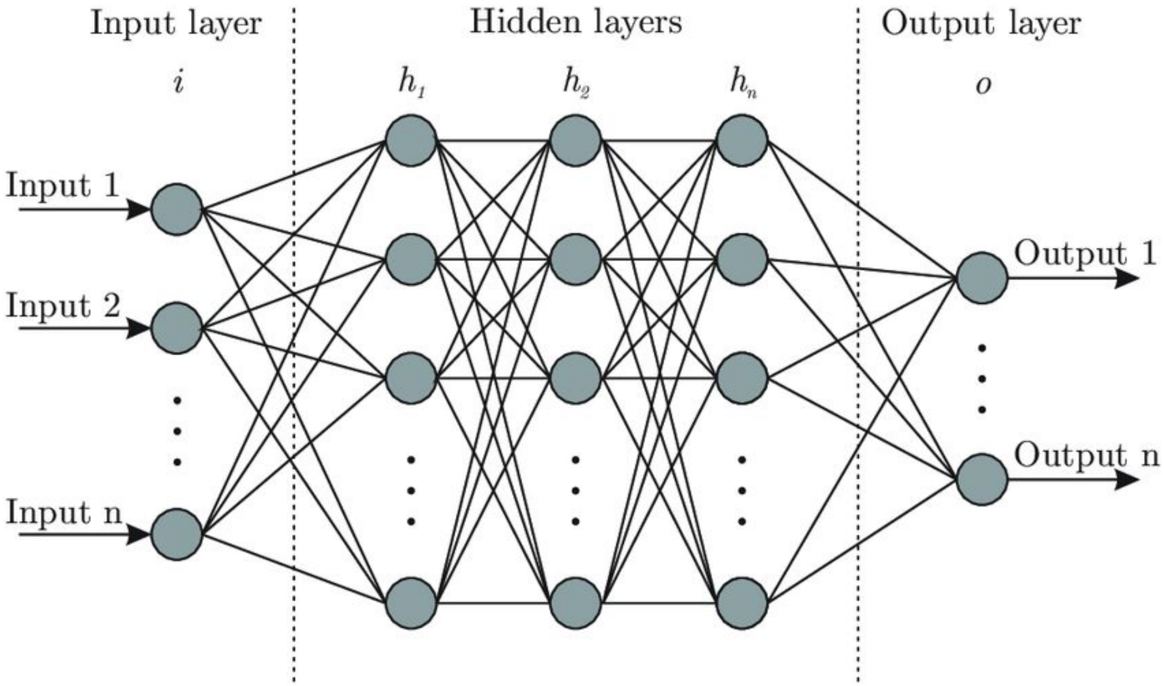
Lecture 5. Explaining Neural Networks

counterfactuals · adversarial examples · prototypes · influential instances

Tatiana Makhalova
MADE, 16.10.2021

Part 1. Neural networks. Basics

Artificial neural networks

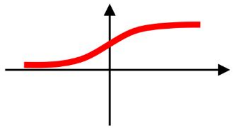
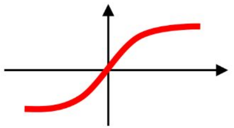
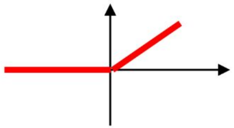


Neurons and its activation

A neuron has the following form

$$z_j = \sum \omega_{ij} x_i + b_j$$

Some of activation functions $\varphi(z_j)$ for **hidden** layers:

Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent (tanh)	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	

Sigmoid and tanh suffer from **saturation** problem. The functions are only really sensitive to changes around their mid-point of their input, such as 0.5 for sigmoid and 0.0 for tanh. They also suffer from the **vanishing** gradient problem.

ReLU is free from these disadvantages

Last-layer activations and loss functions

Problem Type	Last-layer activation	Loss function
binary classification	sigmoid	binary cross entropy
multiclass, single-label classification	softmax	categorical cross entropy
multiclass, multilabel classification	sigmoid	binary cross entropy
regression to arbitrary values	-	MSE
regression to values between 0 and 1	sigmoid	MSE / binary cross entropy

Loss vs cost functions

Generally cost and loss functions are synonymous but cost function can contain regularization terms in addition to loss function, although it is not always necessary:

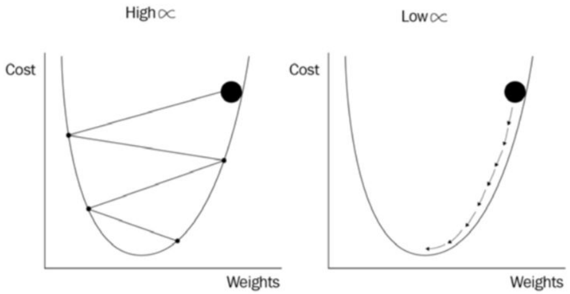
$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} |w_i|}$$

loss function

cost function

Network training. Main steps

1. Draw a **batch** of training samples x and corresponding targets y
2. **Forward pass**: run the network on x to obtain predictions y_{pred}
3. Compute the **loss** of the network on the batch, a measure of the mismatch between y_{pred} and y
4. **Backward pass**: compute the gradient of the loss with regard to the network's parameters
5. Move all weights of the network towards anti-gradient (that slightly reduces the loss on this batch) with a chosen **step**



Batch, mini-batch, and stochastic gradient descent

Let us consider a simple linear regression $y = \boldsymbol{\theta}^\top \boldsymbol{x}$

```
def gradientDescent(X, y, learning_rate = 0.001, batch_size = 32, n_epoch = 10):

    theta = 0
    for itr in range(n_epoch):
        mini_batches = create_mini_batches(X, y, batch_size)
        for mini_batch in mini_batches:
            X_mini, y_mini = mini_batch
            theta = theta - learning_rate * gradient(X_mini, y_mini, theta)
            error_list.append(cost(X_mini, y_mini, theta))
 return theta, error_list
```

Depending on the batch size, it can be:

- True SGD: `batch_size = 1`
- Mini-batch SGD: `1 < batch_size < #objects (often, 32)`
- Batch SGD: `batch_size = #objects`

Optimizers: setting a suitable batch size

- stochastic gradient descent: drawing a single instance
- mini-batch stochastic gradient descent: drawing a subset of instances
- batch stochastic gradient descent: using of the whole training set

Rules of thumb:

1. mini-batches of size 32 might be a good default
2. tune batch size and learning rate after all other hyperparameters
3. the size can be tuned by checking up learning curves (training and validation error vs amount of training time)
4. for BGD use relatively relatively larger learning rate and more training epochs, for SGD use a relatively smaller learning rate and fewer training epochs

Blueprinting a training process

1. Define a problem and get a dataset
2. Select suitable evaluation metrics (to monitor on validation data)
3. Decide an evaluation protocol
4. Prepare your data
    a. use values between 0 and 1
    b. have roughly the same ranges for all features
    c. perform an additional normalization (with mean 0 and std 1)
    d. do some feature engineering if you don't have enough features
    e. use a special number, e.g., 0, for missing values so that a NN learn to ignore them
    f. Develop a model that is better than a basic baseline
5. Develop a model that overfits
6. Regularizing your model and tuning your hyperparameters

Chollet, Francois. Deep learning with Python. Simon and Schuster, 2017.

Tackling with overfitting

To define the capacity of you model try to create one that overfits by

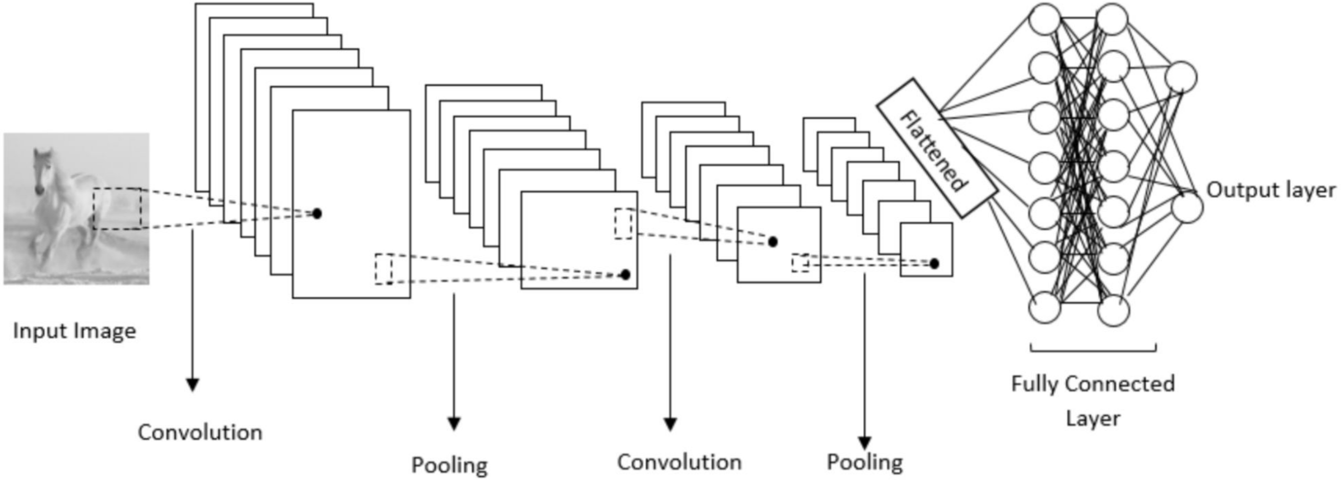
- adding layers
- making the layers bigger
- adding more epochs

Then just remove excessive elements…

To regularize your model:

- add dropout (to break neuron coalitions)
- get more data (including “data augmentation”)
- add weight regularization (L1 and L2, check the previous lecture)
- reduce the network capacity

Architecture of the CNN

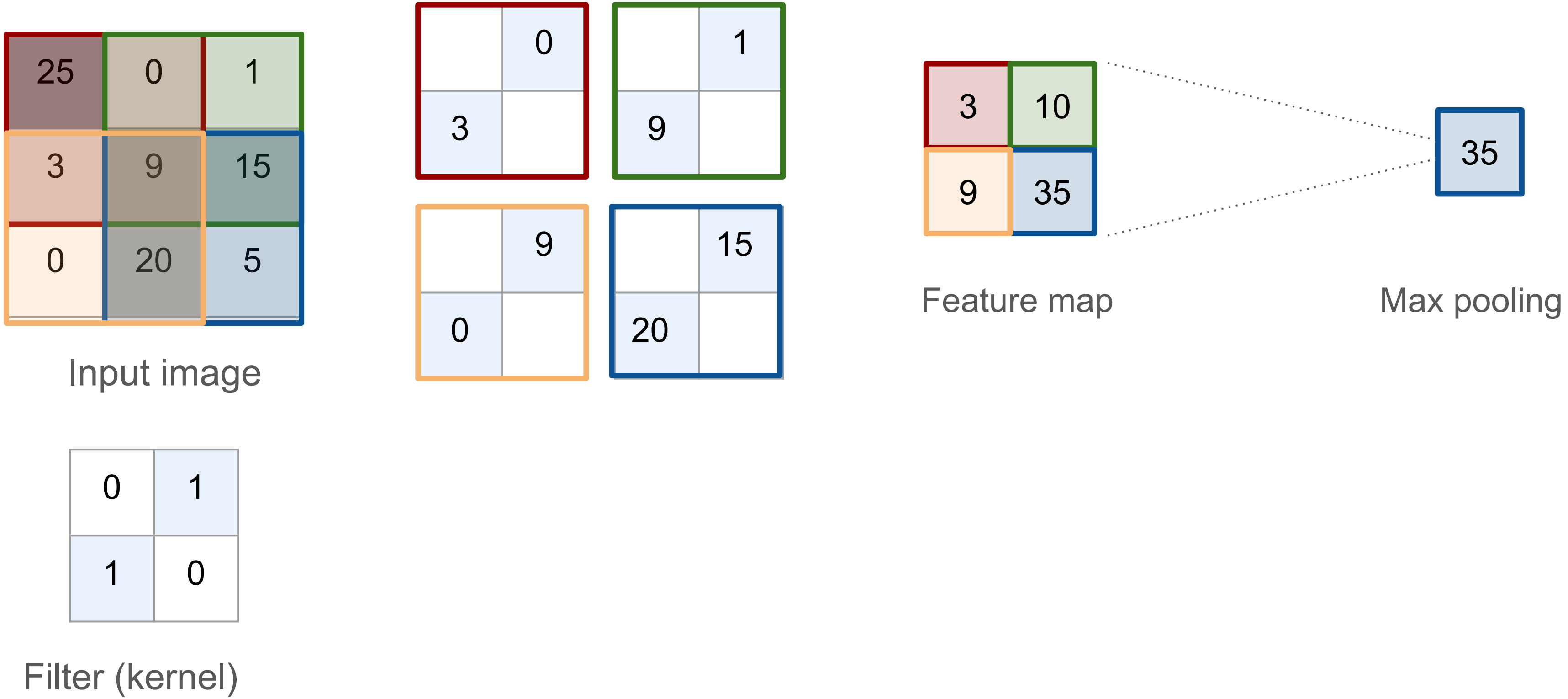


Motivation: dense layers learn global patterns; while convolutional layers learn local patterns
Key parameters: the **size** of the patches extracted from the inputs (kernel size), **depth** of the output feature map
Other parameters: strides and padding.

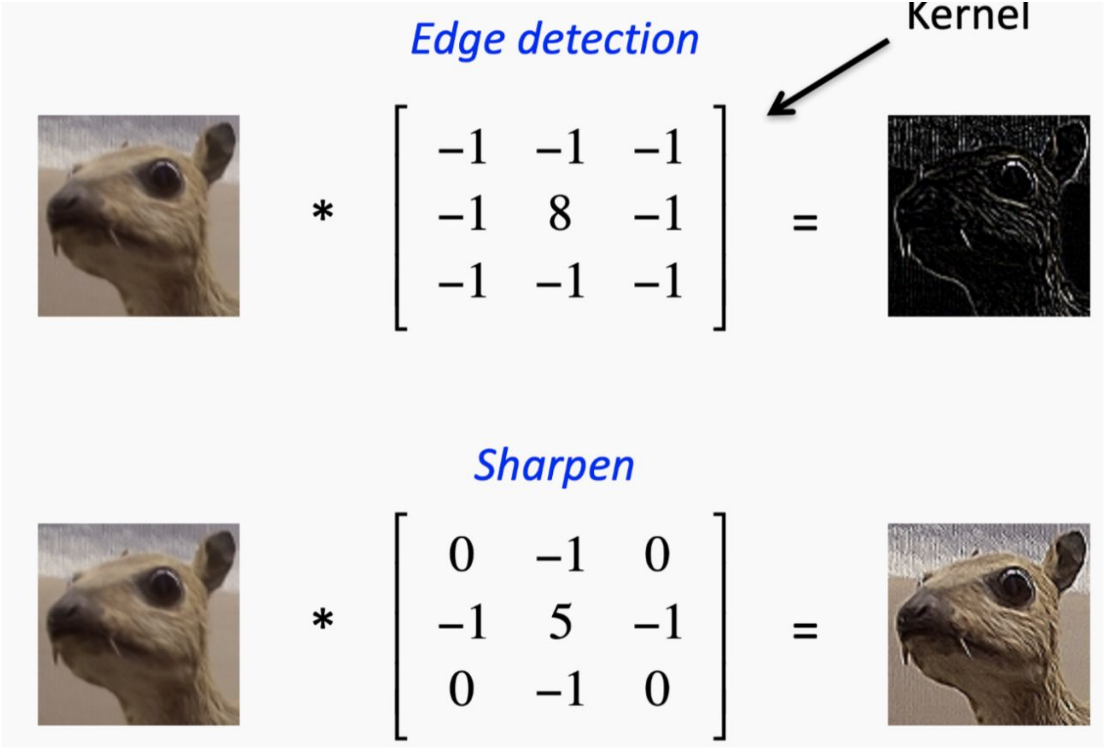
To understand better the aforementioned parameters, check https://poloclub.github.io/cnn-explainer/

Picture source: Deep RL (book)

Convolution operation



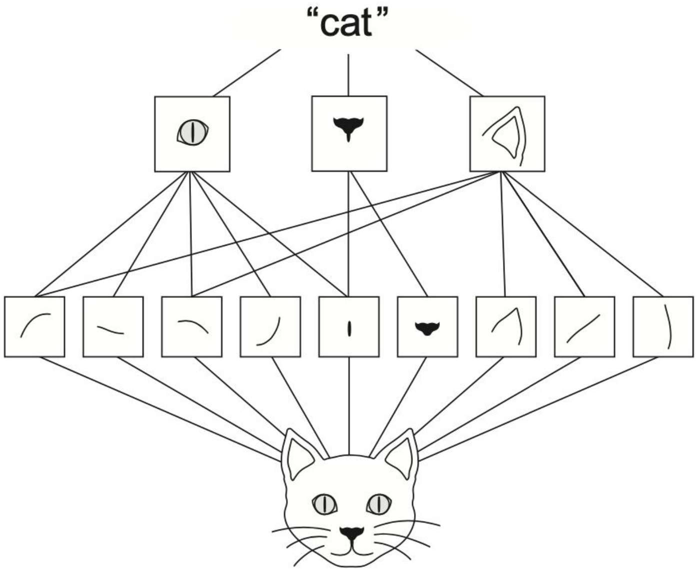
Convolution operation



Credit: https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac

Activation visualization

Idea: to see how the the layers are activated for a given image



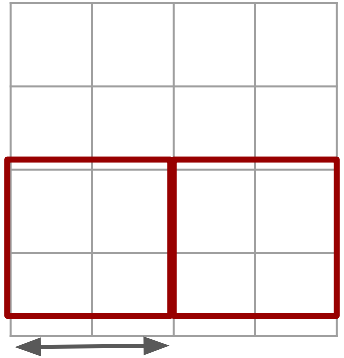
It is known that **lower layers** capture the elementary patterns, e.g., the horizontal, vertical, diagonal lines, textures, etc

High layers capture more complicated patterns, e.g., eyes, ears, noses, etc

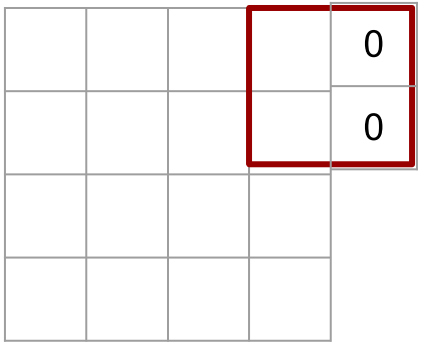
Picture source: François Chollet, Deep Learning with Python

Application of filters

Strides

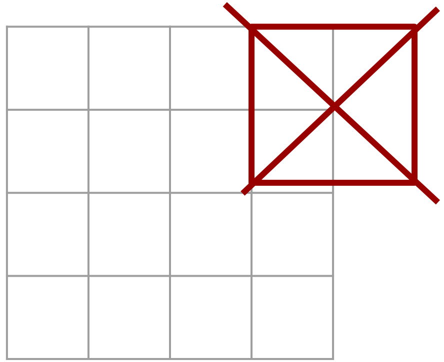


stride = 2



same (zero) padding

the padding ensures that the output has the same shape as the input data



valid (no) padding

ConvNets: recap

- **Convolutional layers:** searching local patterns
  - application of kernels and an activation function
- **Pooling layers:** dimension reduction
  - maximum pooling (better)
  - average pooling

Understanding ConvNets (in computer vision)

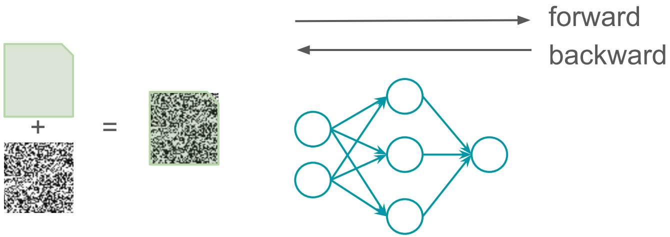
Main questions to study

- **How** ConvNets see a dataset?
  - visualization of feature maps / filters
- **What** is a model of a given class?
  - obtaining a “typical image” for a class
- **Why** a given image is classified as an instance of a certain class?
  - identification of image fragments that most affect the ConvNet output

Approaches for attributing importance to the input example

Perturbation-based approaches

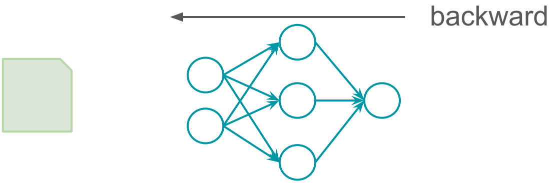
Introducing perturbations to individual inputs or neurons and observe the impact on later neurons or the output



1. Requires 2 passes
2. May underestimate the **importance of features** that have **saturated** their contribution to the output

Backpropagation-based approaches

Propagating back to the input neurons the output of the neural network



Part 2. Example-based explanation

counterfactual explanations · adversarial examples · prototypes · influential instances

Part 2.1. Counterfactual examples

About different counterfactuals...

Causal inference:

x_1	x_2	...	x_n	y
4	6	...	3	1

$do(X_2 = 5)$

x_1	x_2	...	x_n	y
4	**5**	...	3	**?**

Explanation methods:

x_1	x_2	...	x_n	y
4	6	...	3	1

Let y = 0. What is the smallest (δ_1, δ_2 ..., δ_n)?

x_1	x_2	...	x_n	y
4 + δ_1	6+ δ_2	...	3+ δ_n	**0**

Aamodt, Agnar, and Enric Plaza. "Case-based reasoning: Foundational issues, methodological variations, and system approaches." AI communications 7.1 (1994): 39-59

Intuition behind the counterfactual examples

The models where the **reasoning** is **based on examples**:

- very complex problems that are difficult to formalize, e.g., medicine:

  “I do perform action A because in my past in similar situations performing A worked well”

- formal models: K nearest neighbours

Aamodt, Agnar, and Enric Plaza. "Case-based reasoning: Foundational issues, methodological variations, and system approaches." AI communications 7.1 (1994): 39-59

Counterfactual explanation

A **counterfactual explanation** of a prediction describes the **smallest change** to the feature values that changes the prediction **to a predefined output**”[1]

Counterfactual explanation: what are the smallest changes should to be done to that the model change its decision?

Where it can be very useful: credit scoring, price for housing, image classification

Example:

x_1	x_2	...	x_n	y
4	6	...	3	1

counterfactual →

x_1	x_2	...	x_n	y
$4 + \delta_1$	$6 + \delta_2$	...	$3+ \delta_n$	**0**

Christoph Molnar Interpretable, “Counterfactuals”, Machine Learning, A Guide for Making Black Box Models Explainable

Basic settings

$\boldsymbol{x} \in D \in \mathbb{R}^{n \times d}$ an instance from the dataset described in d-dimensional space

$y = f(\boldsymbol{x})$ the answer of the model (class label, real value)

$x' = \boldsymbol{x} + \boldsymbol{\delta} \notin D$ a perturbed instance which is close to $\boldsymbol{x}'$

y' a predefined (desired) prediction for $\boldsymbol{x}'$ such that $y' \neq y$

Goal: to find an instance of x' quite close to x such that the answer to the model be quite close to the predefined one, i.e., we need **to find the smallest changes to change the output of the model**

Desired properties for counterfactual examples

A counterfactual instance should

- produce the **predefined prediction** as **closely** as possible
- be as **similar** as possible to the original **instance** by feature values
- change as **few features** as possible
- have feature **values** that are **likely**

The method should generate multiple diverse counterfactual explanations so that the decision subject gets access to **multiple** viable **ways** of generating a different **outcome**

Related issues

- What is the **best** counterfactual explanation? **Which feature**(s) to choose?
- How to find it? Is it always possible to find $x' \approx x$ such that $f(x') \approx y'$
- What is "allowable" distance between original instance x and the counterfactual one, i.e., x'?
- Is the counterfactual instance always possible to find? (the problem of *out of distribution -- OOD --* where we cannot get high-confident results for the model, see *epistemic uncertainty* for further detail)
- Rashomon effect: each counterfactual tells you a different story

Two-objective criterion by Wachter et al.

$$L(x, x', y', \lambda) = \lambda \cdot \boxed{(\hat{f}(x') - y')^2} + \boxed{d(x, x')} = \lambda \cdot L_{pred} + L_{dist}$$

x' produces the predefined prediction as closely as possible

x' is as similar as possible to the instance regarding feature values

$$d(x, x') = \sum_{j=1}^{p} \frac{|x_j - x'_j|}{MAD_j}$$

we use the L1 distance to ensure that a small number of instances will be selected (see the Lasso regression for the reference)

$$MAD_j = \text{median}_{i \in \{1,\dots,n\}}(|x_{i,j} - \text{median}_{l \in \{1,\dots,n\}}(x_{l,j})|)$$

Wachter, Sandra, Brent Mittelstadt, and Chris Russell. "Counterfactual explanations without opening the black box: Automated decisions and the GDPR." (2017)

Desired properties for counterfactual examples

A counterfactual instance should

- produce the **predefined prediction** as **closely** as possible $(f(x') -- y')^2$
- be as **similar** as possible to the **instance** regarding feature values $d(x',x)$
- change as **few features** as possible ***L1 distance for d***
- have feature **values** that are **likely**

The method should generate multiple diverse counterfactual explanations so that the decision subject gets access to **multiple** viable **ways** of generating a different **outcome**

Alibi

Let's slightly adjust the objective. Let x_0 be the instance to explained, and $x = x_0+\delta$ be the counterfactual example to be found

The objective to be minimized is $L(x_0, \delta, \lambda) = \lambda \cdot L_{pred} + L_{dist}$

It is the same except for a slight adjustment in the losses

$L_{dist} = \beta||\delta||_1 + ||\delta||_2^2 = \beta L_1 + L_2$ (instead of "lasso" we "elastic net" regularization)

$L_{pred} = \max([f(x_0 + \delta)]_{y_0} - \max_{i \neq y_0} [f(x_0 + \delta)]_i, -\kappa)$ (where κ cas the divergence between $[f(x_0+\delta)]_{y0}$ and $[f(x_0+\delta)]_i$)

Van Looveren, Arnaud, and Janis Klaise. "Interpretable Counterfactual Explanations Guided by Prototypes." arXiv preprint arXiv:1907.02584 (2019)

κ

What is L_{pred}?

In the models generating counterfactual, we speak about the **classification** models.

f outputs the probability of x_0 to belong to class y_0 (i.e., the predicted by f class for x_0)

$$L_{pred} = \max(\boxed{[f(x_0+\delta)]_{y_0} - \max_{i \neq y_0}[f(x_0+\delta)]_i} - \kappa)$$

- The difference in probabilities to belong to the class of x_0 and the largest probability among the rest of classes

What is L_{pred}?

$$L_{pred} = \max(\boxed{[f(x_0 + \delta)]_{y_0}} - \boxed{\max_{i \neq y_0} [f(x_0 + \delta)]_i}, -\kappa)$$

Example: let κ = 0.3

f, probability to belong to a class	P(in 1)	P(in 2)	P(in 3)	P(in 4)
x_0	0.86	0.10	0.03	0.01
$x_0 + \delta_1$	**0.71**	**0.20**	0.02	0.07
$x_0 + \delta_2$	**0.30**	0.01	**0.65**	0.04

max(0.71 - **0.20,** -0.3) = 0.51

max(0.30 - **0.65,** -0.3) = -0.30

Remark: we minimize the distance only if the perturbed instance $x_0 + \delta$ is too similar (in the terms of the class probabilities) to the class of x_0

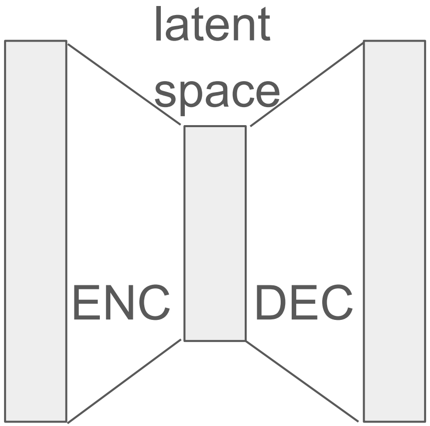
Alibi

What if $x_0 + \delta$ does not respect the training data manifold (i.e., out-of-distribution)?

$$L(x_0, \delta, \lambda) = \lambda \cdot L_{pred} + L_{dist} + L_{AE}$$

where $L_{AE} = \gamma \cdot ||x_0 + \delta - AE(x_0 + \delta)||_2^2$ reconstruction loss evaluated by autoencoder

The autoencoder learns a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore insignificant data ("noise")

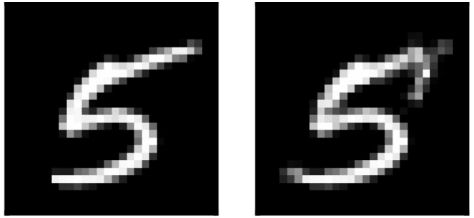


Alibi

$$L(x_0, \delta, \lambda) = \lambda \cdot \boxed{L_{pred}} + \boxed{L_{dist}} + \boxed{L_{AE}}$$

- $x = x_0 + \delta$ close to a class different from $y_0 = f(x_0)$
- $x_0 + \delta$ and x_0 are close enough, i.e., $\beta||\delta||_1 + ||\delta||_2^2$ is small
- $x_0 + \delta$ is close to the training data manifold

Example:

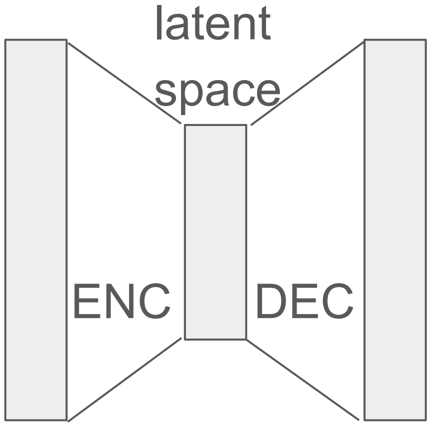


An instance of “5” and its counterfactual
Credit: Van Looveren, Arnaud, and Janis Klaise. "Interpretable Counterfactual Explanations Guided by Prototypes." arXiv preprint arXiv:1907.02584 (2019)

But the counterfactual is not really interpretable? (It does not look like any number other than 5)

Alibi: adding a prototype for better interpretation

- for better interpretation
- to guide the counterfactual search process



For a class *i* the prototype is given by

$$proto_i = \frac{1}{K} \sum_{k=1}^{K} ENC(x_k^i)$$

If we have access to an encoder (Rumelhart et al., 1986), we follow the approach of (Snell et al., 2017) who define a class prototype as the ***mean encoding of the instances which belong to that class****. In the absence of an encoder, we find prototypes through class specific k-d trees (Bentley,1975)*

Van Looveren, Arnaud, and Janis Klaise. "Interpretable Counterfactual Explanations Guided by Prototypes." arXiv preprint arXiv:1907.02584 (2019)

Alibi: adding a prototype

Autoencoder-based:

For a class *i* the prototype is given by $proto_i = \frac{1}{K} \sum_{k=1}^{K} ENC(x_k^i)$

The prototype loss $L_{proto} = \theta \cdot ||ENC(x_0 + \delta) - proto_j||_2^2$

where *j* is the label of the closest class of x_0 in the latent space

$$j = \arg\min_{i \neq y_0} ||ENC(x_0 + \delta) - proto_i||_2^2$$

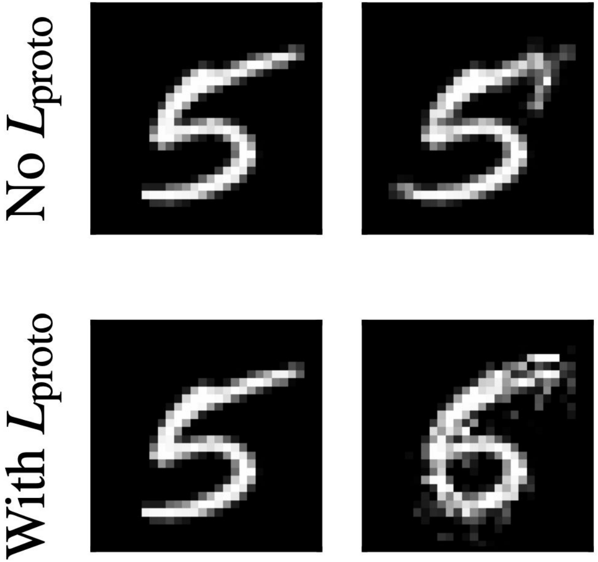
For the k-d tree-based:

$$j = \arg\min_{i \neq y_0} ||x_0 + \delta - x_{i,k}||_2^2$$

where $x_{i,k}$ is the k-th nearest item to x_0 in the k-d tree of class *i*

Alibi: adding a prototype

Example:



Credit: Van Looveren, Arnaud, and Janis Klaise. "Interpretable Counterfactual Explanations Guided by Prototypes." arXiv preprint arXiv:1907.02584 (2019)

Alibi: total loss

$$L = c \cdot \boxed{L_{pred}} + \boxed{L_{dist}} + \boxed{L_{AE}} + \boxed{L_{proto}}$$

- $x = x_0 + \delta$ close to a class different from $y_0 = f(x_0)$
- $x_0 + \delta$ and x_0 are close enough, i.e., $\beta||\delta||_1 + ||\delta||_2^2$ is small
- $x_0 + \delta$ is close to the training data manifold
- $x_0 + \delta$ is close to the prototype of a certain class

Van Looveren, Arnaud, and Janis Klaise. "Interpretable Counterfactual Explanations Guided by Prototypes." arXiv preprint arXiv:1907.02584 (2019)

Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$

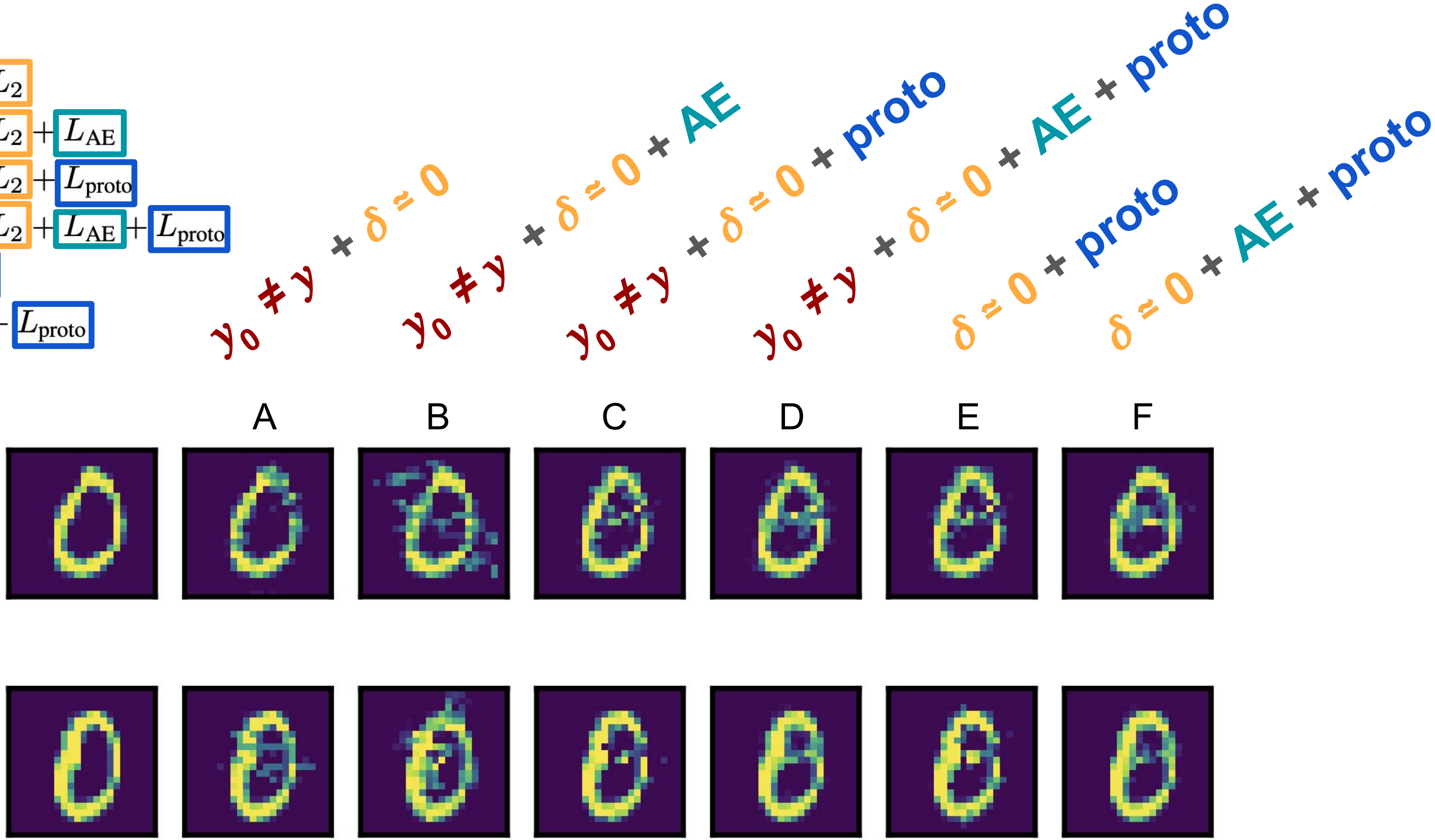
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$

$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$

$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$

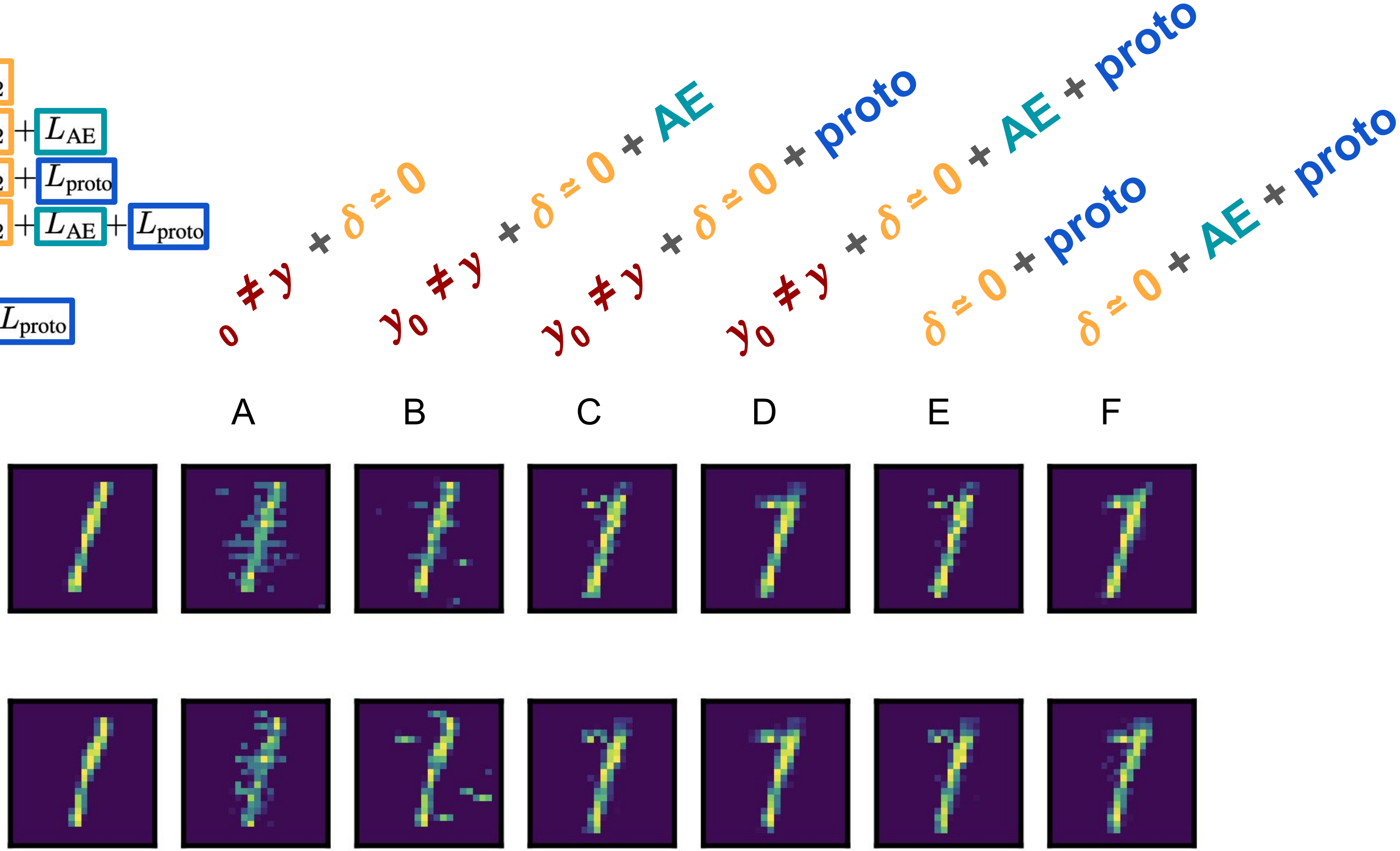
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$

$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$



Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$
$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$



Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$
$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$

$o \neq y + \delta \simeq 0$

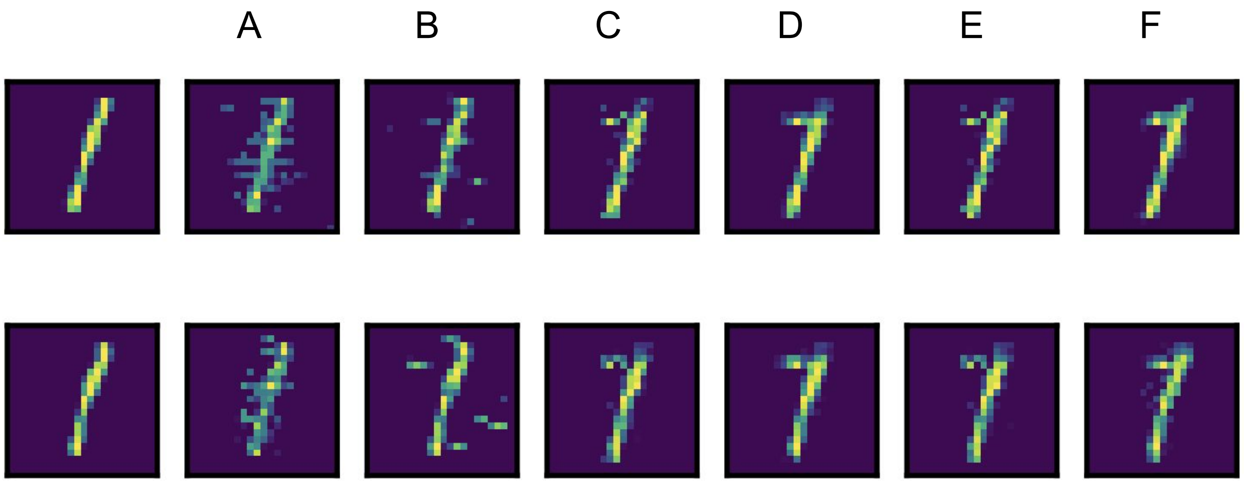
$y_0 \neq y + \delta \simeq 0$ + AE

$y_0 \neq y + \delta \simeq 0$ + proto

$y_0 \neq y + \delta \simeq 0$ + AE + proto

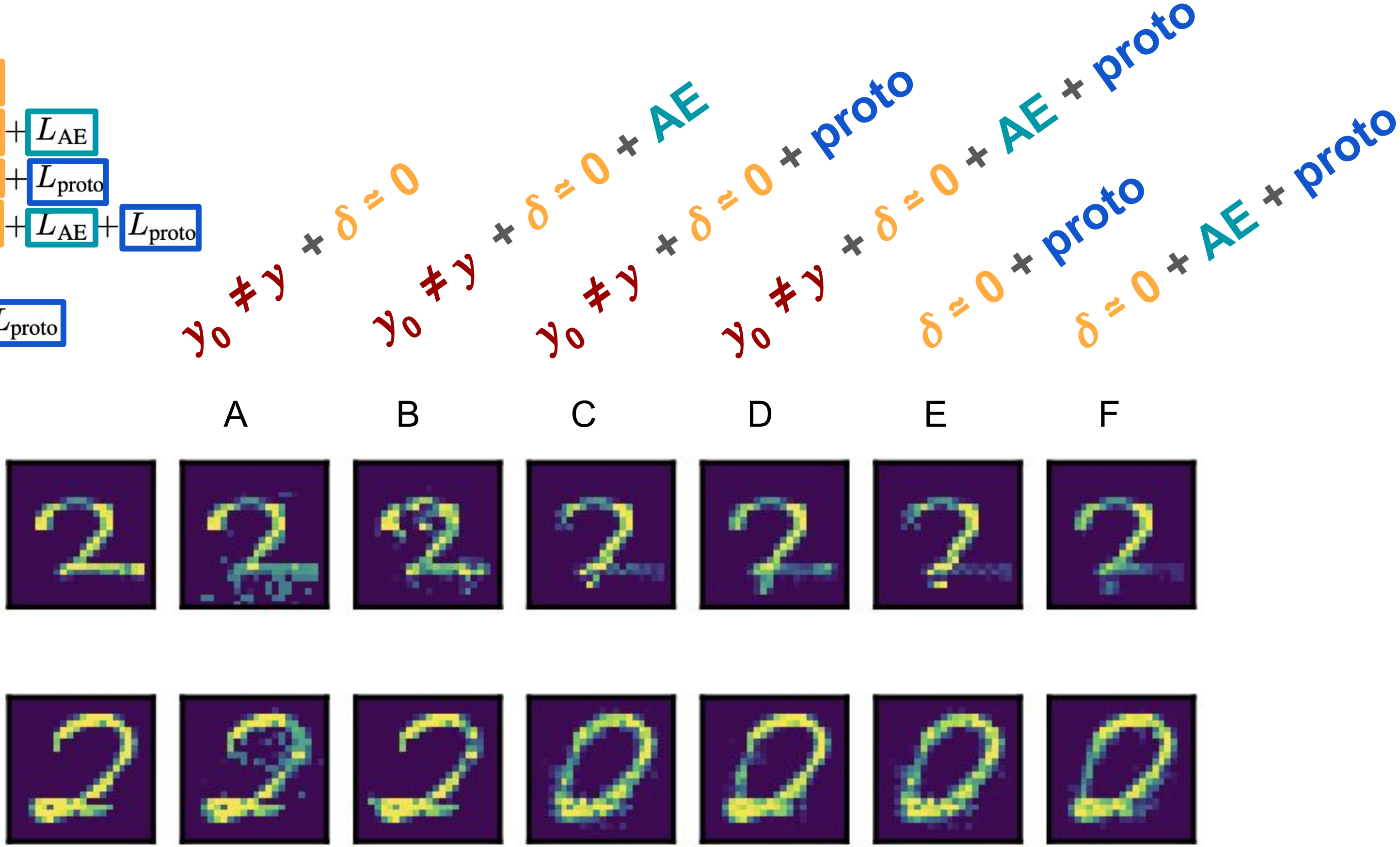
$\delta \simeq 0$ + proto

$\delta \simeq 0$ + AE + proto



Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$
$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$



Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$
$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$

$y_0 \neq y + \delta \simeq 0$

$y_0 \neq y + \delta \simeq 0$ + AE

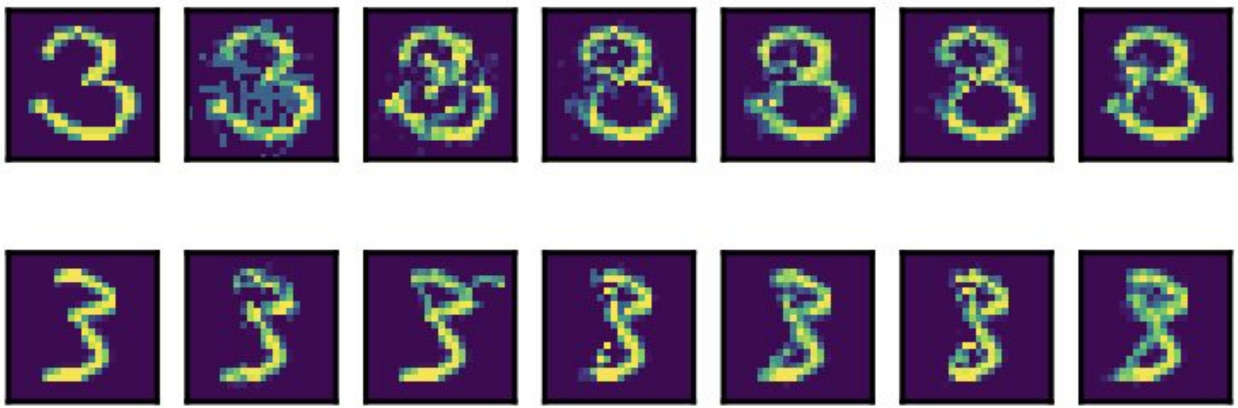
$y_0 \neq y + \delta \simeq 0$ + proto

$y_0 \neq y + \delta \simeq 0$ + AE + proto

$\delta \simeq 0$ + proto

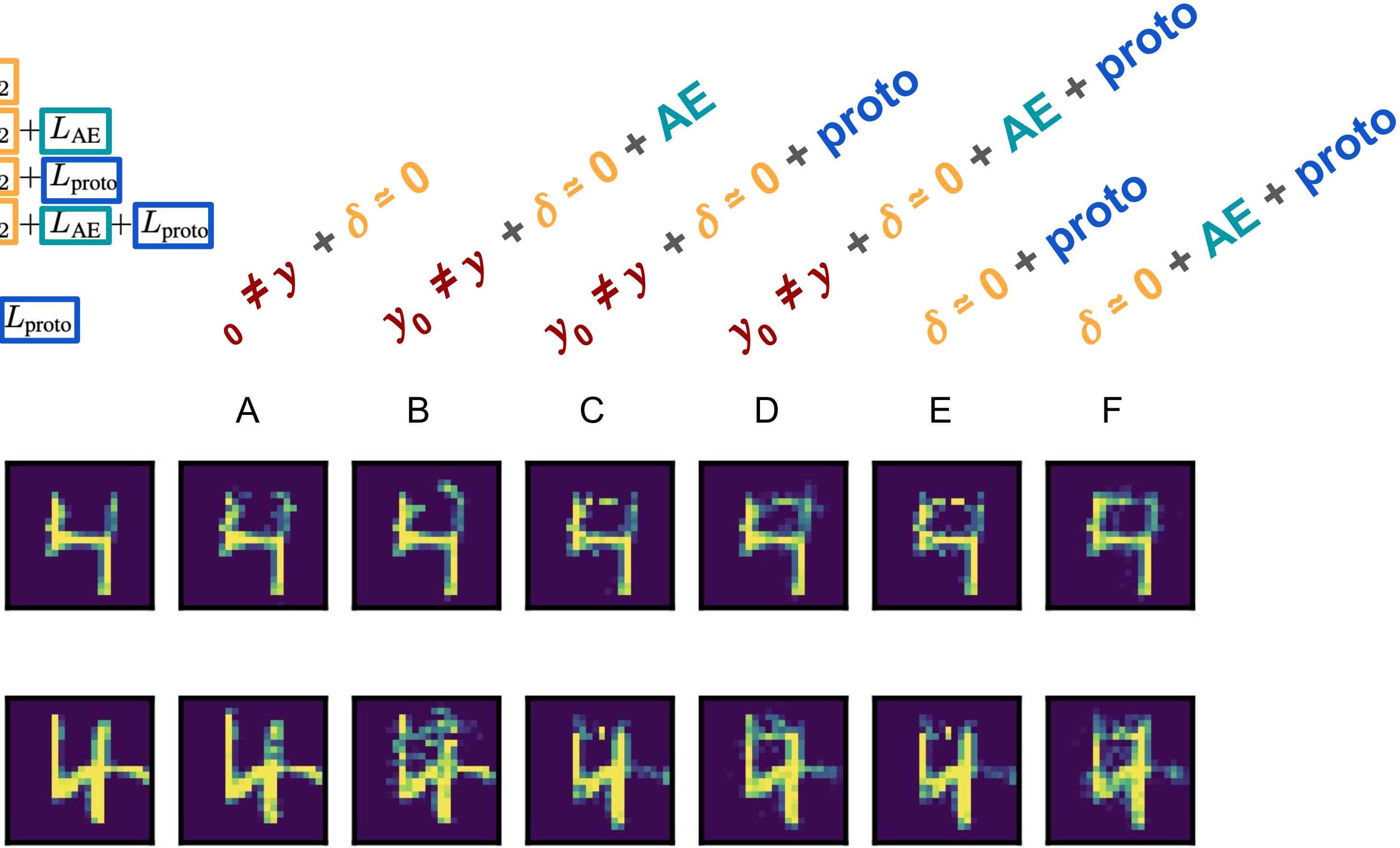
$\delta \simeq 0$ + AE + proto

A B C D E F



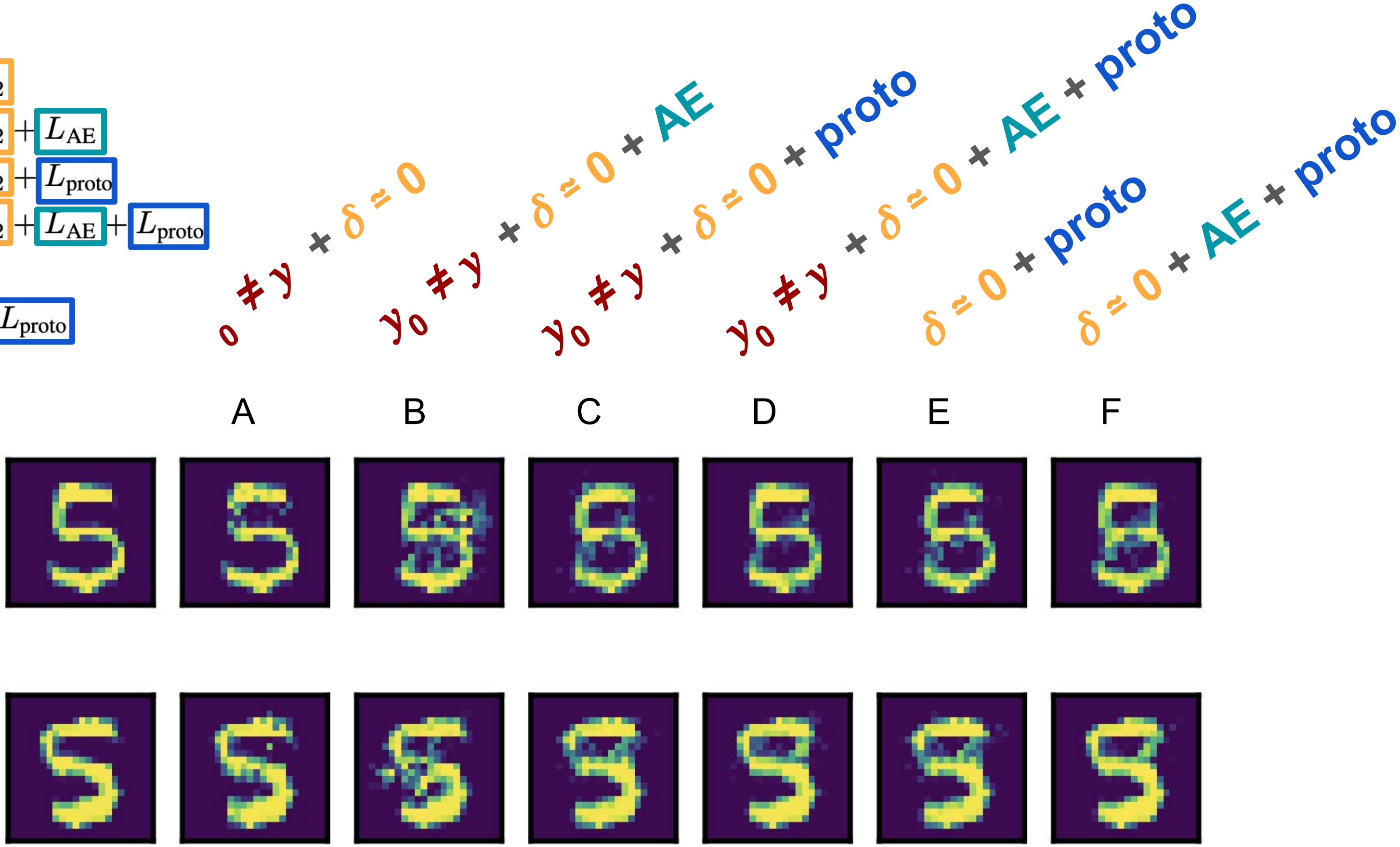
Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$
$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$



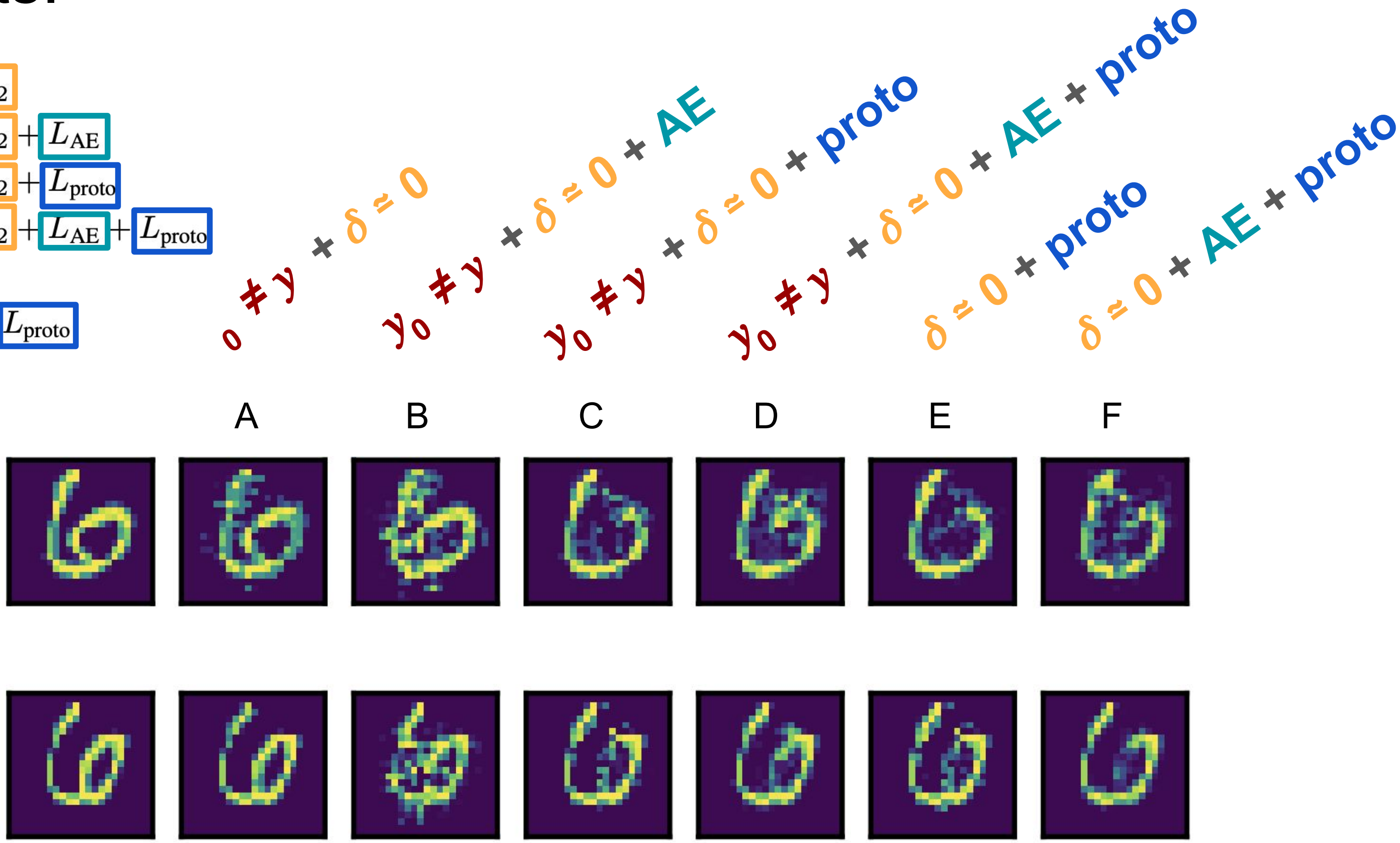
Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$
$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$



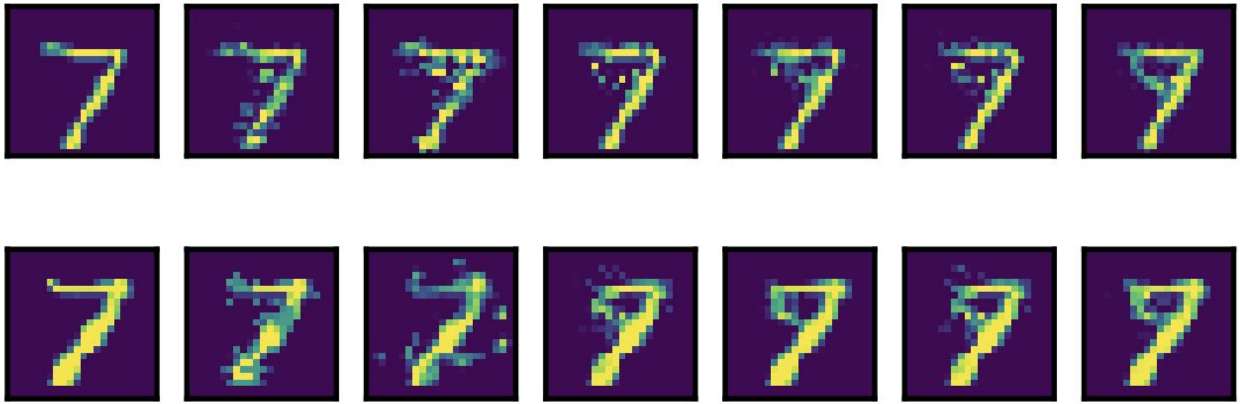
Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$
$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$



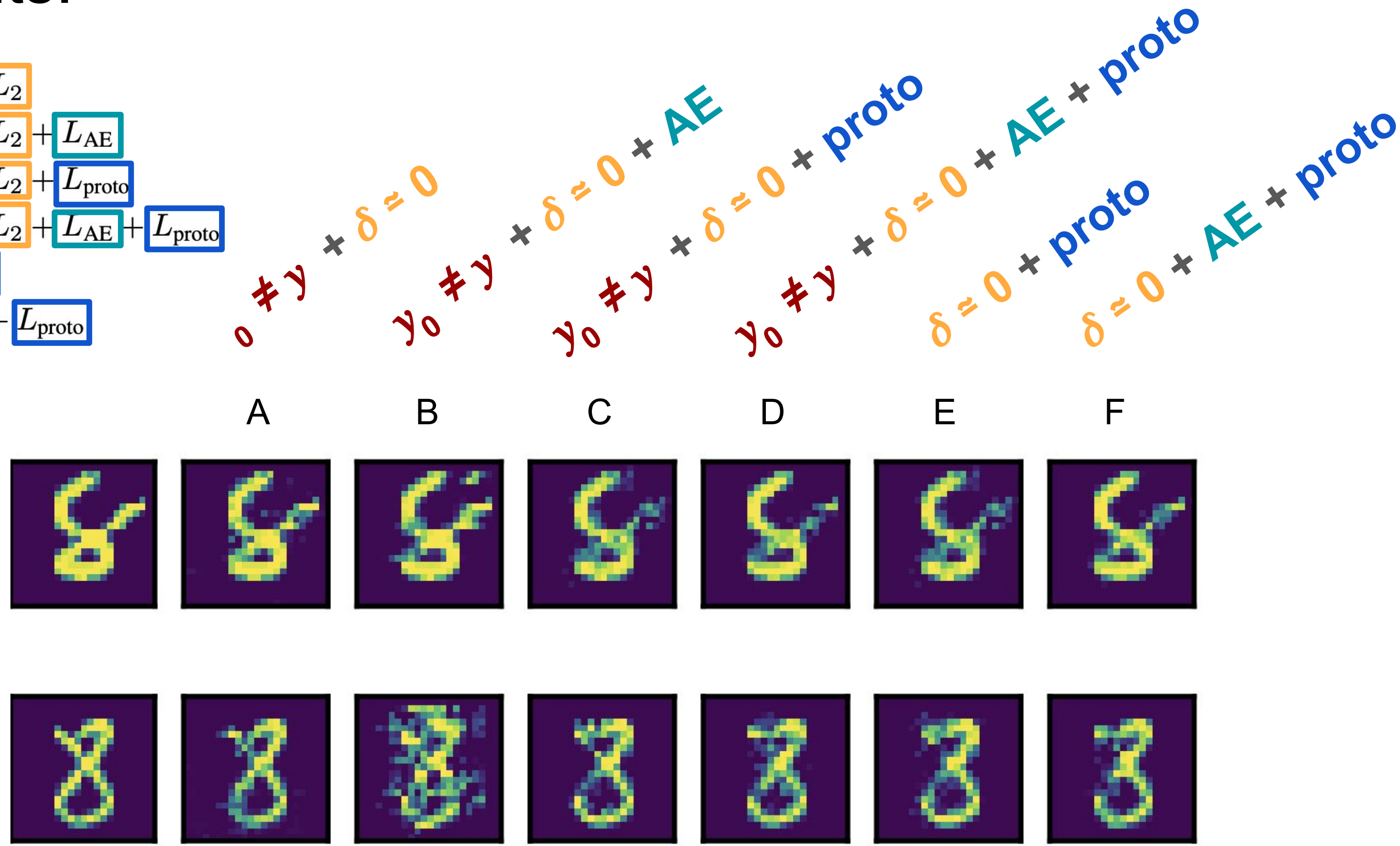
Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$
$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$



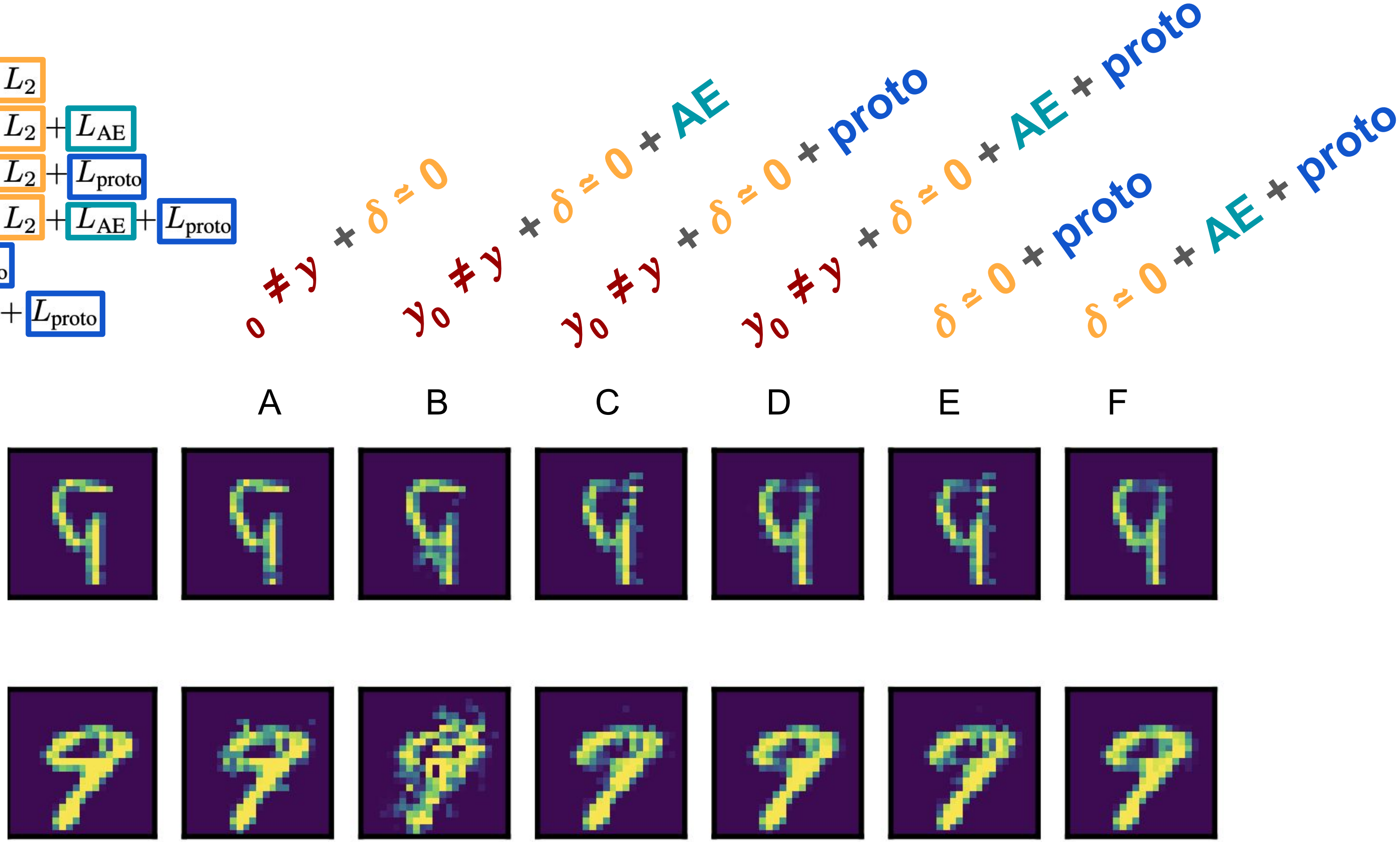
Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$
$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$



Alibi. Results.

$$A = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2}$$
$$B = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}}$$
$$C = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$D = c \cdot \boxed{L_{\text{pred}}} + \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$
$$E = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{proto}}}$$
$$F = \boxed{\beta \cdot L_1 + L_2} + \boxed{L_{\text{AE}}} + \boxed{L_{\text{proto}}}$$



Further reading

Christoph Molnar, “Interpretable Machine Learning. A Guide for Making Black Box Models Explainable”, Chapter 9.3 “Counterfactual explanations”, 2021

Other methods:

- MACE[1] and its Python implementation based on logical formulae and SAT solvers
- DiCE[2] and its Python implementation for differentiable models

1. Karimi, Amir-Hossein, Gilles Barthe, Borja Balle and Isabel Valera. “Model-Agnostic Counterfactual Explanations for Consequential Decisions.” AISTATS (2020)
2. Mothilal, Ramaravind K., Amit Sharma, and Chenhao Tan. “Explaining machine learning classifiers through diverse counterfactual explanations.” Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. 2020

Part 2.2. Deceiving your model. Adversarial examples

Intuition behind

- Find the **vulnerability** of your system by **perturbing** an instance in order your model makes a **false prediction**
- Can be model-agnostic or model-specific (based on gradient)
- Adversarial vs counterfactual:
  - perturbations generated by adversarial attacks are **undetectable** by humans,
  - counterfactual perturbations are **plausible** and **realistic** because the modified samples are contained in the underlying distribution of data that can be encountered in the real world[1]

1. Christoph Molnar, “Interpretable Machine Learning. A Guide for Making Black Box Models Explainable”, 2021

Attacking an ML model

Adversarial examples for QuocNet. A binary car classifier was trained on top of the last layer features without fine-tuning. The randomly chosen examples on the left are recognized correctly as cars, while the images in the middle are not recognized. The rightmost column is the magnified absolute value of the difference between the two images

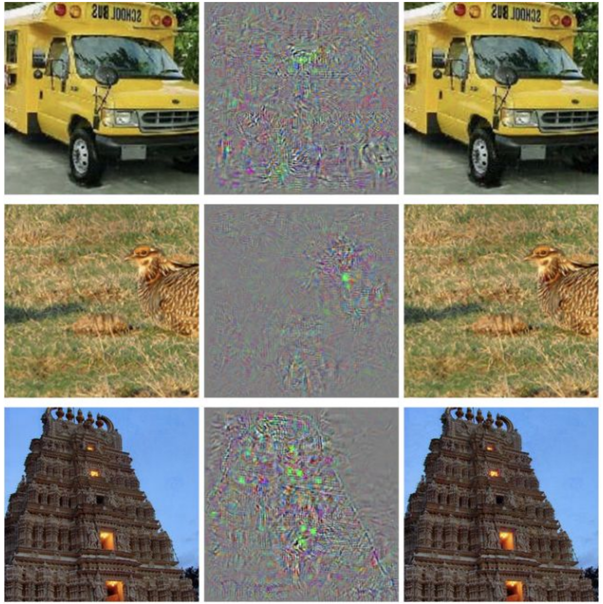
Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013)

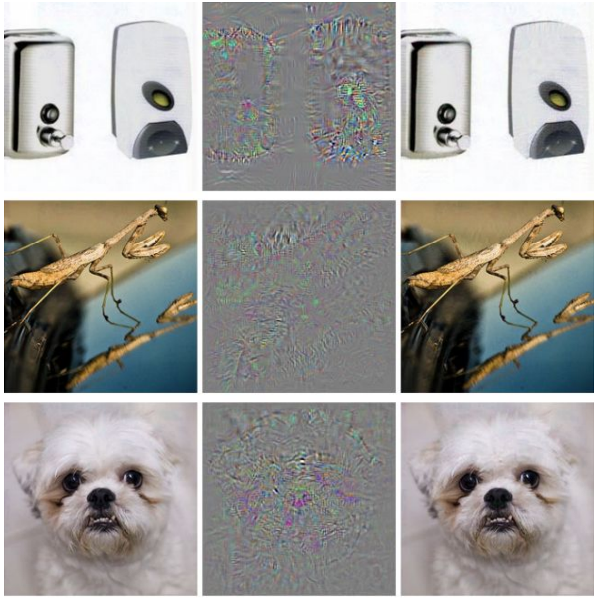
Attacking an ML model

Adversarial examples generated for AlexNet. (Left) is a correctly predicted sample, (center) difference between correct image, and image predicted incorrectly magnified by 10x (values shifted by 128 and clamped), (right) adversarial example. All images in the right column are predicted to be an “ostrich, Struthio camelus”. Average distortion based on 64 examples is 0.006508. for full resolution images. The examples are strictly randomly chosen. There is not any postselection involved. for full resolution images http://goo.gl/huaGPb

Credit: Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013)

Attacking an ML model





Adversarial examples generated for AlexNet. (Left) is a correctly predicted sample, (center) difference between correct image, and image predicted incorrectly magnified by 10x (values shifted by 128 and clamped), (right) adversarial example. All images in the right column are predicted to be an “ostrich, Struthio camelus”. Average distortion based on 64 examples is 0.006508. for full resolution images. The examples are strictly randomly chosen. There is not any postselection involved. for full resolution images http://goo.gl/huaGPb

Credit: Szegedy, Christian, et al. "Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199 (2013)

Important observations for the adversarial attacks

- The same adversarial example is often misclassified by a variety of classifiers with different architectures or trained on different subsets of the training data

- Shallow softmax regression models are also vulnerable to adversarial examples

- Training on adversarial examples can regularize the model—however, this was not practical at the time due to the need for expensive constrained optimization in the inner loop

Szegedy, Christian, Zaremba, Wojciech, Sutskever, Ilya, Bruna, Joan, Erhan, Dumitru, Goodfellow, Ian J., and Fergus, Rob. Intriguing properties of neural networks. ICLR, abs/1312.6199, 2014b. URL http: //arxiv.org/abs/1312.6199

The nature of the problem

The model works well on **naturally occuring data**, but is exposed as a fake when one visits **points in space that do not have high probability** in the data distribution

Let us consider the images, usually stored using 8 bits per pixels (i.e., the information below 1/255 are discarded), ε is the feature **precision**

Let $\boldsymbol{x}' = \boldsymbol{x} + \boldsymbol{\eta}$ be a perturbed instance. If the perturbation is quite small $\|\boldsymbol{\eta}\|_\infty \leq \varepsilon$, that we expect $f(\boldsymbol{x}) = f(\boldsymbol{x}')$

Consider an n-dimensional weight vector $\boldsymbol{w}$ and an adversarial example $\boldsymbol{w}^\top\boldsymbol{x}' = \boldsymbol{w}^\top\boldsymbol{x} + \boldsymbol{w}^\top\boldsymbol{\eta}$

the grow of the activation caused by the adversarial perturbation

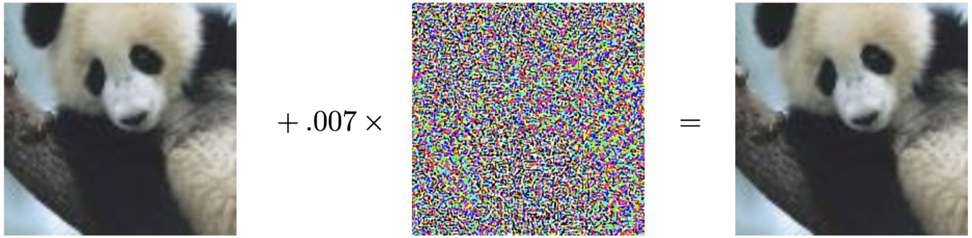
We can maximize this increase subject to the max norm constraint on $\boldsymbol{\eta} = \varepsilon\text{sign}(\boldsymbol{w})$. If avg$|w_i| = m$, thus then the activation will grow by εmn. $||\boldsymbol{\eta}||_\infty$ does not grow with n but the change in activation caused by perturbation by η can grow linearly with n

Hypothesis: neural networks are too linear to resist linear adversarial perturbation

Generation of the adversarial examples

Let $\boldsymbol{\theta}$ be the parameters of a model, $\boldsymbol{x}$ the input to the model, y the targets associated with $\boldsymbol{x}$ and $J(\boldsymbol{\theta}, \boldsymbol{x}, y)$ be the cost used to train the neural network

We can linearize the cost function around the current value of $\boldsymbol{\theta}$, obtaining an optimal max-norm constrained perturbation as follows $\boldsymbol{\eta} = \varepsilon \text{ sign } (\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$



$\boldsymbol{x}$ "panda" 57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$ "nematode" 8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$ "gibbon" 99.3 % confidence

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *arXiv preprint arXiv:1412.6572* (2014)

Main conclusion of Goodfellow et al.

• Adversarial examples can be explained as a property of **high-dimensional dot products**. They are a result of models being **too linear**, rather than too nonlinear.

• The generalization of adversarial examples across different models can be explained as a result of adversarial perturbations being highly aligned with the weight vectors of a model, and different models learning similar functions when trained to perform the same task.

• The **direction of perturbation**, rather than the specific point in space, **matters most.** Space is not full of pockets of adversarial examples that finely tile the reals like the rational numbers.

• Because it is the direction that matters most, adversarial perturbations generalize across different clean examples

• **Adversarial training can result in regularization**; even **further regularization than dropout**

• Models that are **easy to optimize** are **easy to perturb**

• Linear models lack the capacity to resist adversarial perturbation; only structures with a hidden layer (where the universal approximator theorem applies) should be trained to resist adversarial perturbation

• RBF networks are resistant to adversarial examples

• Ensembles are not resistant to adversarial examples

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." *arXiv preprint arXiv:1412.6572* (2014)

Part 2.3 Influential instances

Basics

An **influential instance** is a data instance whose removal has a strong effect on the trained model

If for a model there is an instance having a strong influence on the model predictions, we might not trust to that model.

Approaches for the identification of the influential instances:

- to delete the instance from the training data, retrain the model on the reduced training dataset and observe the difference in the model parameters or predictions
- to upweight a data instance by approximating the parameter changes based on the gradients of the model parameters

The most common measures

To define how strong an instance i influence the model (for deletion-based methods) one uses **DFBETA** and **Cook's distance**.

DFBETA measures the **effect** of deleting an instance **on the model parameters:**

$$DFBETA_i = \beta - \beta^{(-i)}$$

where β is the weight vector when the model is trained on all data instances, and $\beta^{(-i)}$ the weight vector when the model is trained without instance i. It works **only for models with weight parameters**, e.g., logistic regression or neural networks.

Cook's distance[1] measures the **effect** of deleting an instance **on model predictions:**

$$D_i = \frac{\sum_{j=1}^{n} (\hat{y}_j - \hat{y}_j^{(-i)})^2}{p \cdot MSE}$$

p is the number of the parameters

1. Cook, R. Dennis. "Detection of influential observation in linear regression." Technometrics 19.1 (1977): 15-18

Further reading

Christoph Molnar, “Interpretable Machine Learning. A Guide for Making Black Box Models Explainable”, Chapter 10.5 “Influential Instances”, 2021