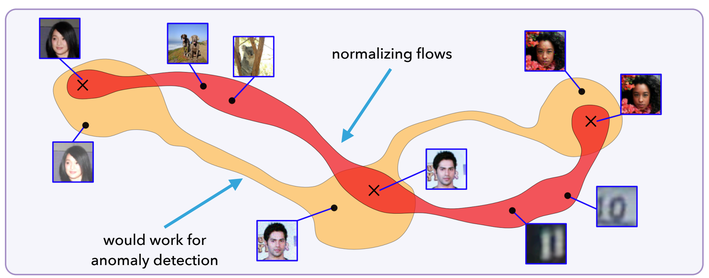
Normalizing flows

- Deep generative models based on invertible neural networks
- We can compute density of the data exactly via change of variables

$$z \sim p_Z \quad x = f_\theta^{-1}(z), \quad p(x) = p_Z(f_\theta(x)) \cdot \left| \det \left(\frac{\partial f_\theta}{\partial x} \right) \right|$$

Inductive biases

- Maximum likelihood training encourages solutions which concentrate all mass on training data, i.e. overfit
- The likelihood assignment outside training data will be determined by inductive biases of the model



Latent space

- There is a direct correspondence between image and latent representation coordinates, the input shape is often visible in latent representation

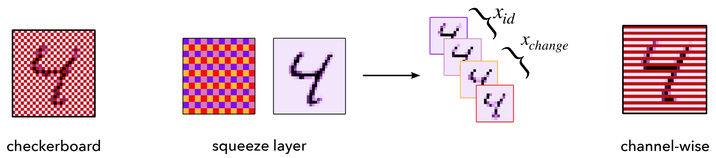
Inductive biases of flows

- Coupling layers:

$$f_{\text{aff}}^{-1}(x_{\text{id}}, x_{\text{change}}) = (y_{\text{id}}, y_{\text{change}}), \quad \begin{cases} y_{\text{id}} = x_{\text{id}} \\ y_{\text{change}} = (x_{\text{change}} + t(x_{\text{id}})) \odot \exp(s(x_{\text{id}})) \end{cases}$$

neural network e.g. ResNet

- x is split into x_{id} and x_{change} using masking:

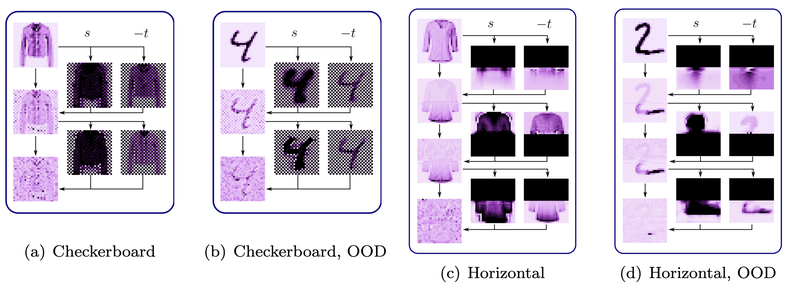


checkerboard squeeze layer channel-wise

What are coupling layers doing?

- Affine coupling layer predicts scale and shift which directly model masked pixels
- The objective pushes values s to be large while keeping the norm $\|z\|$ small: as a result $(-t)$ is approximating masked input and s represents the confidence of approximation

$$\log p_X(x) = \log p_Z(f^{-1}(x)) + \log \left| \det \frac{\partial f^{-1}}{\partial x} \right|$$
$$\log \mathcal{N}(z|0, I) + \sum_{i=1}^{dim(x_{change})} s(x_{id})_i$$

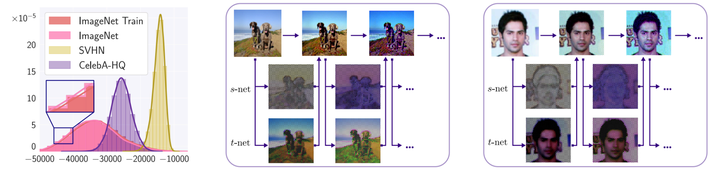


(a) Checkerboard (b) Checkerboard, OOD (c) Horizontal (d) Horizontal, OOD

- Horizontal mask should make it more challenging to predict masked pixels
- However, the layers learn to *co-adapt* and encode information to be used in subsequent layers

Bigger datasets

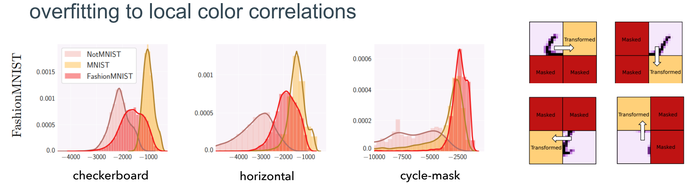
- We observe similar results for flow trained on ImageNet: coupling layers can predict inputs for both in-distribution and OOD data



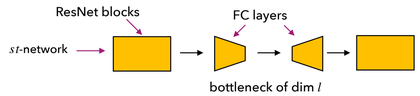
(a) Log-likelihoods (b) ImageNet input, in-distribution (c) CelebA input, OOD

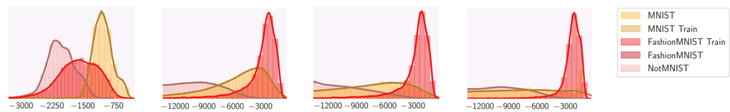
Changing inductive biases

- We change the masking strategy to avoid coupling layer coadaptation and overfitting to local color correlations



- We introduce low-dimensional bottlenecks to the st-networks
- The lower the dimension of the bottleneck, the better the ranking of OOD data

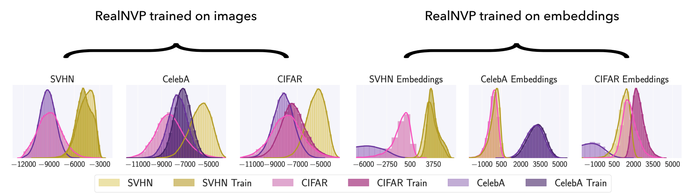




(a) Baseline (b) $l = 100$ (c) $l = 50$ (d) $l = 10$

Image embeddings

- Flows applied to embeddings extracted with a pre-trained CNN instead of raw pixels can detect OOD inputs



https://github.com/PolinaKirichenko/flows_ood