Fast, Effective and Interpretable Deep Learning

Frederik Hvilshøj

March 3, 2020

DEPARTMENT OF COMPUTER SCIENCE
AARHUS UNIVERSITY

Table of contents

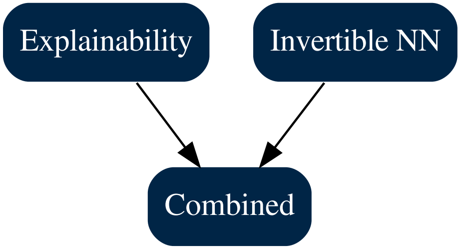
Introuction

Introduction

- Deep Learning is leading paradigm in Machine Learning
- Much effort put into network design and efficiency
- Models may be biased due to model constraints
- Limited insights – knowledge is based on intuition
- Explanations are important for sensitive tasks

This Project

- Explanations of general neural network is challenging
- Invertible functions give a higher level of insights



Normalizing Flows

Normalizing Flows

Goal: Learn a representation of a density $p(x)$

Model: Maximize log-likelihood of the data with respect to the parameters of an invertible Neural Network f.

$$\log(p_\theta(x)) = \log(p_z(f(x))) + \log(|det\frac{\partial f(x)}{\partial x}|)$$

Constraints: invertible f and tractable Jacobian.

$$log\left(|\det\frac{\partial f_k(f_{k-1}(\ldots f_1(x)))}{\partial x}|\right) = \sum_{i=1}^{k}\log\left(|\det\frac{\partial f_i(z_{i-1})}{\partial z_{i-1}}|\right)$$

$$z_0 = x, \quad z_i = f_i(z_{i-1})$$

Normalizing Flow Layers

Coupling layer $c(x) = (x_1, x_2 + m(x_1))$, for $x = (x_1, x_2)$ [4]

Channel permutation Fixed channel permutations of 3D data [5]

1x1 convolutions Learnable channel interactions [9]

Periodic convolutions Circular convolutions over each channel [6]

What if Neural Networks had SVDs?

The Singular Value Decomposition

Definition

The Singular Value Decomposition of $W \in \mathbb{R}^{m \times n}$ is a factorization $W = U\Sigma V^T$, where U is an orthogonal $m \times m$ matrix, Σ is an $m \times n$ rectangular diagonal matrix, and V is an $n \times n$ orthogonal matrix.

For our purposes, assume that W is a square, i.e., $W \in \mathbb{R}^{d \times d}$. Then:

$$U^{-1} = U^T \text{ and } V^{-1} = V^T \qquad O(d^2)$$

$$W^{-1} = (U\Sigma V^T)^{-1} = V\Sigma^{-1}U^T \qquad O(d^2)$$

$$det(W) = \prod_{i=1}^{d} \Sigma_{i,i} \qquad O(d)$$

Other fast computations are largest singular value, condition number, matrix exponential, and Cayley Map.

Use in Neural Networks

Neural Networks

Recall a simple fully-connected layer:

$$h_\ell = \sigma(Wh_{\ell-1}) = \sigma\left(U\Sigma V^T h_{\ell-1}\right)$$

- Normalizing flows
- Recurrent Neural Networks
- W-GAN

Gradients

$$W^{t+1} = W^t - \eta \frac{\partial L}{\partial W}$$
$$\Sigma^{t+1} = \Sigma^t - \eta \frac{\partial L}{\partial \Sigma}$$
$$U^{t+1} = U^t - \eta \frac{\partial L}{\partial U}$$

The Householder Matrix

Definition

For a vector $v \in \mathbb{R}^d$, the corresponding Householder matrix is defined as

$$H = I - 2\frac{vv^T}{||v||_2^2} \tag{1}$$

- Note LHS of Eqn. (1) takes $O(d^2)$ but RHS takes $O(d)$.
- H stays orthogonal under gradient descent on v

$$UX = H_1(H_2(\cdots(H_dX)))$$

with $X \in \mathbb{R}^{d\times m}$, where m is the batch size.

If we represent H_is as matrices, this takes $O(d^3m)$ time (parallel [10]).

We can easily improve by representnig H_i by its vectors v_i instead, $O(d^2m)$ (sequential [10]).

Issues

Issues:

- The parallel algorithm is slow in theory.
- The sequential algorithm is not suited for GPUs.

We propose:

- Algorithm which is suited for GPUs and fast in theory.

Our Algorithm

Lemma

[3] For any m Householder matrices $H_1, \ldots, H_m$ there exists $W, Y \in \mathbb{R}^{d\times m}$ st.

$$I - 2WY^T = H_1 \cdots H_m.$$

Both W and Y can be computed by m sequential Householder multiplications in $O(dm^2)$ time.

H_1	$\cdots$	H_m	$\cdots$	H_{2m}	$\cdots$	H_d

$P_1 = I - 2W_1Y_1^T$ $\quad P_2 \quad \cdots \quad P_{\frac{d}{m}}$

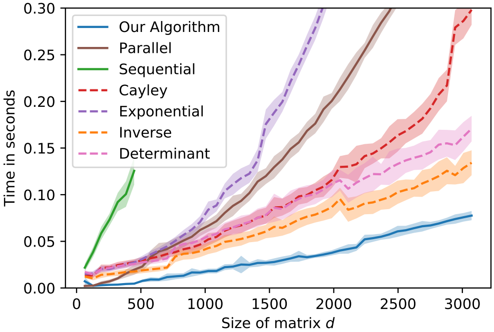
Complexity

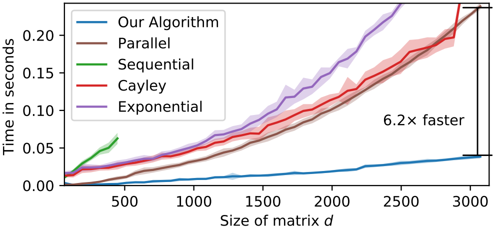
Time:

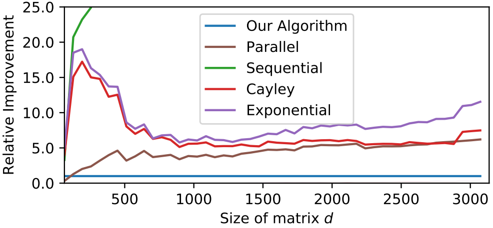
$$P_1 \cdot P_2 \cdots P_{\frac{d}{m}} X \qquad O(d^3 m) \Rightarrow O(d^2 m)$$

sequential opetations:

$$O(\text{compute } I - 2WY^T + \text{multiply } X \text{ with } P_i\text{s}) = O(m + d/m)$$

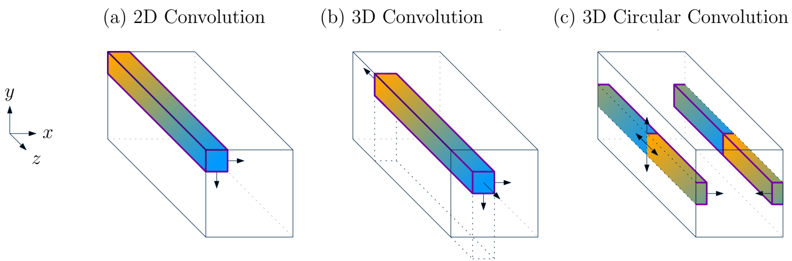

Our Algorithm
Parallel
Sequential
Cayley
Exponential
Inverse
Determinant
Time in seconds
Size of matrix d


Our Algorithm
Parallel
Sequential
Cayley
Exponential
6.2× faster
Time in seconds
Size of matrix d


Our Algorithm
Parallel
Sequential
Cayley
Exponential
Relative Improvement
Size of matrix d

Circular 3D Convolutions

3D Circular Convolutions



Algorithm: Forward $O(hwc \log hwc)$, Determinant $O(hwc)$

Require: Image/activations $X \in \mathbb{R}^{h \times w \times c}$ and kernel $K \in \mathbb{R}^{h \times w \times c}$.

1: $X = \mathrm{DFT}_{3D}(X)$

2: $K = \mathrm{DFT}_{3D}(K)$

3: $X = X \odot K$

4: $X = \mathrm{DFT}_{3D}^{-1}(X)$

Comparison with Periodic Convolutions

For $X, K \in \mathbb{R}^{h \times w \times c}$ and $W \in \mathbb{R}^{c \times c \times h \times w}$

Circular 3D connvolutions

$$Z = F_{3D}^{-1}(F_{3D}(X) \odot F_{3D}(K))$$

Periodic convolutions

$$z_i = F_{2D}^{-1}(\sum_{j=1}^{c} F_{2D}(W_{i,j}) \odot F_{2D}(X_{:,:,j})),$$

Type	Params	Forward	Inverse
Periodic	$h \cdot w \cdot c^2$	$O(hwc^2)$	$O(hwc^3)$
3D-circular†	$h \cdot w \cdot c^2$	$O(hwc^2 \log(hwc))$	$O(hwc^2 \log(hwc))$

Improving Variational Auto-Encoders

Improving Variational Auto-Encoders

Variational lower-bound:

$$\log p(x) \geq \mathbb{E}_{z \sim q(z|x)} [\log p(x|z)] + D_{KL}(q(z|x)||p(z))$$

where $q(z|x) = \mathcal{N}(\mu(x), diag(\sigma(x)))$.

With formalizing flow f, the equation becomes

$$\log p(x) \geq \mathbb{E}_{z \sim p(z|x)} \left[\log p(x|f(z)) + \log |Det \frac{\partial f(z)}{\partial z}| \right] + D_{KL} \left(q(z|x)||p(f(z)) \right)$$

[11] let $f(z) = H_T H_{T-1} \cdots H_1 z$ with each H_i being parametrized by the encoder.

Improving Variational Auto-Encoders

Method	$\leq \ln p(\mathbf{x})$
VAE	-93.89 ± 0.09
VAE+HF(T=1)	-87.77 ± 0.05
VAE+HF(T=10)	-87.68 ± 0.06
VAE+NF (T=10) [17]	-87.5
VAE+NF (T=80) [17]	-85.1
VAE+NICE (T=10) [7]	-88.6
VAE+NICE (T=80) [7]	-87.2
VAE+HVI (T=1) [20]	-91.70
VAE+HVI (T=8) [20]	-88.30

Figure 1: Comparison of the lower bound of marginal log-likelihood measured in nats of the digits in the MNIST test set. For the first three methods the experiment was repeated 3 times. Direct copy from [11].

Explainability

Explanations

Here we focus on explainability, characterized by

> *An active characteristic of a model, denoting any action or procedure taken by a model with the intent of clarifying or detailing its internal functions [1].*

We consider attribution techniques and counterfactual explanations.

- Identifies contribution of each input feature wrt. the output
- "Gradient-like" computations
- Based on different propagation rules

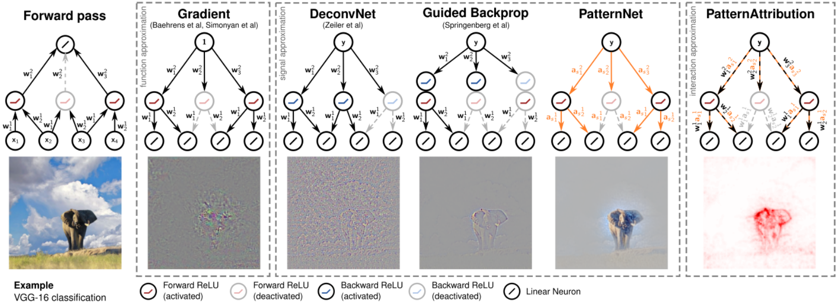


Figure 2: Direct copy from [8].

Counterfactual Explanations

Counterfactual explanations tries to answer the question

> *How can I make a minimal and realistic change to an input of the model such that the predicted outcome changes?*

Common strategy

1. Start from the original input
2. Until prediction change
   2.1 Make small change to the input
   2.2 Query the classifier to gain information

Differences

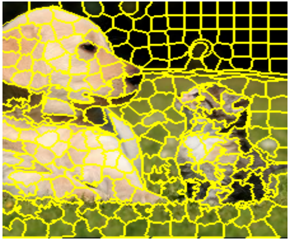
- How to do update
- Information gained from classifier

Improving Explanations with Probabilistic Saliency Estimation

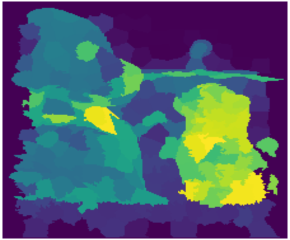
Probabilistic Saliency Estimation [2]

Input ($X \in \mathbb{R}^{h \times w \times c}$)

Super-pixels



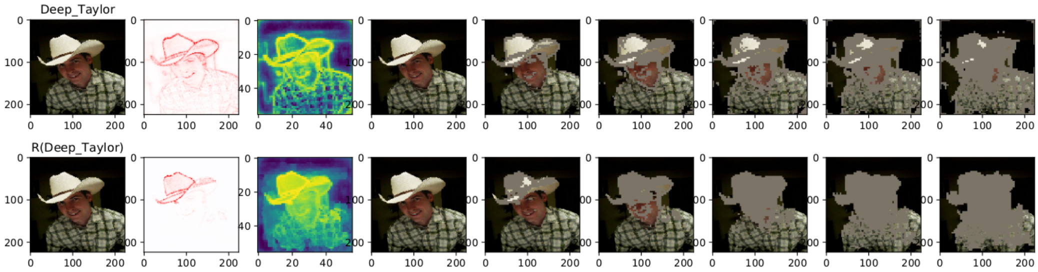
$P \in \mathbb{R}^{h \times w \times c}$



Let $A \in \mathbb{R}^{h \times w \times c}$ be an attribution map, then we propose a refined attribution

$$\hat{A} = A \odot P$$

Experiment



- Fill-methods
- Evidence agains classes
- Larger tests

Attention Mechanisms in Gradient Explanations

Guiding backward computations

A simple dense layer $f_\ell : L' \to L$ of a Neural Network with backpropagation function $g_\ell : L \to L'$ of explanation a_ℓ is

$$f_\ell(h_{\ell-1}) = \sigma(h_{\ell-1}^T W_\ell) \qquad\qquad g_\ell(a_\ell) = a_\ell^T \frac{\partial \sigma(h_{\ell-1} W_\ell)}{\partial h_{\ell-1} w_\ell} W^T$$

with $W \in \mathbb{R}^{L' \times L}$. We experimented with guiding the backpropagation

$$\hat{a}_{\ell-1} = g_\ell(a_\ell) \odot \alpha_\ell(h_{\ell-1}, a_\ell)$$
$$\alpha_\ell(h_{\ell-1}, a_\ell) = \tanh\left([h_{\ell-1} || a_\ell] L_\ell R_\ell\right)$$

with $L_\ell \in \mathbb{R}^{(L'+L) \times d}$ and $R_\ell \in \mathbb{R}^{d \times L'}$, for $d < \frac{L \cdot L'}{L+L'}$

Extensions:

- Change the input to the α_ℓs
- Scale the output to be in range $[-c, c]$

Learning L and R

- Need to definde a loss
- DeconvNet and Guided Backprop are class insensitive

$$L(X, \hat{X}) = ||\hat{X}||_1 + \lambda \mathcal{L}(y, \hat{y}),$$

Extensions:

$$L(X, \hat{X}) = \mathcal{L}(y, \hat{y}) + \lambda_1 ||\hat{X}||_1 + \lambda_2 ||X - \hat{X}||_2 + \lambda_3 vec(\hat{X})^T L vec(\hat{X})$$

$$f^T L f = \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} a_{i,j} (f_i - f_j)^2$$

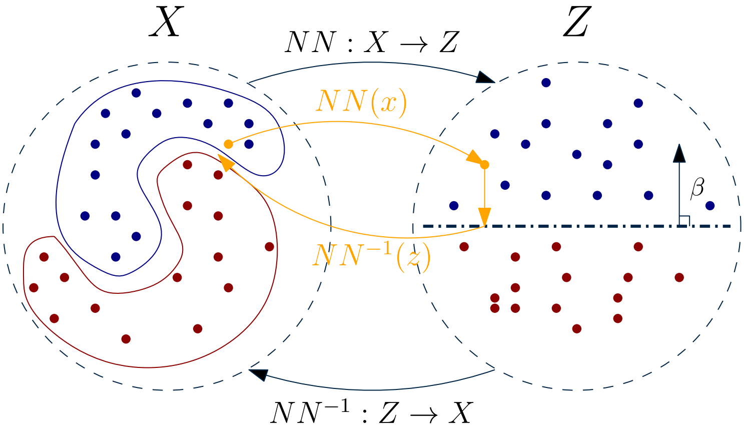
Challenges

- Currently only potentially works with DeconvNet, GBP, and PatternNet
- Loss for attribution methods?
- Are these explanations from the in-sample-distribution?

$$\text{AOPC} = \frac{1}{L+1} \left\langle \sum_{k=0}^{L} f\left(x_{\text{MoRF}}^{(0)}\right) - f\left(x_{\text{MoRF}}^{(k)}\right) \right\rangle_{p(x)}$$

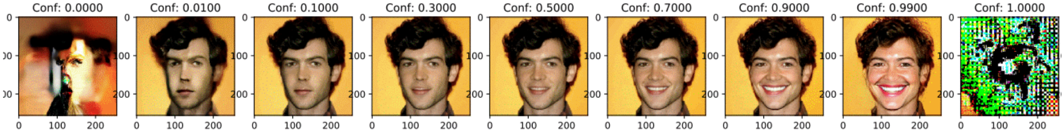
Generating Counterfactual Explanations

Construction



$$f(x) = \sigma\left(\beta^T NN(x)\right) \qquad f^{\dagger}(y) = NN^{-1}\left(f(x) - \frac{\langle f(x), \beta\rangle \beta}{||\beta||}\right)$$

Properties and Extensions



Properties

- First "one-evaluation-solution" (Fast)
- Built to produce counterfactual explanations

Extensions

- Training NN and β jointly
- Evaluating Counterfactual explanations
- Bound difference in X from change in Z space
  - Lipschitz constraint on NN

$$Score(X, \hat{X}) = ||\Delta_X||_1 + vec(\Delta_X)Lvec(\Delta_X) + \mathbf{1}_{f(\hat{X}) \neq \bar{f}(\hat{X})}$$

References i

[1] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *arXiv preprint arXiv:1910.10045*, 2019.

[2] Caglar Aytekin, Alexandros Iosifidis, and Moncef Gabbouj. Probabilistic saliency estimation. *Pattern Recognition*, 74:359–372, 2018.

[3] Christian Bischof and Charles Van Loan. The WY Representation for Products of Householder Matrices. *SIAM Journal on Scientific and Statistical Computing*, 1987.

[4] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-Linear Independent Components Estimation. In *ICLR (Workshop)*, 2015.

[5] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *ICLR*, 2019.

[6] Emiel Hoogeboom, Rianne van den Berg, and Max Welling. Emerging Convolutions for Generative Normalizing Flows. In *ICML*, 2019.

[7] Jörn Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. In *ICLR*, 2019.

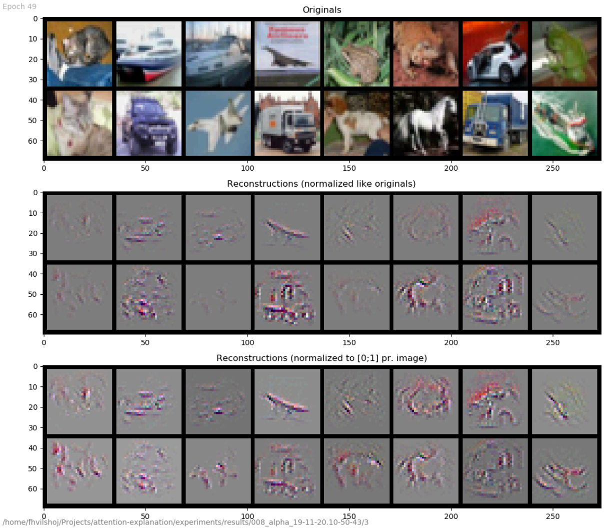
[8] Pieter Jan Kindermans, Kristof T. Schütt, Maximilian Alber, Klaus Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: Patternnet and Patternattribution. In *ICLR*, 2018.

[9] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions. In *NeurIPS*, 2018.

[10] Zakaria Mhammedi, Andrew Hellicar, Ashfaqur Rahman, and James Bailey. Efficient Orthogonal Parametrisation of Recurrent Neural Networks Using Householder Reflections. In *ICML*, 2017.

[11] Jakub M. Tomczak and Max Welling. Improving Variational Auto-Encoders using Householder Flow. *arXiv preprint arXiv:1611.09630*, 2016.

Attention example



Attention example

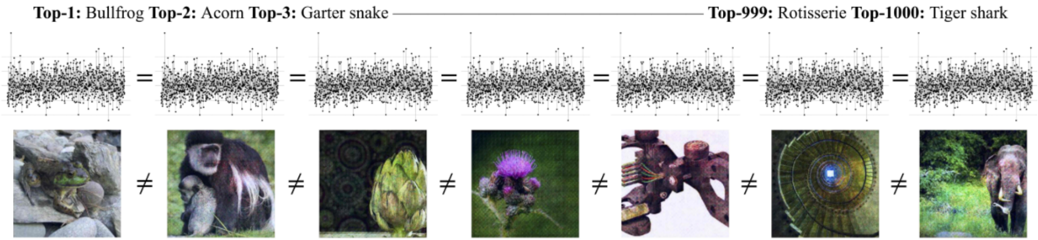


Figure 3: Direct copy from [7].