Explainable Monotonic Networks and Constrained Learning for Interpretable Classification and Weakly Supervised Anomaly Detection

Valentine Wargnier-Dauchelle, Thomas Grenier, Françoise Durand-Dubief, François Cotton, Michaël Sdika

Explainable Monotonic Networks and Constrained Learning for Interpretable Classification and Weakly Supervised Anomaly Detection

Valentine Wargnier-Dauchelle[a,*], Thomas Grenier[a], Françoise Durand-Dubief[a,b], François Cotton[a,c], Michaël Sdika[a]

[a] *INSA Lyon, Universite Claude Bernard Lyon 1, CNRS, Inserm, CREATIS UMR 5220, U1294, F-69100, Lyon, France*
[b] *Service de Neurologie, Hôpital Neurologique, Hospices Civils de Lyon, F-69500, Bron, France*
[c] *Service de Radiologie, Centre Hospitalier Lyon-Sud, Hospices Civils de Lyon, F-69310, Pierre-Bénite, France*

Abstract

Deep networks interpretability is fundamental in critical domains like medicine: using easily explainable networks with decisions based on radiological signs and not on spurious confounders would reassure the clinicians. Confidence is reinforced by the integration of intrinsic properties and characteristics of monotonic networks could be used to design such intrinsically explainable networks. As they are considered as too constrained and difficult to train, they are often very shallow and rarely used for image applications. In this work, we propose a procedure to transform any architecture into a trainable monotonic network, identifying the critical importance of weights initialization, and highlight the interest of such networks for explicability and interpretability. By constraining the features and the gradients of a healthy vs pathological images classifier, we show, using counterfactual examples, that the network decision is more based on the radiological signs of the pathology and outperforms state-of-the-art methods for weakly supervised anomaly detection.

Keywords: Monotonic network, Constrained learning, Weights initialization, Interpretability, Anomaly detection

*Corresponding author: valentine.wargnier.dauchelle@insa-lyon.fr

1. Introduction

Deep learning is widely used in medical image analysis for its effectiveness on various tasks like segmentation, detection, classification or registration. Nevertheless, the decision of a network with good performances may not rely on relevant clinical features [1]. Moreover, due to their highly non-linear nature and their large number of parameters, deep networks decisions are difficult to explain. This lack of interpretability hinders their diffusion in critical areas like medicine. Improving the explicability and the interpretability of the networks is thus a key issue. In terms of explicability, we want the decision to be easy to explain, the explanation to be human readable and guaranteed properties are welcomed. For interpretability, we expect the decision to be based on relevant areas in the image. Attributions maps, such as GradCam [2], can be used to localize the important areas in the image for the decision. Counterfactual examples can also be used to identify areas that need to be changed to modify the network prediction, but they are difficult to interpret. At the same time, several solutions have been proposed to improve interpretability. For example, in [3], a loss is added to smooth attributions maps during the training. In [4], the gradient of the output according to the input of a classifier is constrained to be small in non-interesting areas penalizing its norm in the mask of the annotated region. In [5], the gradient of a healthy vs pathological classifier is also constrained, but in an unsupervised way, such that all voxels of healthy images contribute to the healthy classification. With this method, it is also possible to segment the anomalies in a weakly supervised way. State-of-the-art anomaly detection methods often aim at learning the healthy distribution and at segmenting anomalies on test pathological images using the weak reconstruction of the unseen patterns. Auto-encoders (AE) [6] or variational auto-encoders (VAE) [7] are the most used for this type of task. Some regularization of attributions can be added to their training to improve the segmentation performances as in [8]. Generative adversarial networks can also be used [9] to improve the reconstruction. However, anomaly detection can only be used for segmentation, not classification, and state-of-the-art methods based on classification do not rely on intrinsic explicability properties and the performances are based on a complete generalization during inference. Monotonic networks have interesting intrinsic properties: input variable behavior is straightforward due to the monotonic link between the input features and the decision.

Several techniques are used in the literature to build monotonic networks.

The simplest way is to force the layers' weights to be non-negative and activation functions to be monotonically increasing as in [10, 11]. Some authors, as in [12] and [13], argue that these constraints are too restrictive. Thus, [12] introduces three specific layers to build a monotonic network: linear embeddings, calibrators and ensembles of lattices, thus imposing a specific architecture. [13] constrains the monotonicity to each block of two layers using Mixed-Integer Linear Programming (MILP). On the other hand, [14] states that convex monotonic networks that-is-to-say networks with non-negative weights and convex activation functions are not too restrictive when the network has multiple outputs. To do so, the decision function becomes a difference of convex functions to allow non-convex decision boundaries on the last layer. They reach state-of-the-art performances for classification on several toy datasets (CIFAR, MNIST, etc). In [15], the convexity issue is solved by using convex activation functions on part of the layer output and concave activation functions on the remaining part.

We claim that the issues of non-negative weights networks might be due to optimization difficulties instead of a lack of capacity. Indeed, these monotonic networks are often harder to train and small architectures with fewer capacities are often used to achieve convergence. We also claim that this convergence problem is linked to the non-variance preservation of the network's internal layers with classic weights initializations. Most popular methods for deep network weights initialization (Xavier [16], Kaiming [17]) use well-chosen random weights such that feature variance is preserved across layer: features and gradients do not explode/vanish at initialization. However, specific layers such as, maxpool, residual and non-negative linear layers are not accounted for in these procedures. As modern neural networks can be very deep to increase model capabilities and performances, the non-preservation of the variance is accumulated along the network and it could be huge for the last layers of the model. To initialize the weights without any assumptions, [18] proposes an iterative procedure (LSUV) to rescale weights layer by layer. Nevertheless, this initialization is very long due to multiple iterations.

In this work, we propose 1/ an architecture, named MoE, for healthy vs pathological classification which enjoys several intrinsic explicability properties using monotonic networks: an ordered and bounded latent space and an improved readability of counterfactual examples. To do so, we propose 2/ a recipe to transform every architecture into a trainable monotonic network with non-negative weights. 3/ We theoretically identify why state-of-the-art initialization methods are not suited for non-negative networks and

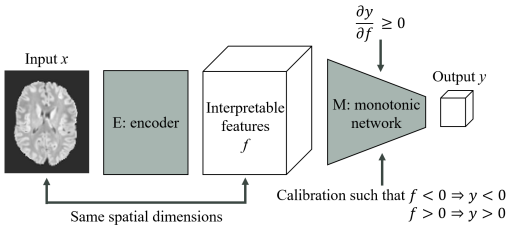


Figure 1: MoE architecture. An encoder E computes the features f with the same spatial dimension of its input x. The logit y of the healthy vs pathological classification are then computed with a monotonic and calibrated classifier M.

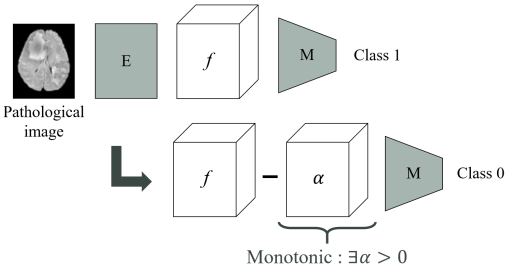


Figure 2: Counterfactual example generation. From a pathological image originally classified as pathological (class 1) with interpretable features f, we search the counterfactual difference α as $f - \alpha$ is classified as healthy (class 0).

propose a new initialization scheme that enables their training. 4/ We formally characterize two-layer monotonic networks and show that our simple parametrization is not over-restrictive. 5/ We also propose to constrain, during the training and in an unsupervised way, the features and the gradients to improve the interpretability of our classifier. With this constrained and explainable network, we obtain a more interpretable classification with a decision more based on the radiological signs, and so we are able to segment the pathological areas in a weakly-supervised way using counterfactual examples.

2. Methodology

In this work, we propose to use monotonic networks to improve the interpretability of a classifier. To do so, we propose, in Section 2.1, a new architecture with a non-negative network to add intrinsic explicability properties. In Section 2.2, we detail how the non-negative network is built. Then, in Section 2.3, we theoretically characterize the 2-layers monotonic networks to justify our choice to use non-negative networks as monotonic networks. Section 2.4 proposes a new weights initialization method, necessary to train the proposed networks. Finally, in Section 2.5, we propose to constrain our network during training to improve the interpretability in the case of a healthy vs pathological images classification.

2.1. Using monotonic network for intrinsic explicability

The architecture of our network, denoted as MoE, is presented in Fig. 1. It consists of an encoder E that produces the interpretable features f, followed by a monotonic network M that computes the logits y of the binary classification. These networks are detailed in the two following subsections.

2.1.1. Calibrated Monotonic Network

A network $M : \mathbb{R}^n \to \mathbb{R}^m$ is monotonically increasing if:

$$\forall x, \forall i, \frac{\partial M(x)}{\partial x_i} \geq 0 \tag{1}$$

where x is the input and i is the index in the input's components. For simplicity, monotonic networks will always refer to monotonically increasing networks. With monotonicity, the network output increases (and decreases) with the input making the prediction more explainable with respect to the input behaviour.

We propose to build monotonic networks with another guaranteed property: if all components of the features f are non-negative (resp. non-positive), then the subject is classified as positive (resp. negative). Formally, this calibration property can be written as:

$$(\forall i,\ f_i \geq 0) \Rightarrow y \geq 0 \qquad \text{and} \qquad (\forall i,\ f_i \leq 0) \Rightarrow y \leq 0. \tag{2}$$

Details on how to easily enforce this property are given in Section 2.2.1.

For a healthy vs pathological classifier, as the latent space f is ordered in same direction as the network output, high-value features are directly related to a high probability of being pathological, and the proposed calibration creates a known boundary between the negative/healthy and positive/pathological classes.

2.1.2. Encoding Interpretable Features

The encoder E aims at extracting features f from the input. As f will be the input of a monotonic network, it will enjoy explicability properties of Section 2.1.1. The raw input cannot directly be used as the input of the monotonic network, unless a monotonic relation between the voxels values and the desired output exists. The non-monotonic encoder E rearranges the features of the input image such that the network output is indeed monotonic with respect to these features. We also enforce other properties in E to

improve explicability. E is composed of few convolutional layers, keeping the same size as the input which would also make these features locatable, avoiding the interpolation of GradCam-like methods. As we will see in Section 2.5.1, it may also be useful to force biases to be zero in E.

2.1.3. Explicability with Counterfactual Examples

In addition to the previous explicability properties, our proposition also improves the readability of counterfactual examples. As presented in Fig. 2, we aim to find the minimum changes α of the features f of a "pathological" subject for it to be classified as "healthy". Formally, α is estimated as in [19]:

$$\min_{\alpha} M(f-\alpha) + \lambda \|\alpha\|_1 \tag{3}$$

where λ is a regularization weight. We consider that convergence is reached when $M(f-\alpha) < m$ where $m \leq 0$ is a margin from the decision boundary. Now, as M is a non-negative network, a non-negative α solution to this problem exists . As we will see in the experiments, this makes the generated counterfactual example much easier to read and understand. These examples can be used as an explanation of the network indicating which areas are useful to make the pathological decision: thresholding these maps provides a weakly-supervised segmentation of the pathology.

2.2. Transformation into non-negative weights network procedure

To build the monotonic network M, we start from any network and transform it into a non-negative network using the following rules.

2.2.1. Linear Layers (FC/Conv)

In the monotonic network M, the weights of the fully connected (FC) and convolution layers (Conv) are parameterized with a non-negative function. We choose the function $\varphi : W \mapsto W^2/\sqrt{1+W^2}$ which is differentiable and enables to have null weights. To enforce the calibration property (Eq. 2), it is sufficient to keep all the biases set to zeros and use an increasing activation function σ that vanishes in zero. Indeed, in this case, for a given input f, the output of an FC/Conv layer with non-negative weights w satisfies:

$$\forall i,\ f_i \geq 0 \text{ (resp. } \leq 0) \ \Rightarrow \sigma\left(\sum_i w_i f_i\right) \geq \sigma(0) = 0 \text{ (resp. } \leq 0)$$

and the calibration property is enforced if these constraints are enforced for all FC/Conv and activation layers.

2.2.2. Activation Functions

Activation functions are set to any standard increasing function $r(x)$ (like ReLU) on the first half of the layer channels and to $-r(-x)$ on the remaining channels. As in [15], this means that the network does not have to be convex.

2.2.3. Normalisation Layers

Normalisation layers are defined as $y = \frac{x-\mathbb{E}(x)}{\sigma(x)}$ where the mean $\mathbb{E}$ and standard deviation σ are computed along some given dimensions of the input tensor depending on the normalisation type (batch normalisation [20], instance normalisation [21], etc). As these layers would break the monotonicity property, they are removed from the monotonic part of the network.

2.3. Two-layer monotonic networks characterization

The proposed conversion procedure guarantees the monotonicity of the network by forcing each layer to be increasing. One can wonder if this imposes too much restriction. With the proposition below, we characterize monotonic two-layer networks to force monotonicity only two layers by two layers.

Proposition 2.1. *Let g be the two-layer network defined by $g(x) = A\sigma(Bx)$, with σ the ReLU function. If we note D_v the diagonal matrix with the vector v on its diagonal, then:*
Assertion 1: If $A = \tilde{A}D_a$, $B = D_b\tilde{B}$ with $(a, b) \in \mathbb{R}^n \times \mathbb{R}^n$, $\text{sign}(a) = \text{sign}(b)$ and $\tilde{A}$ and $\tilde{B}$ non-negative then g is monotonic.
Assertion 2: If B is surjective and g is monotonic then $\exists (a, b) \in \mathbb{R}^n \times \mathbb{R}^n$ such that $A = \tilde{A}D_a$ and $B = D_b\tilde{B}$ with $\tilde{A}$ and $\tilde{B}$ non-negative and $\text{sign}(a) = \text{sign}(b)$.

Proof. Note first that if for a given x, $w(x)$ is defined as $w(x) = \sigma'(Bx)$, then $w(x) \geq 0$ and $g'(x) = AD_{w(x)}B$.

Let's prove the first assertion. If $A = \tilde{A}D_a$ and $B = D_b\tilde{B}$ then $g'(x) = \tilde{A}D_aD_{w(x)}D_b\tilde{B} = \tilde{A}D_{aw(x)b}\tilde{B}$. Consequently, g is monotonic as:

$$\frac{\partial g_i}{\partial x_j}(x) = \sum_k \tilde{a}_{i,k}a_k w_k(x) b_k \tilde{b}_{k,j} \geq 0.$$

Let's now prove the second assertion. As B is surjective, for all l, there exists x^l such that Bx^l is 1 for the l^{th} component and -1 for all other components so the l^{th} component of $w(x_l)$ is one and all other components are null. As g is monotonic:

$$\forall l, \forall i, \forall j, \frac{\partial g_i}{\partial x_j}(x^l) = \sum_k a_{i,k} w_k(x^l) b_{k,j} = a_{i,l} b_{l,j} \geq 0 \tag{4}$$

Let's b be the first column vector of B. Then for all i and l, $a_{i,l} a_l \geq 0$. This implies that all the components of the l^{th} column vector of the matrix A have the same sign as a_l: A can be factored as $A = \tilde{A} D_a$ with $\tilde{A}$ non negative. Using this factorization, Inequality 4 becomes:

$$\tilde{a}_{i,l} a_l b_{l,j} \geq 0$$

Consequently, one can similarly factor B as $B = D_a \tilde{B}$. □

When $a = b$ is a vector with value 1 on half of its components and -1 on the other half, $D_a \sigma(D_b \cdot)$ is the activation functions $ReLU(x)$ and $-ReLu(-x)$ of Section 2.2.2. Thus, with B surjective, the proposed monotonic network construction almost covers all the possible two-layer monotonic networks. For more flexibility, the only solution is to relax the number and the position of the positive/negative components. This supports our assumption that non-negative weights networks are not a too constrained solution to build monotonic networks. In addition, in [13], two-layer by two-layer monotonic networks were obtained by solving a computational demanding MILP at each pass through the network. whereas our parameterisation guarantees that the network is monotonic at no cost.

2.4. Weights initialization

In this section, we show that state-of-the-art random weights initializations are unsuitable for some layers, especially for non-negative weights layers. We also propose a new initialization method to solve this issue.

2.4.1. A need for a new initialization method

In [16, 17], it is shown that if both weights and inputs of a linear layer are independent, identically distributed (iid), centered and mutually independent then the weights variance can be chosen to preserve the feature variance. Although these random initializations are very effective in stabilizing the

training, we found that residual blocks, maxpool, and FC/Conv layers with non-negative weights are not accounted by these methods. Note that these limitations are usually handled transparently with the use of normalisation layers but these layers cannot be used for monotonic networks.

Residual blocks. They encode $y = x + F(x)$ where F is typically few convolutions. The variance of the output is given by $\mathbb{V}(Y) = \mathbb{V}(X) + \mathbb{V}(F(X)) + 2\text{Cov}(X, F(X))$ which becomes $V(Y) = 2 + 2\text{Cov}(X, F(X))$ if X is standardized and F is variance preserving at initialization. The variance is likely to at least double at each residual block.

MaxPooling layers. They encode $y_i = \max\left(x_{is+k}\right)_{k \in N}$ with s the stride and n the size of the neighborhood N. For example, when X follows a uniform distribution, Y will follow a $Beta(n, 1)$ distribution [22] whose mean is $\frac{n}{n+1}$ and variance is $\frac{n}{(n+1)^2(n+2)}$: features will not be centered and their variance will be reduced.

FC/Conv layers with non-negative coefficients. In the proposition below, we give some insights of the consequences of non-negative weights on the features correlation. Among other things, it explains why state-of-the-art initialization methods cannot work for networks with non-negative weights.

Proposition 2.2. *Let x be the input of a linear layer (FC/Conv), $\mu = \mathbb{E}(x)$, $C = \text{Cov}(x)$ and ρ defined as $\rho = \frac{\sum_i C_{i,i} + \mu_i^2}{\sum_{i,j} C_{i,j}}$. Let also $y = \sum_{i=1}^n a_i x_i + t_a$ and $z = \sum_{i=1}^n b_i x_i + t_b$ be two components of the output of this linear layer where n is the number of non-null coefficients, a_i and b_i are the weights and t_a and t_b are the biases. If*

1. *$\forall i$, a_i (resp. b_i) are iid with expectation μ_w and variance σ_w^2,*
2. *$\forall i$, $\forall j$,$\forall k$: x_i, a_j and b_k are independent.*

then the correlation between y and z is given by:

$$\text{corr}(x, y) = \frac{\mu_w^2}{\mu_w^2 + \sigma_w^2 \rho}. \tag{5}$$

Proof. As $\mathbb{E}(y) = \mu_w \sum_i \mu_i + t_a$ (and similarly for $\mathbb{E}(z)$), the covariance

between y and z is:

$$\begin{aligned}
\mathrm{Cov}(y,z) &= \mathbb{E}\left((y-\mathbb{E}(y))(z-\mathbb{E}(z))\right) \\
&= \sum_{i,j} \mathbb{E}\left((a_i x_i - \mu_w\mu_i)((b_j x_j - \mu_w\mu_j)\right) \\
&= \sum_{i,j} \mu_w^2 \mathbb{E}(x_i x_j) - \mu_w^2 \mu_i \mu_j \\
&= \mu_w^2 \sum_{i,j} C_{i,j}
\end{aligned}$$

The variance of y (and similarly for z) is given by:

$$\begin{aligned}
\mathbb{V}(y) &= \mathbb{E}(y-\mathbb{E}(y))^2 \\
&= \sum_{i,j} \mathbb{E}\left((a_i x_i - \mu_w\mu_i)((a_j x_j - \mu_w\mu_j)\right) \\
&= \sum_{i,j} \mathbb{E}(a_i a_j) E(x_i x_j) - \mu_w^2 \mu_i \mu_j \\
&= \sum_{i,j} \mathbb{E}(a_i a_j)(C_{i,j} + \mu_i\mu_j) - \mu_w^2 \mu_i \mu_j \\
&= \sigma_w^2 \sum_i (C_{i,i} + \mu_i^2) + \mu_w^2 \sum_{i,j} C_{i,j}.
\end{aligned}$$

By calculating the ratio of $\mathrm{Cov}(y,z)$ over $\sqrt{\mathbb{V}(y)\mathbb{V}(z)}$, we obtain Eq. 5. □

Eq. 5 implies that it is essential, as in [16, 17], that $\mu_w = 0$ to avoid correlated features. For a network with non-negative coefficients, this is not possible. In Fig. 3, we plot the correlation of the features as a function of $\frac{\sigma_w^2}{\mu_w^2}$ for a two convolutional layers network with null biases and non-negative weights for several values of the number of channels n after the first convolution and an input with identity covariance. This figure shows that the features are highly correlated in the second layer even for a small ratio $\frac{\sigma_w^2}{\mu_w^2}$. This invalidates the independence requirement for the use of [16, 17] initializations and explains why these methods cannot be used in this context.

2.4.2. Weights initialization procedure

As in [18], we propose a programmatic way to initialize the weights of a deep network. This initialization aims to preserve the features' variance

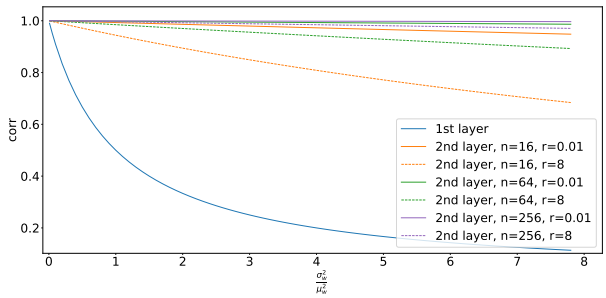


Figure 3: Features correlation for the two first layers as a function of $\frac{\sigma_w^2}{\mu_w^2}$ and for several weights support size n of the considered layer. For the second layers, two values of $r = \frac{\sigma_{w_1}^2}{\mu_{w_1}^2}$ have been drawn.

along the whole network as in [16, 17] but without hypotheses on the input distribution of each layer. To do so, the weights are first initialized using a standard random initialization and we draw a random input noise mini-batch, x_{train}, using a Gaussian distribution $\mathcal{N}(0, 1)$ (but any distribution can be chosen). Then, we follow the computational graph of the network, and rescale, in order, the weights of each linear layer by the standard deviation of the layer output to obtain a unit variance output. The rescaled output is then used as the input of the next layer.

Our method has the benefit of taking into account the global architecture of the network without assuming that the input of the linear layers is standardized: if a residual block or a maxpooling layer precedes a linear layer, the weights will naturally be adjusted. It does not assume independence of the input feature and can work for a network with non-negative weights. It also compensates for the removal of normalisation layers which explains, in part, why non-negative networks are difficult to train. As a result, the proposed weights initialization plays a crucial role in the ability to train non-negative networks.

From an implementation point of view, this initialization can be done in a single forward pass through the network. Each linear layer $y = Wx$ is replaced with the wrapper of Algorithm 1 for the initialization only. This wrapper takes into account non-negative networks for which weights are parameterized by a non-negative function ($W = \varphi(\tilde{W})$). φ is the identity function for non-parameterized layers. Note that φ^{-1} is just the right inverse of φ, as required by the weights parameterization module of Pytorch. Then, a forward pass of the model is run with the random noise input x_{train}. Fi-

Algorithm 1 Initialization wrapper

function LinearWrapper(x)
  $y = \varphi(\tilde{W})x$
  $\sigma = \sqrt{V(y)}$
  $\tilde{W} = \varphi^{-1}(\varphi(\tilde{W})/\sigma)$
  return $\varphi(\tilde{W})x$
end function

nally, the (rescaled) linear layers are returned to their original place in the network. In practice, our initialization function can be run on any model, independently from the architecture and the input features distribution.

Even if our initialization shares some similarities with LSUV [18], there are several significant differences. First, our formulation is also valid for parameterized layers. Then, each time a layer is normalised, a full forward on the whole network is done in LSUV whereas only this layer is computed with our implementation: the LSUV implementation has a quadratic complexity while ours is linear. Thus, for a 2D ResNet152 [23] and a batch of size 50, the initialization takes around 12s with our proposition against 22min for [18]. Finally, they use a data minibatch whereas we use random noise to avoid over-fitting on the selected data.

2.5. Constrained non-negative networks for interpretable classification and anomaly detection

We consider a binary classification between healthy images (negative class) and patient images (positive class). In this case, an interpretable network should base its decision on the radiological signs of pathology. In addition to the intrinsic explicability properties of the proposed non-negative network, we propose to use several constraints, based on these properties, to make our network more interpretable. The network MoE is thus trained with the loss function :

$$L = L_C + L_{nF} + L_{dF} + \alpha_G L_G \tag{6}$$

where L_C is the classification loss function (here a binary cross-entropy), L_{nF} and L_{dF} constrain the features f and finally L_G is a regularization of the gradients of M weighted with the coefficient α_G. These constraints do not require any additional annotation: only the image label, already necessary for the classification task, is required.

2.5.1. Negative features constraint for healthy subjects

When the network is calibrated, its output is necessarily negative for a completely negative input. Therefore, we propose to constrain the feature maps given by the encoder E as input of the monotonic part M to be negative for healthy images. To do this, we define the loss function L_{nF} as:

$$L_{nF}(x_0) = \|ReLU(E(x_0))\|_1 \tag{7}$$

and use it only on healthy subjects' images x_0. For this constraint, it is important to also force null biases in the encoder E in order to avoid only learning very negative biases in E.

2.5.2. Matching features distribution of the two classes

As we want the classification to be based on the presence of the pathology, areas outside the pathology, which are visually healthy, should not be discriminative. With the previous constraint, features in healthy tissue will be negative. Thus, we want the distributions of negative features to be similar for both classes. To achieve this, we use the Kullback-Leibler loss to match the distributions of negative features:

$$L_{dF}(x_0, x_1) = KL(P_0, P_1) \tag{8}$$

where x_0 and x_1 are respectively a healthy and a pathological minibatch, P_i corresponds to the distributions of the negative part of the interpretable features $E(x_i)$ and KL is the Kullback-Leibler divergence. This constraint prevents the network from classifying the data using differences in the "healthy part" of the images. It can be seen as a form of domain adaptation included in the learning process.

2.5.3. Gradient regularisation

In the case of monotonic networks, the gradient of the output relative to the input of the monotonic network is non-negative. An adaptation of [5] would therefore be to constrain the gradients of the monotonic part M to be small for healthy images rather than negative. This constraint should be seen as a regularization in the same way as in [24]. The weighting coefficient α_G must be relatively small compared with the other loss functions. A L1

norm is used for this regularization:

$$L_G(x_0) = \left\| \frac{\partial M}{\partial f}(E(x_0)) \right\|_1 . \tag{9}$$

Note that as M is monotonic, the L1 norm is just a sum.

3. Experiments

Our method was evaluated in two stages. First, in Section 4.1, we verified that the proposed initialization allows us to obtain similar or better training than Kaiming's method as it is still the most widely used random initialization method. We also evaluate whether the proposed architecture using a non-negative network, reaches similar or better classification performances on various tasks and various network architectures for the monotonic network. Next, in Section 4.2, we evaluated the interpretability of our method and its segmentation performances on a healthy vs tumor images classification task, by comparing our method with: the AE [6], the VAE [7], f-AnoGAN [9], Silva-Rodríguez [8], GradCam [2], Erion [3], Ross [4] as used in [5] and Wargnier-Dauchelle [5]. We also used two baseline models: a classification network with the same architecture as our proposition but non-monotonic and unconstrained (Baseline) and a non-monotonic but constrained one (BaselineC) with the loss functions L_{dF} and L_{nF} of the Section 2.5. The loss function L_G was not added as the convergence is unstable in this case. For the two baselines and our proposition, the segmentation is based on the difference α in Eq. 3 to generate a counterfactual example on the interpretable features.

3.1. Data

The following public medical datasets mixing different modalities, classification tasks and data dimensions were used and split according to Table 1. MedNIST: The toy medical dataset MedNIST[1] is composed of 64×64 2D images divided into 6 classes: abdomen CT, breast MRI, chest CT, chest

[1]The MedNIST dataset was gathered from several sets from TCIA, the RSNA Bone Age Challenge, and the NIH Chest X-ray dataset. The dataset is kindly made available by Dr. Bradley J. Erickson M.D., Ph.D. (Department of Radiology, Mayo Clinic) under the Creative Commons CC BY-SA 4.0 license.

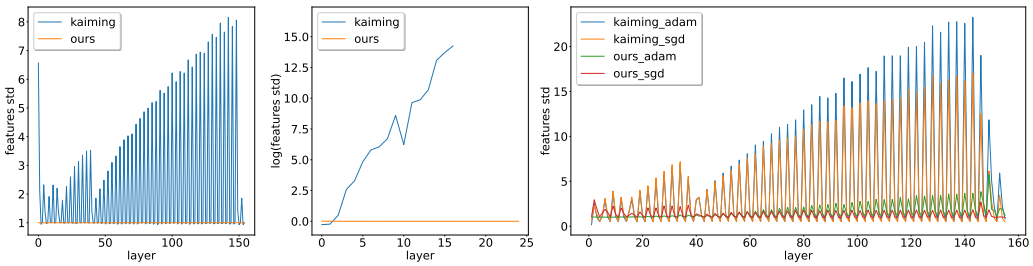
X-ray, hand X-ray and brain CT. Brain Tumors: The Brain Tumors MRI dataset [25] is composed of 512 × 512 2D images of T1 (with and without contrast enhancement), T2 and FLAIR MRI (axial, coronal and sagital views). There are 4 classes: no tumor, glioma, meningioma and pituitary tumor. Single-Cell: The public single-cell proteomics dataset [26] proposed 13 values of surface markers for 20 different healthy cellule types. Tumor vs healthy classification: For this task, three healthy subjects FLAIR MRI datasets were used: MPI [27], kirby21 [28] and IBC [29]. MICCAI BraTS 2020 []has been used for brain tumor images. These 3D images were acquired in different centers with multi-brand, 1.5T and 3T scanners. They were preprocessed using FSL FLIRT [30] affine registration on MNI atlas MRI and HD-BET brain extraction [31].

Table 1: Train/validation/test split for each dataset.

Datasets	N_{train}	N_{val}	N_{test}
MedNIST	47163	5895	5995
Healthy: MPI, kirby21, IBC	64, 22, 8	15, 5, 2	15, 5, 2
Tumors: BraTS2020	280	40	49
Brain Tumors	2870	99	295
Single-Cell	58374	6486	16215

3.2. Implementation details

The networks were implemented using Pytorch. Several classification architectures were used: a 3D 70x70 PatchGan [32], 2D ResNets [23] (from 18 to 152 layers) that have been converted or not into a non-negative network (as described in Section 2.2) and a monotonic multilayer perceptron (MLP) [33]. The non-monotonic encoder E is a single convolution/instance normalisation/LeakyReLU block (with a kernel size of 3 and 32 channels) for the weights initialization experiments and two such blocks (with 8 channels and respectively a kernel size of 7 and 3) for the interpretability experiments. For a fair comparison, the same encoder E has also been added to the non-monotonic baseline networks. PatchGAN was used for the pathological vs healthy classifications with Adadelta [34] optimizer and the initial learning rate set to 1. For MedNIST, all ResNet sizes were used and a ResNet152 trained with Adam [35] and a learning rate fixed to 10^{-5} was used for the final test accuracy evaluation. A ResNet34 was used for the tumor type classification with the Adam optimizer and a learning rate of 10^{-3}. Finally,



(a) Non-monotonic (b) Monotonic (c) Non-monotonic trained model

Figure 4: Standard deviation of the features as a function of the depth in a ResNet152 network for a given random input drawn in $\mathcal{N}(0, 1)$ (different from the one used for the proposed rescaling) using either the Kaiming initialization or ours. Left: original implementation. Middle: after our conversion into a monotonic network, the plot is in log-scale and with NaN value after the 18^{th} layer for Kaiming and constant with our proposition. Right: after training for the non-monotonic network.

the monotonic MLP was trained on Single-Cell as in [33]. For the computation of the L_{dF} loss (Eq.. 8), differentiable histograms with 50 bins and a triangle Parzen window were used for the distributions P_i of the negative interpretable features. We set $\alpha_G = 10^{-2}$. For the generation of the counterfactual examples, the Adam optimizer was used with a learning rate of 10^{-5}. We chose a margin $m = -15$ and a weight $\lambda = 10^{-4}$.

3.3. Metrics

The classification performances were measured with the mean accuracy. In Section 4.2, we detail the accuracy for the healthy class (TNR) and the pathological class (TPR). For the anomaly detection, the quality of the segmentation maps was measured using the Dice between the thresholded maps and ground truth pathology mask, the area under the precision-recall curve (AUPRC) and the receptive-operative curve (AUROC) considering voxel level classification. The thresholds were chosen as the PRC operating point on the validation dataset for each channel. In Section 4.2.2, the best validation channel has been chosen for our proposal and baselines (Baseline and BaselineC).

4. Results

4.1. Assessment of the proposed initialization and architecture

In this section, we verify that with the proposed initialization and architecture for non-negative neural networks, we obtain similar or better training through several indicators: feature variance, gradient, network convergence and performances.

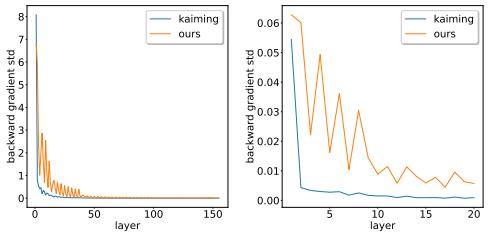
4.1.1. Features variance

In Fig. 4, the standard deviation of the initial internal features as a function of the layer in a ResNet152 is plotted for the Kaiming initialization and our procedure. The features were computed using a random noise input x_{test} different from the x_{train} minibatch used for the rescaling procedure. As one can see, the variance drastically increases with the layer depth with Kaiming but remains constant for our method (8 times higher for Kaiming on classical ResNet in Fig. 4a). This is even more true for the monotonic network for which variance goes to infinity from a certain layer for Kaiming while it remains equal to 1 for our initialization. As shown in Section 2.4.1, correlation between channels is increased by non-negative weights layers: for a standard PatchGAN, the channels are uncorrelated while in its non-negative counterpart, the correlation increases from 50% for the first layer to more than 80% for the fourth one. Thus, the uncorrelated input feature assumption required for the Kaiming method is not satisfied for non-negative networks: this could explain why it is not appropriate for these networks.

In Fig. 4c, the standard deviation of the features *after training* is plotted against the depth. One can see that, similarly to the standard deviation before training, the standard deviation after training is lower with our method. This implies that the initialization influences the final weights. We also can notice that the variance is larger with Adam optimizer [35] than with SGD.

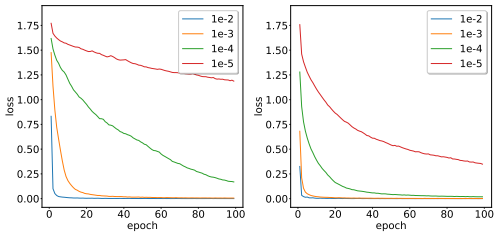
4.1.2. Gradient of the loss

The real objective of variance preservation is to avoid vanishing or exploding gradients. In Fig. 5, we plotted the standard deviation (mean is zero) of the gradient of the loss against the layer depth just after Kaiming or our initialization. Results show that with Kaiming, the gradient increases exponentially in the first layers whereas it is smoother with our initialization, especially with the very deep architecture ResNet152. Thus, the proposed initialization is less prone to exploding gradients.



(a) ResNet152 (b) ResNet18

Figure 5: Standard deviation of the first backward gradient as a function of the depth for a ResNet152 (left) and a ResNet18 (right). A comparison between Kaiming and the proposed initialization is made.



(a) Kaiming (b) Ours

Figure 6: Training loss for a ResNet152 network trained on MedNIST with Kaiming (left) or the proposed initialization (right) for different learning rates and SGD optimizer.

4.1.3. Network training convergence

Learning rate. In Fig. 6, the training curves on MedNIST are plotted for different learning rates with the SGD optimizer with our method or the Kaiming initialization. With a high learning rate, our proposition seems a little bit better with a faster convergence of the loss. The difference between the two initialization methods is more important with a lower learning rate. Indeed, with our initialization, the model convergence is better as the loss decreases faster. Our proposition seems less dependent to the learning rate than Kaiming initialization. Note that with an adaptive optimizer such as Adam, the difference between the two initialization methods is less visible.

Network depth. In this section, we evaluate the interplay between initialization and network depth. ResNets with different depths were trained on MedNIST and the differences between the training curves with a Kaiming initialization and with our initialization were plotted in Fig. 7. As the loss difference is always positive, the convergence is always better with our proposition. The difference is higher with deeper architecture like ResNet152. One can also notice a difference between architectures with bottleneck (ResNet50, 121 and 152) or not. Indeed, for the non-bottleneck architectures, the difference is the largest at the beginning of the training and then Kaiming initialized models seem to catch up the models initialized with our method.

Normalisation layers. A ResNet152 was trained on MedNIST with and without normalisation layers and the training curves were plotted in Fig. 8.

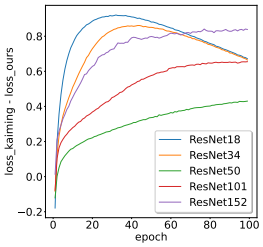


Figure 7: Training loss difference between Kaiming and the proposed initialization for different ResNet depths.

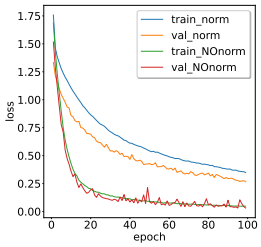


Figure 8: Training losses with and without normalisation layers in a ResNet152 with our initialization.

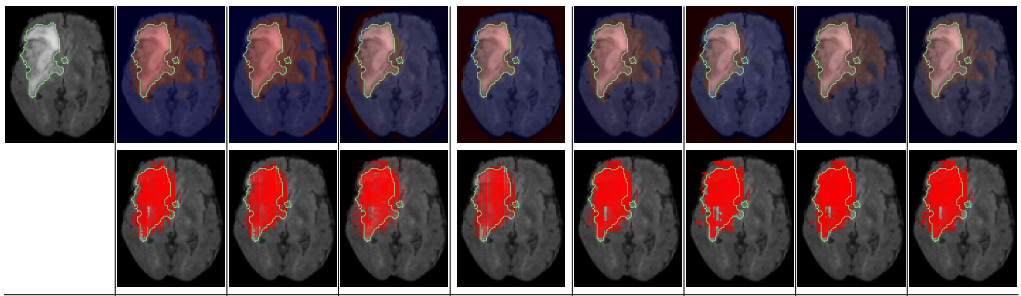
Without normalisation layers, the model diverges when it is initialized with Kaiming whereas it reaches a better convergence than the model with normalisation layers with our initialization. Normalisation layers seem less essential with our initialization.

4.1.4. Classification performances

Table 2: Test accuracy for monotonic and non-monotonic networks on different tasks. A comparison is made between Kaiming and our initialization. NaN means divergence.

Task	Standard		Monotonic	
	Kaiming	Ours	Kaiming	Ours
Single-Cell	/	/	0.83	**0.92**
Healthy vs tumors	**1.00**	**1.00**	0.98	**1.00**
Brain Tumors	0.72	**0.73**	NaN	**0.71**
MedNIST	**1.00**	**1.00**	NaN	**0.98**

A comparison of test classification accuracy for several architectures (monotonic or not) and classification tasks is reported in Table 2. For non-monotonic networks, accuracy is similar for both initializations. As variance in non-negative network rapidly diverges (Fig. 4), Kaiming initialization is unusable for deep architectures (Brain Tumors/ResNet34 or MedNIST/ResNet152) while our initialization enables a correct training. For smaller architectures, the model converges for both initializations but the accuracy of the MLP is 10 points lower with Kaiming initialization. Moreover, the convergence is faster with our proposition: when monotonic network are trainable with Kaiming, early stopping is reached well after early stopping with our initialization. Thus, any architecture can be converted into a monotonic network using the proposed method with improved model convergence and good performances.



Dice	0.55 ± 0.19	0.55 ± 0.19	0.55 ± 0.20	$\mathbf{0.56 \pm 0.19}$	0.54 ± 0.20	0.50 ± 0.19	0.56 ± 0.20	0.54 ± 0.20
AUROC	0.94	0.93	0.96	0.91	0.96	**0.97**	0.96	0.96
AUPRC	0.56	**0.58**	0.56	**0.58**	0.54	0.49	0.56	0.55

Figure 9: Example of interpretable features (top), counterfactual difference (α, bottom) and metrics (computed on the whole test database) for the channels of the interpretable features with our proposition. The tumor is delimited in green. The MRI is on the top left.

4.2. Weakly supervised anomaly detection and interpretable classification

In this section, we evaluate the interpretability of our classification and the segmentation performances.

4.2.1. Counterfactual examples for the channels of the interpretable features

We propose to use the difference α presented in Section 2.1.3 to visualize the areas of the image which are important for the decision, but also to use them for a weakly supervised segmentation of the pathological area. In Fig. 9, we can see each individual interpretable feature maps f, the counterfactual difference α as well as the segmentation metrics obtained for each channel when our proposition is used (constrained non-negative network as described in Section 2.5). We can see that all channels must be modified in the tumor region to obtain an image classified as healthy, even if the features do not only highlight the tumor. Thus, our network's decision seems relevant. Quantitatively, the metrics validate this hypothesis, with a Dice higher than 50% for all channels.

4.2.2. Weakly supervised segmentation

In the Fig. 10, we visually compare our proposition to state-of-the-art anomaly detection methods on two examples. For the first example, our method is comparable to Ross and Wargnier-Dauchelle. However, for the second example, our method seems to be the most focused on the pathology. We can also note that when the counterfactual examples are computed with

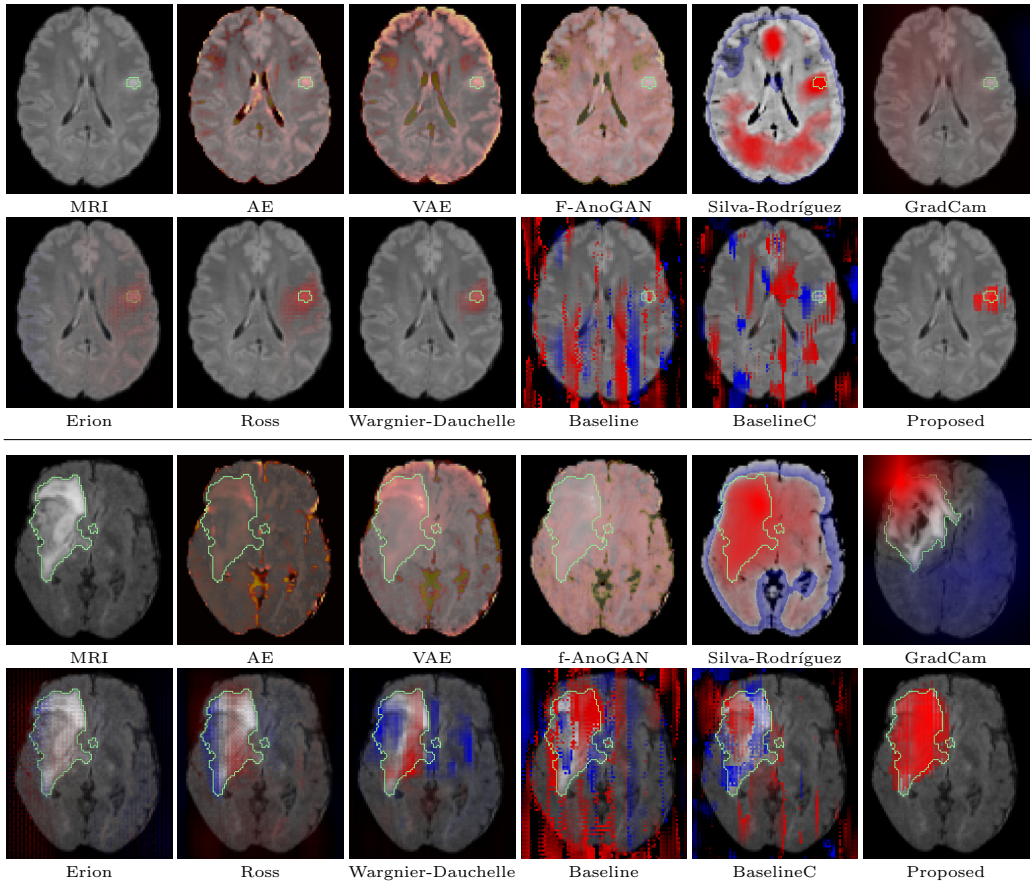


Figure 10: Segmentation maps for different methods. Manual annotation is drawn in green. Blue represents healthy relevance and red pathological relevance for classification attribution methods. High attributions are in red for Silva-Rodríguez. For reconstruction methods, reconstruction error scale is from black to yellow. For counterfactual examples, blue corresponds to negative values of α and red to positive values.

a non-monotonic network, either unconstrained (Baseline) or constrained (BaselineC), the results are clearly worse. First, the difference for generating the counterfactual example is both positive and negative, unlike non-negative networks. This is very detrimental to the reading and interpretation of the maps. Then, the network does not seem to use the presence of the tumor to make its decision since the strong values of these maps (negative and positive) are not particularly located in the tumor area. Mediocre results of Baseline and BaselineC are not surprising as the counterfactual example estimation process used can be viewed as an adversarial attack, not known to

Table 3: Comparison to state-of-the-art for tumor segmentation and image classification (whenever possible).

	Segmentation			Images class.	
Method	Dice	AUROC	AUPRC	TPR	TNR
AE	0.26 ± 0.11	0.90	0.16	/	/
VAE	0.25 ± 0.14	0.91	0.15	/	/
f-AnoFAN	0.17 ± 0.10	0.79	0.06	/	/
Silva-Rodríguez	0.37 ± 0.17	**0.92**	0.32	/	/
GradCam	0.12 ± 0.16	0.62	0.04	**1.00**	**1.00**
Erion	0.29 ± 0.15	0.70	0.19	**1.00**	**1.00**
Ross	0.48 ± 0.20	0.80	0.39	**1.00**	**1.00**
Wargnier-Dauchelle	0.51 ± 0.16	0.73	0.45	**1.00**	0.95
Baseline	0.04 ± 0.03	0.60	0.03	**1.00**	**1.00**
BaselineC	0.04 ± 0.03	0.66	0.04	**1.00**	**1.00**
Proposed	$\mathbf{0.56 \pm 0.19}$	0.91	**0.58**	**1.00**	**1.00**

always be readable. What is remarkable is that with the same process, our non-negative network with constrained learning produces such meaningful counterfactual examples.

Quantitative results are given in Table 3 for each method. Our method outperforms all state-of-the-art methods in terms of Dice and AUPRC, with perfect classification performances. The Dice is 5 points higher than Wargnier-Dauchelle, and AUPRC is 13 points higher. AUROC is similar to the best in state-of-the-art methods.

5. Conclusion

In this paper, we present MoE, a framework to improve the interpretability and explicability of a healthy vs pathological classifier based on a calibrated non-negative network. To do so, we propose a recipe to convert a network into a monotonic network. We also presented theoretical results explaining why state-of-the-art initialization methods cannot be used for non-negative networks and proposed an initialization procedure that enables deep non-negative networks to be trained in a stable way and also benefits to classical non-monotonic networks. A formal characterization of two-layer monotonic networks is also presented, allowing to alleviate costly process such as [13]. Experiments show that, with our method, models converge and reach good classification performances while benefiting from the interesting properties of non-negative networks, especially in terms of explicability. Using

constraints on the interpretable features and the gradient of the monotonic part of this network, we show that the classifier is more interpretable as the counterfactual examples are more readable and in accordance with the radiological signs of the pathology. In addition, our method outperforms state-of-the-art methods for weakly supervised anomaly detection. In future works, the proposed classifier could be included into other frameworks like adversarial networks or guided diffusion models to increase their performances. It could also be used for other tasks like regression or prediction.

Acknowledgments

This work was supported by the LABEX PRIMES (ANR-11-LABX-0063, ANR-11-IDEX-0007) in a lab member of France Life Imaging network (ANR-11-INBS-0006). This work was performed using HPC resources from GENCI-IDRIS (AD011012544/AD011012589). Finally, this work was partly funded by APIDIFF: "Projet Emergence", CNRS-INS2I.

References

[1] V. Wargnier-Dauchelle, T. Grenier, F. Durand-Dubief, F. Cotton, M. Sdika, A more interpretable classifier for multiple sclerosis, in: 18th Int. Symposium on Biomed. Imag., IEEE, 2021, pp. 1062–1066.

[2] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proc. of the IEEE Int. conf. on computer vision, 2017, pp. 618–626.

[3] G. Erion, J. D. Janizek, P. Sturmfels, S. M. Lundberg, S.-I. Lee, Improving performance of deep learning models with axiomatic attribution priors and expected gradients, Nature Machine Intelligence (2021) 1–12.

[4] A. S. Ross, M. C. Hughes, F. Doshi-Velez, Right for the right reasons: training differentiable models by constraining their explanations, in: Proc. of the 26th Int. Joint Conf. on AI, 2017, pp. 2662–2670.

[5] V. Wargnier-Dauchelle, T. Grenier, F. Durand-Dubief, F. Cotton, M. Sdika, A weakly supervised gradient attribution constraint for interpretable classification and anomaly detection, IEEE Trans. on Med. Imag. (2023).

[6] C. Baur, B. Wiestler, S. Albarqouni, N. Navab, Deep autoencoding models for unsupervised anomaly segmentation in brain mr images, in: Int. MICCAI brainlesion workshop, Springer, 2018, pp. 161–169.

[7] D. Zimmerer, F. Isensee, J. Petersen, S. Kohl, K. Maier-Hein, Unsupervised anomaly localization using variational auto-encoders, in: Int. Conf. on Med. Image Computing and Computer-Assisted Intervention, Springer, 2019, pp. 289–297.

[8] J. Silva-Rodríguez, V. Naranjo, J. Dolz, Looking at the whole picture: constrained unsupervised anomaly segmentation, in: BMVC, 2021.

[9] T. Schlegl, P. Seeböck, S. M. Waldstein, G. Langs, U. Schmidt-Erfurth, f-anogan: Fast unsupervised anomaly detection with generative adversarial networks, Med. image analysis 54 (2019) 30–44.

[10] H. Daniels, M. Velikova, Monotone and partially monotone neural networks, IEEE Trans. on Neural Networks 21 (6) (2010) 906–917.

[11] J. Sill, Monotonic networks, Advances in neural inf. processing systems 10 (1997).

[12] S. You, D. Ding, K. Canini, J. Pfeifer, M. Gupta, Deep lattice networks and partial monotonic functions, Advances in neural inf. processing systems 30 (2017).

[13] X. Liu, X. Han, N. Zhang, Q. Liu, Certified monotonic neural networks, Advances in Neural Inf. Processing Systems 33 (2020) 15427–15438.

[14] S. Sivaprasad, A. Singh, N. Manwani, V. Gandhi, The curious case of convex neural networks, in: Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proc., Part I 21, Springer, 2021, pp. 738–754.

[15] D. Runje, S. M. Shankaranarayana, Constrained monotonic neural networks, in: Int. Conf. on Machine Learning, PMLR, 2023, pp. 29338–29353.

[16] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proc. of the 13th Int. Conf. on Artificial Intelligence and Statistics, Vol. 9 of Proc. of Machine Learning Research, PMLR, 2010, pp. 249–256.

[17] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proc. of the IEEE Int. Conf. on Computer Vision (ICCV), 2015.

[18] D. Mishkin, J. Matas, All you need is a good init, in: 4th Int. Conf. on Learning Representations (ICLR), 2016.

[19] S. Wachter, B. Mittelstadt, C. Russell, Counterfactual explanations without opening the black box: Automated decisions and the gdpr, Harv. JL & Tech. 31 (2017) 841.

[20] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Int. conf. on machine learning, pmlr, 2015, pp. 448–456.

[21] D. Ulyanov, A. Vedaldi, V. Lempitsky, Instance normalization: The missing ingredient for fast stylization, arXiv:1607.08022 (2016).

[22] J. E. Gentle, Computational statistics, Springer, 2010.

[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. of the IEEE conf. on computer vision and pattern recognit., 2016, pp. 770–778.

[24] D. Varga, A. Csiszárik, Z. Zombori, Gradient regularization improves accuracy of discriminative models, Schedae Informaticae 27 (2018).

[25] S. Bhuvaji, A. Kadam, P. Bhumkar, S. Dedge, S. Kanchan, Brain tumor classification (mri) (2020). doi:10.34740/KAGGLE/DSV/1183165.

[26] J. H. Levine, E. F. Simonds, S. C. Bendall, K. L. Davis, D. A. El-ad, M. D. Tadmor, O. Litvin, H. G. Fienberg, A. Jager, E. R. Zunder, et al., Data-driven phenotypic dissection of aml reveals progenitor-like cells that correlate with prognosis, Cell 162 (1) (2015) 184–197.

[27] A. Babayan, M. Erbey, D. Kumral, J. D. Reinelt, A. M. Reiter, J. Röbbig, H. L. Schaare, M. Uhlig, A. Anwander, P.-L. Bazin, et al., A mind-brain-body dataset of mri, eeg, cognition, emotion, and peripheral physiology in young and old adults, Scientific data 6 (1) (2019) 1–21.

[28] B. A. Landman, A. J. Huang, A. Gifford, D. S. Vikram, I. A. L. Lim, J. A. Farrell, J. A. Bogovic, J. Hua, M. Chen, S. Jarso, et al., Multi-parametric neuroimaging reproducibility: a 3-t resource study, Neuroimage 54 (4) (2011) 2854–2866.

[29] A. L. Pinho, A. Amadon, T. Ruest, M. Fabre, E. Dohmatob, I. Denghien, C. Ginisty, S. Becuwe-Desmidt, S. Roger, L. Laurier, et al., Individual brain charting, a high-resolution fmri dataset for cognitive mapping, Scientific data 5 (1) (2018) 1–15.

[30] M. Jenkinson, P. Bannister, M. Brady, S. Smith, Improved optimization for the robust and accurate linear registration and motion correction of brain images, Neuroimage 17 (2) (2002) 825–841.

[31] F. Isensee, M. Schell, I. Pflueger, G. Brugnara, D. Bonekamp, U. Neuberger, A. Wick, H.-P. Schlemmer, S. Heiland, W. Wick, et al., Automated brain extraction of multisequence mri using artificial neural networks, Human brain mapping 40 (17) (2019) 4952–4964.

[32] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros, Image-to-image translation with conditional adversarial networks, in: Proc. of the IEEE conf. on computer vision and pattern recognit., 2017, pp. 1125–1134.

[33] A.-P. Nguyen, D. L. Moreno, N. Le-Bel, M. Rodríguez Martínez, Mononet: Enhancing interpretability in neural networks via monotonic features, Bioinform. Advances (2023) vbad016.

[34] M. D. Zeiler, Adadelta: an adaptive learning rate method, arXiv preprint arXiv:1212.5701 (2012).

[35] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).