

Explaining neural networks

Moritz Wolter

March 13, 2023

High-Performance Computing and Analytics Lab

Overview

Linear classifiers

Interpretation by examination

Case Study: Deepfake detection

Input Optimization

The problem with deep CNN

Saliency Maps and Integrated Gradients

Literature

Code Snippets

Motivation

- Neural networks are potent black-box methods.
- Some very deep convolutional neural networks have hundreds of layers and use up to 600mb of disk storage.
- Let's do what we can to open the box!!

Linear classifiers

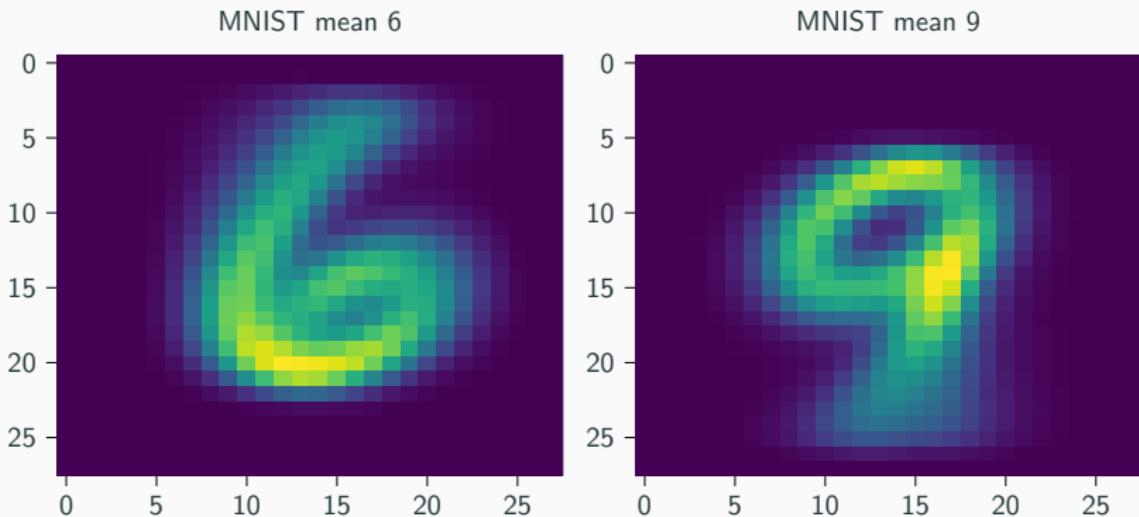
Definition of a linear classifier

Linear classifiers consist of a dense layer without an activation,

$$\mathbf{o} = \mathbf{Ax} + \mathbf{b}. \quad (1)$$

With $\mathbf{A} \in \mathbb{R}^{m,n}$, $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$. Linear only works on simple problems that are linearly separable.

Binary MNIST



Cross-Entropy

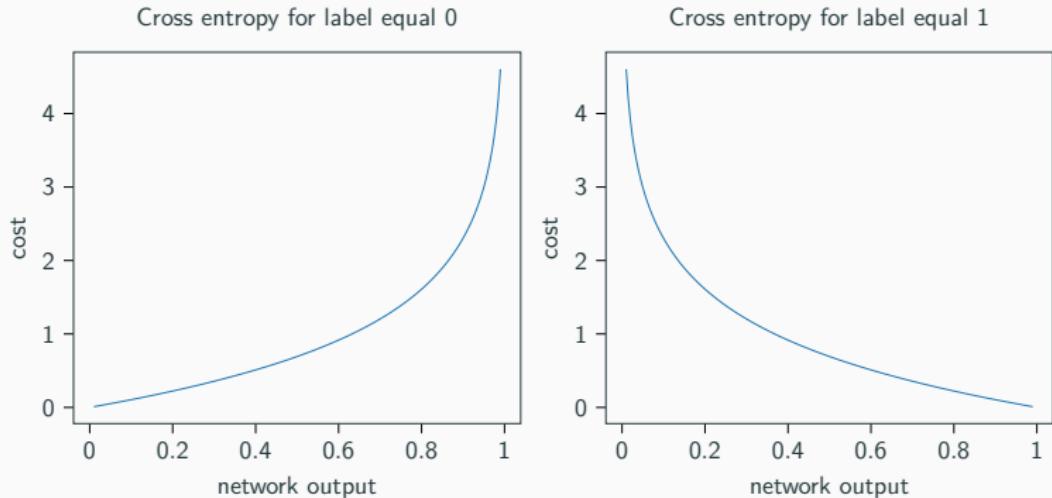
Recall the definition of the cross entropy

$$C_{\text{ce}}(\mathbf{y}, \mathbf{o}) = - \sum_k^{n_o} (\mathbf{y}_k \ln \mathbf{o}_k) + (\mathbf{1} - \mathbf{y}_k) \ln (\mathbf{1} - \mathbf{o}_k). \quad (2)$$

With $\mathbf{y}, \mathbf{o} \in \mathbb{R}^{n_o}$ defined as vectors of length n_o .

Cross-Entropy

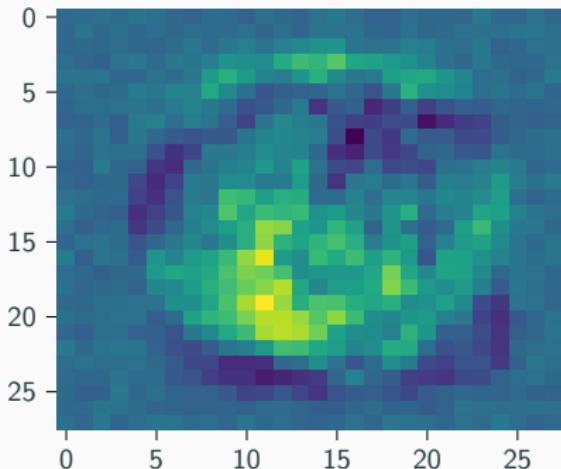
To understand what is going on lets consider the two cases $y_k = 0$ and $y_k = 1$.



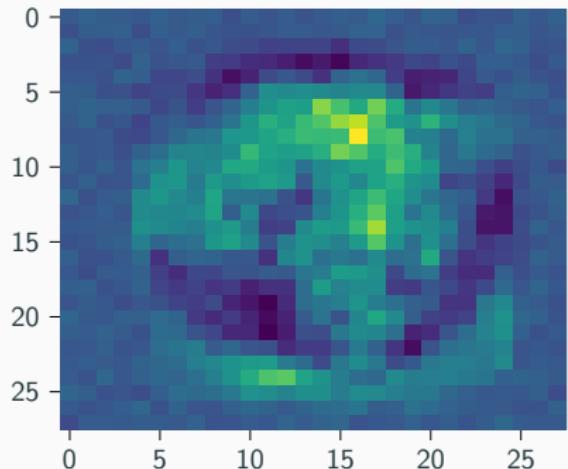
Cross-entropy pushes the output towards the label.

Interpretation by examination

MNIST binary classifier 6



MNIST binary classifier 9



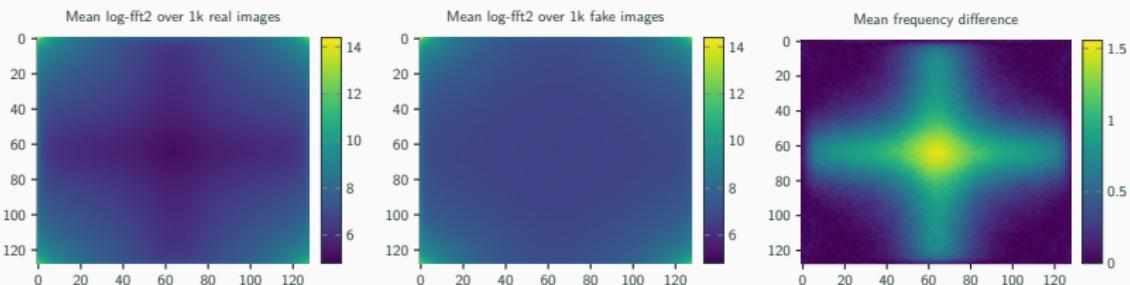
If linear is possible linear is great!!

What deep-fakes are

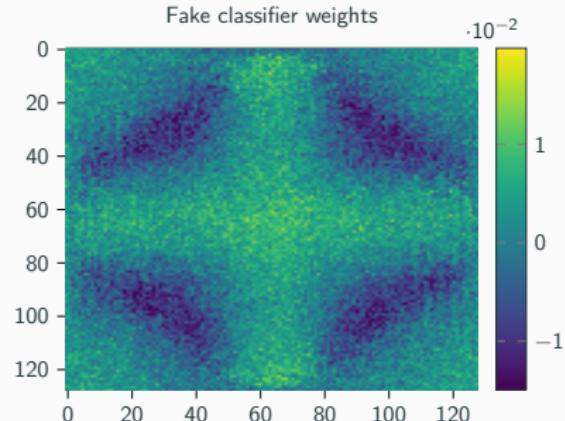
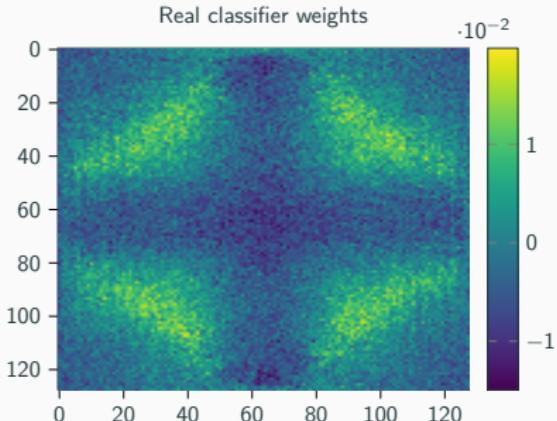
Generative models can generate images. Consider the samples below:



What blows the con?



Fake detectors



StyleGAN-generated fakes can be classified with around 99% accuracy this way [Fra+20].

Summary

- For linearly separable binary problems weight inspection works great!
- Engineered features allow the inspection to reveal something about the data.

Input Optimization

Motivation

Let's be honest. Most linearly separable binary problems are academic.

An input CNN-Layer

Kernel-shape: (3, 3, 1, 32)

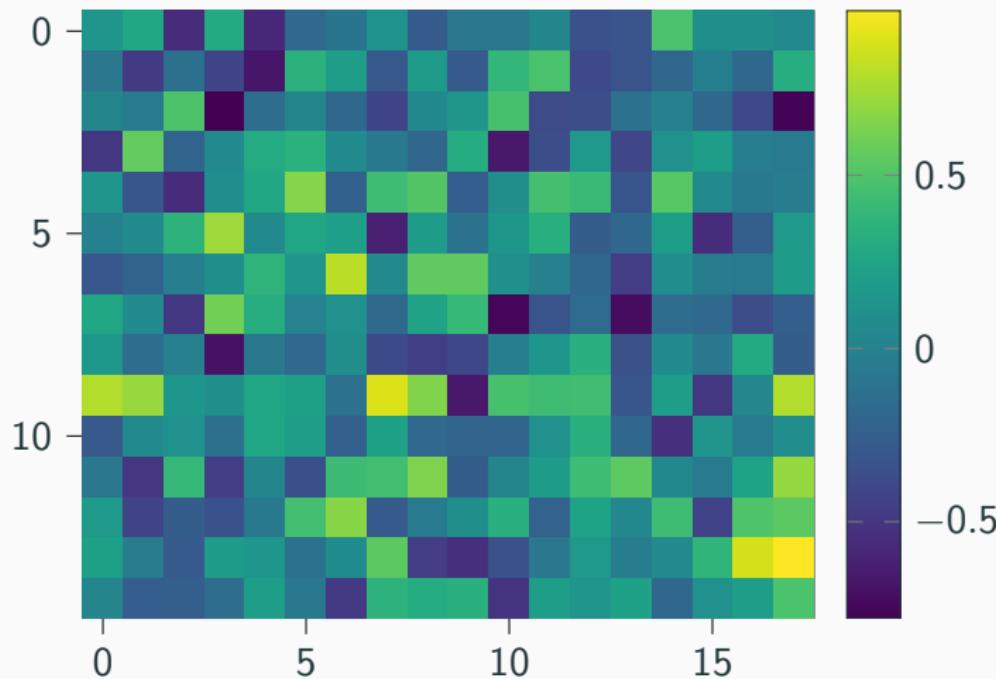


Figure: Plot of the input layer kernel weights trained on MNSIT.

Motivation Reloaded

- How do we verify the correct operation of deep nonlinear networks?
- It is *very* hard to interpret the weights of deep networks directly.
- Unit tests would require extensive re-training.

How to make a single neuron extremely happy

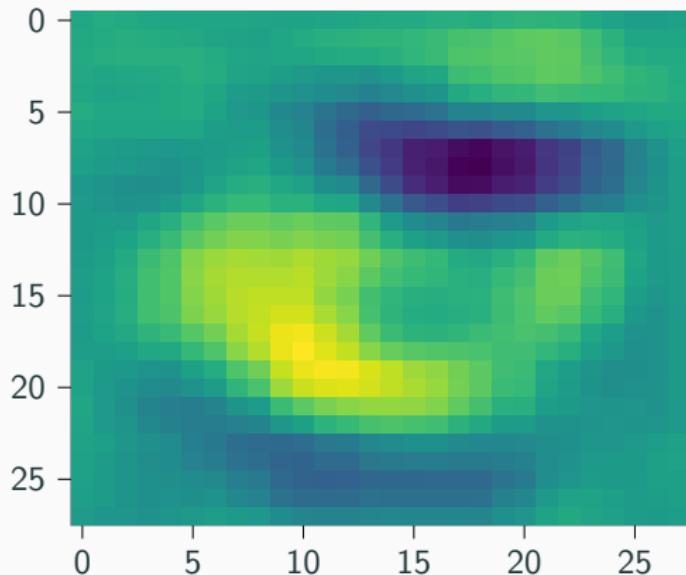
What if we turned the optimization problem around and optimized the input instead of the weights? Consider

$$\max_{\mathbf{x}} y_i = f(\mathbf{x}, \theta), \quad (3)$$

with network weights θ , input \mathbf{x} , and y_i , the activation of the i -th output neuron!

The 6-neurons favorite input

Starting from $\mathbf{x} = \mathbf{1} \in \mathbb{R}^{1,28,28,1}$ using image $\mu\text{-}\sigma$ -normalization after every step with a step size of one and a positive update yields:



Saliency Maps

[SVZ13] tell us to optimize

$$\arg \max_{\mathbf{I}} S_c(\mathbf{I}) - \lambda \|\mathbf{I}\|_2^2. \quad (4)$$

With S_c , the classification score for class c. \mathbf{I} is the input image, and $\|\mathbf{I}\|_2$ represents the 2-norm image channels. λ is a regularization parameter.

CNN-Saliency Map

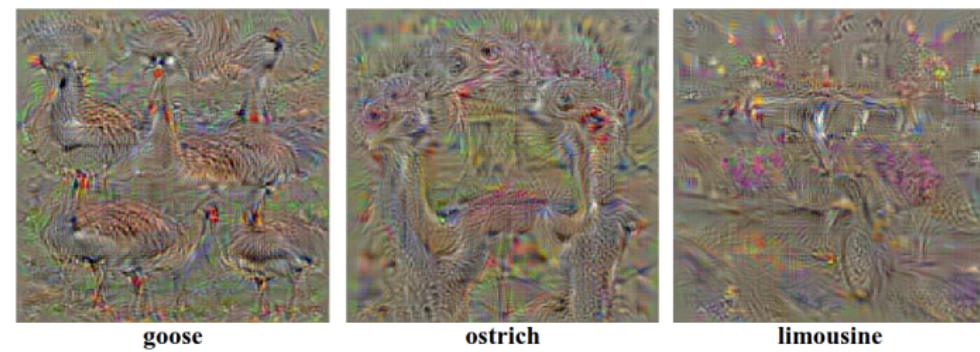


Figure: Input optimization saliency maps as shown in [SVZ13].

Integrated-gradients

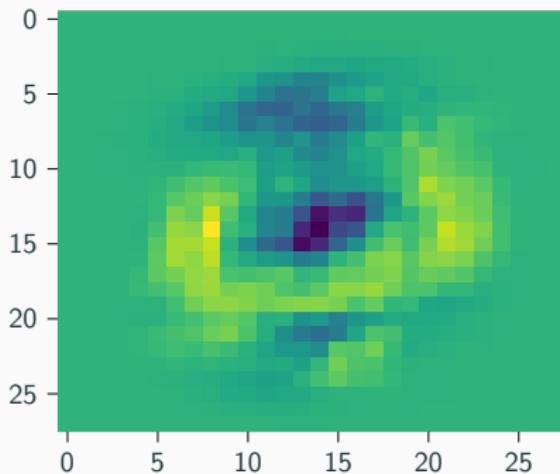
[STY17] propose to estimate individual input contributions to an output neuron via,

$$\text{IntegratedGrads}_i(x) = (x_i - x'_i) \cdot \sum_{k=1}^m \frac{\partial F(x' + \frac{k}{m} \cdot (x - x'))}{\partial x_i}. \quad (5)$$

$\frac{\partial F}{\partial x_i}$ denotes the gradients with respect to the input color channels i . x' denotes a baseline black image. And x symbolizes an input we are interested in. Finally, m denotes the number of summation steps from the black baseline image to the interesting input.

Integrating the gradients of a multilayer CNN on MNIST

Integrated gradients for the zero neuron on the MNIST-validation set.



Images from the Wild

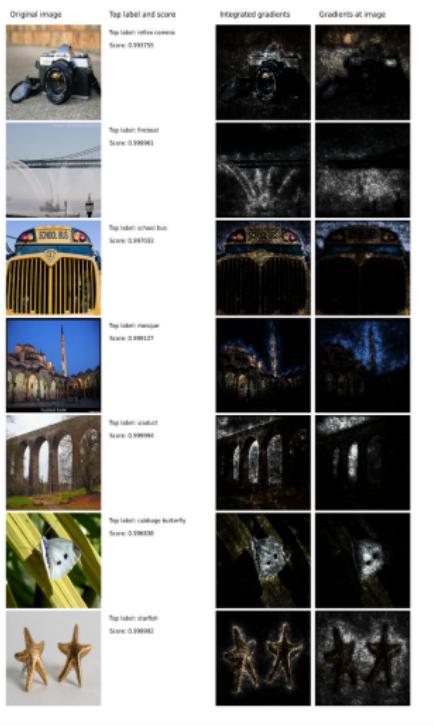


Figure: Integrated gradient visualization of input saliency for a very deep-CNN trained on Imagenet [Den+09]. Image taken from [STY17].

Conclusion

- Before you do anything else, look at the openai microscope at <https://microscope.openai.com/models>.
- We can look at features and weights and work with input optimization to understand what is going on.

Literature

References

- [Den+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A large-scale hierarchical image database.” In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

Literature ii

- [Fra+20] Joel Frank, Thorsten Eisenhofer, Lea Schönherr, Asja Fischer, Dorothea Kolossa, and Thorsten Holz. “Leveraging frequency analysis for deep fake image recognition.” In: *International conference on machine learning*. PMLR. 2020, pp. 3247–3258.
- [SVZ13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep inside convolutional networks: Visualising image classification models and saliency maps.” In: *arXiv preprint arXiv:1312.6034* (2013).

- [STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan.
“Axiomatic attribution for deep networks.” In:
International conference on machine learning. PMLR.
2017, pp. 3319–3328.

Code Snippets

Input optimization

```
neuron = 6
def forward_pass(x):
    out = net.apply(weights, x)
return out[0, neuron]
get_grads = jax.value_and_grad(
    forward_pass)
```

Using slurm

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --job-name=mnist
#SBATCH --output=mnist-%j.out
#SBATCH --error=mnist-%j.err
#SBATCH --gres gpu:1
#SBATCH --cpus-per-task=6
#SBATCH --time=1:00:00
#SBATCH --mem=50gb
source /home/YOURUNIID/.bashrc
conda activate base
python src/mnist.py
```