ISNet: Costless and Implicit Image Segmentation for Deep Classifiers, with Application in COVID-19 Detection

Pedro R.A.S. Bassi1,* and Andrea Cavalli1,*

1Alma Mater Studiorum - University of Bologna, Bologna, Italy
*e-mail: pedro.salvadorbassi2@unibo.it, andrea.cavalli@unibo.it

ABSTRACT

This work proposes a novel deep neural network (DNN) architecture, Implicit Segmentation Neural Network (ISNet), to solve the task of image segmentation followed by classification. It substitutes the common pipeline of two DNNs with a single model. We designed the ISNet for high flexibility and performance: it allows virtually any classification neural network architecture to analyze a common image as if it had been previously segmented. Furthermore, in relation to the unmodified classifier, the ISNet does not cause any increment in computational cost at run-time. We test the architecture with two applications: COVID-19 detection in chest X-rays, and facial attribute estimation. We implement an ISNet based on a DenseNet121 classifier, and compare the model to a U-net (performing lung/face segmentation) followed by a DenseNet121, and to a standalone DenseNet121. The new architecture matched the other DNNs in facial attribute estimation. Moreover, it strongly surpassed them in COVID-19 detection, according to an external test dataset. The ISNet precisely ignored the image regions outside of the lungs or faces. Therefore, in COVID-19 detection it reduced the effects of background bias and shortcut learning, and it improved security in facial attribute estimation. ISNet presents an accurate, fast, and light methodology. The successful implicit segmentation, considering two largely diverse fields, highlights the architecture’s general applicability.

Keywords: COVID-19 Detection; Chest X-ray; Layer-wise Relevance Propagation; Image Segmentation; Neural Networks; Facial Attribute Estimation

1 Introduction

Deep neural networks (DNNs) are complex models with potentially millions of trainable parameters. Due to this intrinsic aspect, it is challenging to interpret them and to understand why a classifier makes a certain decision. However, this understanding is paramount for critical tasks, such as diagnosis assisted by artificial intelligence (AI). Layer-wise relevance propagation (LRP) is a technique designed to create heatmaps for deep classifiers. Heatmaps are graphics that explain the model’s behavior by making explicit how each part of an input image affects the DNN output. We can create a heatmap for an input image and one class. If positive, the values in the map (i.e. relevance) indicate how strongly the classifier associated the image regions with the class. If negative, they represent areas that decreased the classifier confidence in the class (e.g., regions associated with the other possible classes). LRP is a technique created to improve DNN explainability, and, if bias is present in the model, it is challenging to interpret them and to understand why a classifier makes a certain decision. However, this understanding is paramount for critical tasks, such as diagnosis assisted by artificial intelligence (AI). Layer-wise relevance propagation (LRP) is a technique designed to create heatmaps for deep classifiers. Heatmaps are graphics that explain the model’s behavior by making explicit how each part of an input image affects the DNN output. We can create a heatmap for an input image and one class. If positive, the values in the map (i.e. relevance) indicate how strongly the classifier associated the image regions with the class. If negative, they represent areas that decreased the classifier confidence in the class (e.g., regions associated with the other possible classes). LRP is a technique created to improve DNN explainability, and, if bias is present in the figure background and affects the classification process, LRP will reveal it.

Background features can divert the focus of a classifier, reducing its ability to interpret the useful information in an image. They can carry information that either conflicts with what we wish to classify, confounding the classifier, or that correlates with the classes, biasing the classifier. If a background bias is also present during evaluation of DNNs, it artificially improves performance metrics. This scenario favors shortcut learning 2, a condition where deep neural networks learn decision rules that perform well on standard benchmarks, but poorly on real-world applications. Image segmentation is the task of semantically partitioning a figure, assigning a category to each of its pixels. The process allows a differentiation between objects, and the identification of the background. Segmentation is a powerful tool to address the problem of background bias. When applied as a preprocessing step prior to classification, it allows the removal of the image background, preventing the classifier from learning with its features. Therefore, for classification problems that benefit from segmentation, a common pipeline has two DNNs. The first DNN segments the important regions of the image. Its output is used to erase the background, creating a segmented image that the second DNN classifies.

In this work, we propose a drastically different strategy for performing image segmentation before classification. We created a novel DNN architecture that is capable of implicitly segmenting its input. It does not produce a segmentation mask, but it learns to precisely select the image regions considered for the classification process. We call this model Implicit Segmentation Neural Network or ISNet. After training with images and segmentation targets, the architecture learns how to classify unsegmented images as if they had previously been segmented by another DNN. During the training procedure, an ISNet comprises a classifier and another structure, which we call ‘LRP block’. The novel structure shares parameters with the classifier, and is linked to it via multiple skip connections. Its job is to perform Layer-wise Relevance Propagation through the classifier, producing heatmaps for all possible classes. Besides proposing a strategy to implement LRP rules as DNN layers (which form the LRP block), the ISNet introduces the concept of relevance segmentation in the heatmaps. That is, the ISNet learns to minimize the classifier’s undesired/background relevance, forcing it to focus only on the regions of interest, resulting in implicit image segmentation.
A major advantage of the ISNet architecture is that, after the training procedure, the LRP block can be removed or inactivated without altering the classifier behavior. It will still only focus on the regions of interest. Consequently, the ISNet presents no additional computational cost at run-time relative to a common classifier. In contrast to our proposal, the accuracy of a standard classification neural network will drop significantly if it is trained with segmented images and later receives unsegmented samples during run-time. The ISNet allows a single model to perform the task of two neural networks, a segmenter and a classifier. Thus, it creates a strong economy of memory and processing time. This benefit is even more solid when we consider models that will be deployed in portable or less capable devices. Furthermore, the architecture is very flexible. Virtually any classification neural network can be converted to an ISNet. In this work, we opted for a Densely Connected Convolutional Network (DenseNet) as the classifier.

Currently, the most popular open databases of COVID-19 chest X-rays contain either no or few COVID-19 negative/control cases. Due to this limitation, to this date most studies resorted to mixed source datasets to train DNNs to differentiate between COVID-19 patients, healthy people, and patients with other diseases (e.g. non-COVID-19 pneumonia). In these datasets, different classes come from different databases assembled in distinct hospitals and cities. This raises concerns that the dissimilar sources could contain different biases, which may help DNNs to classify the images according to their source dataset, rather than according to the disease symptoms. One study accurately determined the source datasets after removing all lung regions from the images, proving the presence of background bias. A review concluded that, if models are allowed to analyze the whole X-ray or a rectangular bounding box around the lungs, they tend to strongly focus on areas outside of the lungs. Thus, they fail to generalize or achieve satisfactory performance on external datasets (i.e., datasets with dissimilar sources in relation to the training images). Moreover, the review identified that the problem is a cause of shortcut learning. A paper utilized external datasets to evaluate traditional COVID-19 detection DNNs, whose reported results had been highly positive, and it demonstrated a strong drop in their performances. For these reasons, researchers have resorted to testing on external datasets to understand the bias and generalization capability of DNNs trained on mixed databases. The approach shows reduced and more realistic performances in relation to the standard evaluation methodologies. A common conclusion of these works is that lung segmentation improves generalization capability, reducing the bias induced by mixed training datasets. Therefore, we believe that the ISNet is suited to the task of COVID-19 detection using the usual mixed datasets of chest X-rays.

To test the precision of the ISNet architecture’s implicit segmentation, we trained it on a mixed dataset based on one of the largest open COVID-19 chest X-ray databases, with the objective of distinguishing COVID-19 positive cases, normal images, and non-COVID-19 pneumonia. The two diseases have similar symptoms, making their differentiation non-trivial, and both produce signs in chest X-rays. Examples of COVID-19 signs are bilateral radiographic abnormalities, ground-glass opacity, and interstitial abnormalities. Examples of pneumonia signals are poorly defined small centrilobular nodules, airspace consolidation, and bilateral asymmetric ground-glass opacity. We evaluate the optimized DNN in a cross-dataset approach, using images collected from external locations in relation to the training samples. This evaluation methodology can also be described as testing with an out-of-distribution (ood) dataset, and it shall allow us to understand if ignoring the image background increases generalization performance. We present more details about the databases in section 4.8.

To evaluate the general applicability of the ISNet architecture, and its ability to achieve implicit segmentation in diverse domains, we trained the DNN for a second task: facial attribute estimation. We utilized 30000 samples from the CelebA database, which is composed of natural images, in three channels, displaying people in a wide variety of poses, and containing background clutter. Using these in the wild images, the ISNet will classify facial features, while implicitly segmenting the face region, which can assume many shapes, sizes, and locations in the figures. Facial attribute estimation is a multi-label classification problem, and it is fundamentally different from COVID-19 detection in chest X-rays. Since a paper indicated that the classification of more attributes causes the classifiers to naturally focus more on the faces, we opted to classify only 3 facial attributes (rosy cheeks, high cheekbones and smiling), increasing the implicit segmentation difficulty. Past works did not necessarily search for, or found, an accuracy improvement when using face segmentation before facial attribute estimation. However, they pointed out that background features may influence classifiers. Therefore, segmentation can reduce bias and improve security, which is crucial when the classified attributes will be used for user identification. In this work, we utilize the facial attribute estimation task to assess the ISNet segmentation capability in evaluating natural images, and in addressing a challenging segmentation problem. For this reason, we will not employ a machine learning pipeline that localizes, aligns, and crops around the face before performing classification (a common approach to increase accuracy).

As a more extreme test for the ISNet, we also trained it on artificially biased versions of the datasets, for both COVID-19 detection and facial attribute estimation. We added different geometrical shapes in the image backgrounds, according to their classes. A standard classifier trained on this dataset will show increased performance on images containing the shapes, but will lose accuracy for samples without the artificial bias, clearly displaying the effects of shortcut learning. If the ISNet implicit segmentation is successful, it should produce the same result in both cases, and the analysis of its heatmaps will show no LRP relevance over the artificial shapes. Finally, in all tasks we compared the ISNet containing a DenseNet121 classifier to a standard segmentation-classification pipeline with 2 DNNs, comprising a U-Net segmenter followed by a DenseNet121 classifier. We also compared our proposal to a standalone DenseNet121, without prior image segmentation. The ISNet PyTorch implementation is publicly available at https://github.com/PedroRASB/ISNet.

2 Results

2.1 COVID-19 Detection

We analyzed the DNNs on the test dataset for COVID-19 detection, with the three classes (COVID-19, normal and non-COVID-19 pneumonia) collected from distinct localities/datasets in relation to the training database. Table 1 shows class and average precision, recall, F1-Score and specificity for the trained DNNs (ISNet, U-Net followed by DenseNet121 and standalone
Table 1. Performance metrics for the deep neural networks in COVID-19 detection

| Class     | Precision | Recall     | F1-Score   | Specificity |
|-----------|-----------|------------|------------|-------------|
| ISNet     |           |            |            |             |
| Normal    | 0.381 +/-0.016, [0.35,0.413] | 0.917 +/-0.014, [0.889,0.944] | 0.538 +/-0.017, [0.505,0.571] | 0.804 +/-0.007, [0.789,0.818] |
| Pneumonia | 0.928 +/-0.009, [0.91,0.946] | 0.556 +/-0.014, [0.529,0.583] | 0.696 +/-0.012, [0.673,0.718] | 0.97 +/-0.004, [0.963,0.978] |
| COVID-19  | 0.921 +/-0.007, [0.908,0.934] | 0.92 +/-0.007, [0.906,0.934] | 0.92 +/-0.005, [0.91,0.93] | 0.928 +/-0.006, [0.916,0.94] |
| Macro-average | 0.743 +/-0.007, [0.73,0.756] | 0.798 +/-0.007, [0.783,0.811] | 0.718 +/-0.008, [0.702,0.735] | 0.901 +/-0.003, [0.894,0.907] |

U-Net segmenter followed by DenseNet121 classifier

| Class     | Precision | Recall     | F1-Score   | Specificity |
|-----------|-----------|------------|------------|-------------|
| Normal    | 0.446 +/-0.019, [0.408,0.483] | 0.796 +/-0.021, [0.756,0.837] | 0.571 +/-0.018, [0.535,0.607] | 0.869 +/-0.006, [0.857,0.882] |
| Pneumonia | 0.791 +/-0.015, [0.763,0.82] | 0.466 +/-0.014, [0.439,0.494] | 0.586 +/-0.013, [0.561,0.611] | 0.915 +/-0.006, [0.903,0.028] |
| COVID-19  | 0.723 +/-0.011, [0.702,0.744] | 0.838 +/-0.009, [0.819,0.856] | 0.776 +/-0.008, [0.76,0.792] | 0.708 +/-0.011, [0.686,0.73] |
| Macro-average | 0.653 +/-0.009, [0.636,0.67] | 0.7 +/-0.009, [0.683,0.717] | 0.645 +/-0.009, [0.626,0.663] | 0.831 +/-0.004, [0.823,0.839] |

DenseNet121 without segmentation

| Class     | Precision | Recall     | F1-Score   | Specificity |
|-----------|-----------|------------|------------|-------------|
| Normal    | 0.364 +/-0.02, [0.324,0.402] | 0.57 +/-0.026, [0.518,0.618] | 0.444 +/-0.02, [0.403,0.482] | 0.869 +/-0.006, [0.856,0.881] |
| Pneumonia | 0.827 +/-0.018, [0.792,0.861] | 0.294 +/-0.013, [0.27,0.32] | 0.434 +/-0.015, [0.405,0.463] | 0.958 +/-0.005, [0.949,0.967] |
| COVID-19  | 0.649 +/-0.01, [0.629,0.67] | 0.916 +/-0.007, [0.902,0.93] | 0.76 +/-0.008, [0.744,0.775] | 0.549 +/-0.012, [0.525,0.573] |
| Macro-average | 0.614 +/-0.009, [0.594,0.631] | 0.594 +/-0.01, [0.574,0.612] | 0.546 +/-0.01, [0.527,0.565] | 0.792 +/-0.004, [0.784,0.8] |

Results are reported in the following format: mean +/--std, [95% HDI]. Mean refers to the metric’s mean value, according to its probability distribution, and std to its standard deviation. 95% HDI indicates the 95% highest density interval, defined as an interval containing 95% of the metric’s probability mass. Furthermore, any point inside the interval must have a probability density that is higher than that of any point outside. Table 1 indicates in bold text the models’ balanced accuracy (i.e., average recall) and macro-averaged F1-Score (maF1). The ISNet, alternative segmentation-classification pipeline, and DenseNet121 achieved 0.938, 0.833 and 0.808 AUC, respectively.

In all considered test metrics macro-averages (Table 1) the ISNet obtained the highest results, followed by the alternative segmentation-classification pipeline, and finally by the model without segmentation. We also observe that the different models’ 95% highest density intervals do not overlap for any average performance measurement. Consequently, the probability of the models having equivalent performances is minute. Relative to the alternative segmentation methodology, the ISNet achieved a 7.3% increment in macro-averaged F1-Score. In relation to the DenseNet121 without segmentation, the difference rises to 17.2%. In this work, our segmentation targets were automatically generated by a U-Net, trained for lung segmentation in another study. Therefore, we believe that the performances achieved by the ISNet and by the alternative segmentation-classification pipeline could increase even more, provided a dataset containing a large amount of chest X-rays accompanied by manually segmented lung regions.

The reported results may seem worse than other studies in the field of COVID-19 detection, which report remarkably high performances, strongly surpassing expert radiologists (e.g., F1-Scores close to 100%). However, evidence suggests that, currently, such results are obtained when the training and test datasets come from the same distribution/sources (i.e., datasets that are independent and identically distributed, or iid). Moreover, studies show that these strong performances may be boosted by bias and shortcut learning, preventing the neural networks from generalizing, or achieving comparable results in the real-world. Instead, the performances obtained in this study are in accordance with other works that evaluate their DNNs in external databases (i.e., out-of-distribution or ood datasets, whose image sources are diverse in relation to the ones that generate the training samples). For example, an article reported AUC of 0.786 +/-0.025 on an external dataset, when evaluating a DenseNet121 used for COVID-19 detection without lung segmentation. Here, the DenseNet121 obtained 0.808 AUC, which falls into their reported confidence interval. Another paper evaluates COVID-19 detection and utilizes lung segmentation before classification with a DenseNet201. They achieved mean maF1 of 0.754, with 95% HDI of [0.687,0.82], also evaluating their model on an external dataset. We observe that the ISNet maF1 95% HDI, [0.702,0.735], fits inside their reported 95%
We must note that the aforementioned studies use different databases. Thus, caution is required when directly comparing the numerical results.

We also trained the ISNet with the artificially biased dataset (altered with polygons representing the X-ray class), and tested it on the external test dataset, also artificially biased, or in its original version. For both versions, the neural network produced exactly the same confusion matrix. The model’s result was very similar to the ISNet trained in the common dataset (Table 1), presenting a macro-average F1-Score of 0.717 +/-0.01 (and [0.699, 0.736] 95% HDI). If we perform the same procedure using a DenseNet121 without segmentation, it achieves 0.775 +/-0.008 ([0.768, 0.796] 95% HDI) maF1 in the biased test dataset, but the score drops to 0.434 +/-0.01 ([0.414, 0.454] 95% HDI), for the original test database. Consequently, we see that the polygons in the background strongly diverge the normal classifier focus, causing clear shortcut-learning. However, they have no effect on the ISNet, demonstrating the effectiveness of its implicit segmentation. Moreover, the ISNet is resistant to image flipping, rotations, and translations. These operations, used as data augmentation, did not negatively affect the training procedure, nor did they worsen the validation error during preliminary tests with an augmented validation dataset.

To illustrate that it is not possible to simply train a model with segmented images, and then use it without segmentation at run-time, we tested the alternative segmentation methodology after removing the U-Net from its pipeline. That is, we simulated a DenseNet121 trained on segmented images and used to classify unsegmented ones. As expected, this resulted in a dramatic performance drop: maF1 was reduced from 0.645 +/-0.009 to 0.217 +/-0.003 (changing its 95% HDI from [0.626, 0.663] to [0.211, 0.224]). The test shows the advantage of implicit segmentation, as it eliminates the need for a segmentation model during run-time.

As a final note, the classification of chest X-rays as COVID-19, normal or pneumonia constitutes a multi-class, single-label problem. To find interval estimates for the classification performance metrics we employed Bayesian estimation. We utilized a Bayesian model that takes a confusion matrix (displayed in Table 2) as input, and estimates the posterior probability distribution of metrics like class and average precision, recall and F1-Score. The model was expanded in a subsequent study, to also estimate class and average specificity. For the posteriors’ estimation we used Markov chain Monte Carlo (MCMC), implemented with the Python library PyMC3. We employed the No-U-Turn Sampler, using 4 chains and 110000 samples, being the first 10000 for tuning. The estimated distributions allow us to report the means, standard deviations, and the 95% highest density intervals of the performance metrics. The estimation Monte Carlo error is below 10^-4 for all scores. We utilized a macro-averaging and pairwise approach to calculate the area under the ROC curve (AUC), a technique developed for multi-class single-label problems. We do not present interval estimates for this metric, because there is not an established methodology to calculate it (its authors suggest bootstrapping, but it would not be feasible with the deep models and cross-dataset evaluation used in this study).

| True Class | Predicted Class |
|------------|-----------------|
| Normal     | Pneumonia       |
| COVID-19   |                 |
| ISNet      |
| Normal     | 341             |
| Pneumonia  | 466             |
| COVID-19   | 85              |
| U-Net + DenseNet121 |
| Normal     | 296             |
| Pneumonia  | 271             |
| COVID-19   | 95              |
| DenseNet121 |
| Normal     | 211             |
| Pneumonia  | 306             |
| COVID-19   | 63              |

### Table 2. Confusion matrices for the deep neural networks in COVID-19 detection

#### 2.2 Facial Attribute Estimation

Table 3 shows the test performance of the three DNNs in the facial attribute estimation task, analyzing three attributes (rosy cheeks, high cheekbones and smiling). The table cells report the metrics’ mean and error, considering 95% confidence. The ISNet, alternative segmentation-classification pipeline, and DenseNet121 achieved 0.95 +/-0.008, 0.954 +/-0.007 and 0.952 +/-0.008 macro-average AUC, respectively. We observe that, in this task, the three models have similar performances, presenting strong overlap in the 95% confidence intervals for every average score in Table 3, and for AUC.

Training the ISNet in the artificially biased version of the CelebA dataset yielded the same outcome as in the COVID-19 detection task: the proposed model completely ignored the polygons added to the background. The new ISNet achieved the same macro-average F1-Score (0.803 +/-0.027) and balanced accuracy (0.815 +/-0.027) for original test dataset and for its biased version. Unsurprisingly, a common DenseNet121 trained under the same circumstances did not: it generated 0.974 +/-0.012 maF1 on the biased database, and 0.641 +/-0.054 on the original. We again observe that the strong background bias distracts common classifiers, and causes clear shortcut learning. However, it does not affect the ISNet, due to the successful implicit segmentation. Therefore, we see that the ISNet represents a security improvement for facial attribute estimation.
### Table 3. Performance metrics for the deep neural networks in facial attribute estimation

| Class                | ISNet Prec. | ISNet Rec. | ISNet F1-Score | U-Net seg. Prec. | U-Net seg. Rec. | U-Net seg. F1-Score | DenseNet Prec. | DenseNet Rec. | DenseNet F1-Score |
|----------------------|-------------|------------|----------------|-----------------|----------------|---------------------|----------------|--------------|------------------|
| Rosy Cheeks          | 0.647 +/-0.05 | 0.686 +/-0.048 | 0.666 +/-0.049 | 0.615 +/-0.048 | 0.707 +/-0.045 | 0.658 +/-0.047 | 0.624 +/-0.05 | 0.671 +/-0.048 | 0.647 +/-0.049 |
| High Cheekbones      | 0.876 +/-0.018 | 0.81 +/-0.022 | 0.842 +/-0.02 | 0.863 +/-0.018 | 0.843 +/-0.019 | 0.853 +/-0.019 | 0.876 +/-0.018 | 0.822 +/-0.021 | 0.848 +/-0.02 |
| Smiling              | 0.937 +/-0.013 | 0.892 +/-0.017 | 0.914 +/-0.015 | 0.905 +/-0.016 | 0.912 +/-0.015 | 0.908 +/-0.016 | 0.926 +/-0.014 | 0.897 +/-0.016 | 0.911 +/-0.015 |
| Mean                 | 0.82 +/-0.027 | 0.796 +/-0.029 | 0.807 +/-0.028 | 0.794 +/-0.027 | 0.821 +/-0.026 | 0.806 +/-0.027 | 0.809 +/-0.027 | 0.797 +/-0.028 | 0.802 +/-0.028 |

Furthermore, the ISNet ignores undesired features even when analyzing natural images, with cluttered backgrounds, and a wide variety of segmentation masks’ shapes, locations, and sizes.

In facial attribute estimation, removing the U-Net before testing the alternative segmentation methodology causes a devastating effect, as it did in COVID-19 detection: it reduced maF1 from 0.806 +/-0.027 to 0.543 +/-0.191. Facial attribute estimation is a multi-class multi-label classification problem. Therefore, the previously utilized Bayesian model is not adequate for it. To create the interval estimates in Table 3, we employ the Wilson Score Interval. Working with a multi-label task, we can calculate the AUC 95% confidence interval with an already established non-parametric approach.

### 2.3 Speed and Efficiency Analysis

All training procedures were executed in a NVidia RTX 3080 graphics processing unit (GPU) using mixed precision. In facial attribute estimation an epoch considered 24183 train images and 2993 validation samples, loaded in mini-batches of 10 figures. An epoch took about 1300 s for the ISNet, 320 s for the alternative segmentation model, and 240 s for the standalone DenseNet121. It is worth noting that the alternative pipeline also required training a U-Net, which needed 400 epochs of about 180 s. Considering this additional training procedure, the ISNet training time was about 82% longer than the alternative segmentation-classification pipeline’s.

During training, the ISNet is slower than a traditional pipeline of segmentation and classification. The model’s biggest limitation may be the need to generate one heatmap per class. Treating the different heatmap calculations as a propagation of different batch elements allows parallelism, but training time and memory consumption still increases with more classes. The worst-case scenario for this problem is if memory is not sufficient for the batch approach. In this case, the heatmaps will need to be created in series, causing training time to increase linearly with the number of classes. Therefore, more efficient implementations for the LRP block are a promising path for future developments.

At run-time, the ISNet shows clear benefits because the LRP block can be removed from the model, leaving only the classifier. With a NVidia RTX 3080 GPU employing mixed precision and using mini-batches of 10 images, the standard pipeline of a U-Net followed by a DenseNet121 classifies an average of 207 samples per second, while the DenseNet121-based ISNet classifies an average of 353. Utilizing the same GPU and mini-batch size, but without mixed precision, the alternative classification pipeline classifies 143 samples per second, and the ISNet, 298. Therefore, the ISNet is about 70% to 108% faster in this configuration. However, there is an even greater advantage regarding the model’s size: the ISNet has about 8M parameters (the same as the DenseNet121), while the combined U-Net and DenseNet121 have 39M. Thus, the model size is reduced by a factor of almost 5. Naturally, the smaller the classifier model in relation to the segmentation network, the stronger the performance benefit provided by the ISNet architecture.

### 2.4 Heatmap Analysis

Besides being necessary for the calculation of the heatmap loss, the heatmaps created by our LRP block can be analyzed, improving the explainability of our models. Figure 1 presents test images and the associated heatmaps for the two classification problems, according to the three implemented DNNs. For better visualization, we normalized the maps presented in this section, dividing them by their standard deviations. In the X-rays heatmaps, red (positive relevance) indicates areas more associated with COVID-19, and blue (negative relevance) with normal or non-COVID-19 pneumonia. In facial attribute estimation, red and blue colors point positive and negative relevance for the attribute rosy cheeks. In every case, the whiter/brighter a heatmap region, the less the classifier focused on it. We observe that both the ISNet and the alternative segmentation model (U-Net followed by DenseNet121) successfully kept the relevance inside the key areas (lungs/faces), with little to no background attention. The same is not true for the model without segmentation.
| Input Image | ISNet Heatmap | U-Net+DenseNet Heatmap | No Segmentation Heatmap |
|-------------|---------------|------------------------|-------------------------|
| ![Image](image1.png) | ![Heatmap](heatmap1.png) | ![Heatmap](heatmap2.png) | ![Heatmap](heatmap3.png) |
| ![Image](image2.png) | ![Heatmap](heatmap4.png) | ![Heatmap](heatmap5.png) | ![Heatmap](heatmap6.png) |
| ![Image](image3.png) | ![Heatmap](heatmap7.png) | ![Heatmap](heatmap8.png) | ![Heatmap](heatmap9.png) |

**Figure 1.** Heatmaps for the three deep neural networks. Red colors indicate association to the class COVID-19 (X-rays) or rosy cheeks (photographs).

The first two rows of figure 1 show COVID-19 positive X-rays, correctly classified. The third row has an image from a healthy patient, only misclassified by the model without segmentation, which predicted COVID-19. In this image, we observe that the DenseNet121 strongly focuses on text present in the X-ray upper-right region, and associate it to COVID-19.
Interestingly, similar marks were common in the COVID-19 images that we used for training, which may explain the association. In the other two X-ray no segmentation heatmaps we see background attention in, but not limited to, the markings outside of the lungs (upper-left corners). This problem is not observed in the ISNet or in the alternative segmentation-classification pipeline heatmaps. If the model without segmentation miss-classified many X-rays as COVID-19 due to text and markings in the images, its COVID-19 class specificity should be smaller than the other models'. Indeed, in Table 1 we observe a strong drop in COVID-19 specificity when segmentation is not employed (e.g., we have 0.928 +/-0.006 for the ISNet, and only 0.549 +/-0.012 for the DenseNet121).

The two photographs in figure 1 were correctly classified by the 3 models. Both are labeled as containing the 3 considered attributes. In the facial attribute estimation task, we see that the common classifier pays some attention to body features outside of the face, and to the remaining background. However, most of the model’s focus is on the face. This observation, and the weak impact of segmentation on the task’s performances scores (see Table 3), may indicate that background bias does not produce a strong effect on model performance in facial attribute detection using the CelebA database. The two photographs in figure 1 exemplify the wide diversity of segmentation mask sizes in the CelebA dataset. We observe that, classifying both the small and the large face, the ISNet precisely kept the model focus inside the region of interest.

Comparing the two models with segmentation, the attention of the ISNet seems to be more concentrated, while the alternative model appears to have its relevances more distributed over the lungs or faces. Mostly in COVID-19 detection, there is more relevance alongside the lungs’ borders in the alternative segmentation model heatmaps, possibly indicating a source of overfitting. The model may have learned to consider the shape of the lungs’ borders/segmentation masks when distinguishing between the classes in the training dataset.

Figure 2 shows heatmaps for the artificially biased test datasets, considering the ISNet and the DenseNet121 (without segmentation), both trained with artificially biased databases. Again, red colors indicate positive relevance for COVID-19 or rosy cheeks. The images were correctly classified. A triangle that indicates the COVID-19 class is in the X-ray top left corner. The photograph has positive labels for the three attributes. Thus, it has three geometrical shapes, a square indicates rosy cheeks, a triangle indicates smiling, while the circle represents high cheekbones. For the DNN without segmentation, the triangle and square are clearly associated with COVID-19 and rosy cheeks, respectively. They are very pronounced in the heatmaps, evidencing that they strongly influenced classification. This scenario explains the classifiers strong performances in the biased databases. In the photograph’s no segmentation heatmap, the square is the most visible shape because it indicates rosy cheeks, the class chosen for relevance propagation in this example. The ISNet shows no relevance in the geometrical shapes, indicating the success of its implicit segmentation in both applications. Accordingly, the artificial background bias did not affect the model’s classification results.

![Input Image](image1.png) ![ISNet Heatmap](image2.png) ![No Segmentation Heatmap](image3.png)

Figure 2. Heatmaps for positive COVID-19 X-ray and photograph, both images extracted from the artificially biased test datasets. The geometrical shapes indicate the classes. Red colors indicate association to the class COVID-19 (X-ray) or rosy cheeks (photograph)
3 Discussion

It is common to think about DNNs as black boxes. This is because it is challenging to interpret powerful models with millions of trainable parameters. Not truly understanding how a classifier makes a decision is a matter of concern, particularly when dealing with crucial choices, as in the field of AI-assisted diagnosis, or in applications in which security is essential. Layer-wise Relevance Propagation shows us how each region inside an image affects the classifier choice, producing a relevance heatmap. The ISNet shows us how each region inside an image affects the classifier choice, producing a relevance heatmap.

The ISNet architecture is flexible and easily applicable to any type of classification neural network, though which Layer-wise Relevance Propagation is possible. A sequence of two DNNs (segmenter and classifier) is commonly used to solve tasks that can benefit from image segmentation as a preprocessing stage before classification. The ISNet introduces implicit segmentation, eliminating the need for a dedicated DNN segmenter, and without adding any new parameters or computational cost to the classifier at run-time. By replacing a standard pipeline (a U-net followed by a DenseNet121) with an ISNet based on the same DenseNet121 classifier, we obtain a model that is about 70% to 108% faster at run-time, and has almost 80% less parameters. This benefit would be even more pronounced for smaller classification neural networks.

The limitation of the ISNet is its training time. For the DenseNet121-based model, the training time was 82% longer than the standard segmentation-classification pipeline’s. The proposed model creates heatmaps for different classes in parallel, but the length of its training time still increases with the number of classes in the classification problem. More efficient implementations may reduce this limitation. However, in its current formulation, the ISNet exchanges training time for run-time performance. But this trade-off may be very profitable, given that DNNs can be trained in powerful computers, then later deployed in less expensive or portable devices.

Our results show that the ISNet’s superior run-time performance does not come at the price of accuracy. Indeed, the DenseNet121-based ISNet matched the DenseNet121 preceded by a U-net, and a standalone DenseNet121 in facial attribute estimation. In COVID-19 detection, it surpassed both alternative DNNs. The most popular datasets used to classify COVID-19 are mixed, with different classes coming from dissimilar sources. Therefore, lung segmentation can prevent the classifier from learning bias in the image background, which is a danger that has been demonstrated by other studies. In this work, we trained DNNs to classify chest X-rays as COVID-19, non-COVID-19 pneumonia, or healthy. Being trained on a mixed database and evaluated on an external dataset (with distinct image sources, allowing a better measurement of DNN generalization capacity), the ISNet achieved 0.798 +/- 0.007 mean balanced accuracy and 0.718 +/- 0.008 macro-average F1-Score, while the standard segmentation methodology achieved 0.7 +/- 0.009 and 0.645 +/- 0.009, and the model without segmentation reached 0.594 +/- 0.01 and 0.546 +/- 0.01. The ISNet results for the task of COVID-19 detection are promising, and in accordance with other studies that employed lung segmentation and evaluated their DNNs in external databases, a technique that is required to reduce the effects of background bias and shortcut learning in the reported performances. However, clinical tests are needed to ensure adequate real-world performance. In both tasks, the ISNet’s implicit segmentation ability allowed it to ignore even purposefully added bias in the image background, and its heatmaps reveal a precise segmentation of relevance, according to the lungs or the faces. Thus, we observe that the ISNet was successful when analyzing biomedical and natural images, the last containing both background clutter and a wide variety of segmentation target configurations. The accurate implicit segmentation achieved in such diverse applications calls attention to the general applicability of the new architecture.

The use of a LRP-based architecture during training naturally creates a DNN that combines a contracting (classifier) and an expanding (LRP Block) path, and links them via multiple skip connections. This structure is capable of combining context information, which is extracted by the classifier’s later layers, with the high resolution found in its earlier feature maps. Therefore, the ISNet can use segmentation masks for learning how to precisely control the classifier focus. The resulting run-time model efficiently uses the potential and flexibility of a deep neural network. The proposed architecture is a way to reduce bias and to increase confidence in the results achieved by a DNN, without any additional computational cost at run-time. Therefore, we believe that the ISNet is useful for the task of COVID-19 detection, for other mixed dataset scenarios, and for improving security in facial attribute estimation. More generally, we believe that the ISNet can also be advantageous for other problems that would benefit from image segmentation prior to classification.

4 Methods

4.1 Layer-wise Relevance Propagation

Deep neural networks are not easy to interpret. This is because they are complex and nonlinear structures with millions of trainable parameters. Layer-wise relevance propagation (LRP) is an explanation technique tailored for deep models. It was created to solve the challenging task of providing heatmaps to interpret DNNs. Heatmaps are graphics that show how a neural network distributes its attention in the input space. For each class, a heatmap explains if an image region has a positive or negative, strong or weak influence (relevance) on the classifier confidence for that class.

LRP is based on the almost conservative propagation of a quantity called relevance through the DNN layers, starting from one of the network output neurons and ending at the input layer, where the heatmap is produced. The meaning of the relevance in the heatmap is determined by the choice of the output neuron where the propagation starts. Positive values indicate that an input image pixel was associated with the class predicted by the chosen output neuron, while negative values indicate areas that reduce the classifier confidence in the class (e.g., regions that the classifier related to other classes in a multi-class, single-label problem). Furthermore, high (absolute) values of relevance show input features that were important for the classifier decision, i.e. the heatmap makes the classifier’s attention explicit.

Previous studies used LRP to understand which X-ray features were important for a DNN classifying COVID-19 patients, pneumonia patients, and healthy people. One study used lung segmentation as a preprocessing step and found a strong correlation between lung areas with high LRP relevance for the COVID-19 class and regions where radiologists identified...
severe COVID-19 symptoms. Both studies used training datasets from mixed sources and observed a concerning quantity of relevance outside of the lungs when segmentation was not used. In particular, LRP revealed that words exclusively present in the COVID-19 class strongly attracted the classifier’s attention, which associated them with COVID-19. Figure 3 is one example, containing the word “SEDUTO” and the letters “DX” in its upper corners. In the corresponding heatmap, the words’ red coloration indicates their high relevance for the COVID-19 class.

**Figure 3.** COVID-19 positive chest X-ray and respective LRP heatmap, where red regions show positive relevance for the COVID-19 class and blue areas show negative relevance (image parts associated with other classes, i.e. normal or pneumonia). Figure extracted from

LRP starts from the output of the neural network. After choosing one output neuron (according to its predicted class), we can define its relevance as equal to its output value (prior to nonlinear activation), and set the relevance of all other last-layer neurons to zero. Then, LRP uses different rules to propagate the relevance through each DNN layer, one at a time, until it reaches the model’s input. The choice of the set of rules influences the heatmap’s interpretability and the stability of the relevance propagation. Here, we will briefly explain the rules and procedures that were important for the creation of the ISNet.

The most basic rule is called LRP-0. We define the k-th output of a fully-connected layer \( z_k \) before activation as:

\[
z_k = b + \sum_j w_{jk} a_j
\]

Where \( w_{jk} \) represents the layer’s weight that connects its input \( j \) to output \( k \), \( b \) the layer’s bias parameter, and \( a_j \) the value of its input \( j \). LRP-0 propagates the relevance from the layer output, \( R_k \), to its input, \( R_j \), according to the following equation:

\[
R_j = \sum_k w_{jk} a_j z_k R_k
\]

Analyzing the fraction above, we see that LRP-0 redistributes the relevance from the layer k-th output \( (R_k) \) to its inputs \( (j) \) according to how much they contributed to the layer’s k-th activation \( (z_k) \). A second rule, LRP-\( \epsilon \), changes LRP-0 to improve the relevance propagation stability, adding a small constant, \( \epsilon \), to \( z_k \). Being \( \text{sign}(\cdot) \), a function that evaluates to 1 for positive or zero arguments, and to -1 otherwise, LRP-\( \epsilon \) is defined as:

\[
R_j = \sum_k \frac{w_{jk} a_j}{z_k + \text{sign}(z_k) \epsilon} R_k
\]

LRP is a scalable technique that can be efficiently implemented. A four-step procedure exists to rapidly execute the aforementioned LRP rules in a fully connected or convolutional layer with ReLU activation function (see section 4.4). It is possible to adopt a different rule for the DNN input layer, taking into account the nature of the input space. For images, a rule called \( Z^B \) considers the maximum and minimum pixel values allowed in the figures. Average and sum pooling, as linear operations, can be treated similarly to a convolutional layer. We can either handle max pooling as sum pooling, or adopt a winner-takes-all strategy, propagating all the layer output relevance to its inputs that were actually selected by the max pooling operation. Finally, batch normalization layers can be merged with neighboring convolutional or fully connected layers, defining a single equivalent layer, through which we propagate the relevance.

With defined rules for the most common DNN layers, LRP configures a very flexible technique, which is scalable and applicable to virtually any neural network architecture. For example, for the Keras Python library, implementations for many popular deep learning models are available at.
4.2 ISNet: Implicit Segmentation Neural Network

While models that use segmentation as a preprocessing stage before classification generally use a long processing pipeline with two DNNs (a segmenter and a classifier), the ISNet is a fundamentally different approach. We propose a technique to teach virtually any deep classifier to consider only the relevant image regions during classification. In other words, an ISNet produces a classifier capable of implicitly performing image segmentation. Implicitly, in this case, means that the ISNet will not produce a segmentation output (mask); however, it shall classify an unsegmented image as if it were previously segmented.

The main goals of ISNet are to simplify the processing pipeline of segmentation followed by classification, and to be flexible, permitting the use of any classification neural network. Therefore, at run-time, the model does not add any additional computational cost relative to a single standard classifier. Moreover, if a classification architecture is chosen as the ISNet base, it will exactly define the run-time model. We only alter the original architecture during the training stage.

Layer-wise Relevance Propagation was created to improve DNN interpretability\(^1\). With the ISNet, we introduce an additional function for it by using LRP to explicitly control a DNN’s attention. LRP heatmaps show the relevance of each input feature (e.g., pixels) for the DNN classification. During training, we therefore created a method to penalize undesired attention in the maps, forcing the classifier to minimize it and thus to consider only the relevant image regions. Using segmentation masks to define the important areas, the ISNet segments LRP relevance, minimizing the focus on the image background and thus performing implicit image segmentation. To minimize undesired LRP relevance (both positive and negative) during training, we propose a multitask loss function, comprising two terms: the traditional classification loss (which compares the supervised outputs in relation to labels, and a term that quantifies the amount of unwanted attention present in the heatmaps, created by the classifier’s heatmaps with segmentation targets. We precisely define the ISNet loss function in Section 4.3. During training, if a gold standard of segmentation targets is not available for the entire dataset, a pretrained segmenter (e.g., U-Net) can create the segmentation labels.

The training-time ISNet is defined by two modules: a classifier (e.g., DenseNet\(^3\), ResNet\(^27\), VGG\(^28\)) and an LRP block. The LRP block is a DNN created as a mirror image of the classifier. Its layers perform LRP propagation rules, using shared weights and skip connections with the equivalent classifier layers to propagate relevance through them. Thus, the LRP block produces LRP heatmaps. The advantage of defining the LRP relevance propagation process itself as a neural network is that the optimizers in standard deep learning libraries can automatically backpropagate the heatmap loss through the LRP block. Therefore, we avoid the task of manually defining LRP backpropagation rules, which could require changes in the standard optimization algorithms, making the ISNet much more complicated to implement and less practical. All parameters in the LRP block are shared with the classifier. Therefore, it generates the same heatmaps as those created with traditional LRP. Moreover, due to parameter sharing, after backpropagation of the heatmap loss through the LRP block, the optimization of the block’s parameters is automatically reflected in the classifier. Section 4.4 defines all the types of LRP block layers that we created (alternatively, LRP layers), which are enough to perform LRP in a modern, efficient, and complex classifier: a Densely Connected Convolutional Network\(^3\).

For ISNets to work properly, we need to produce one heatmap for each possible class, starting the LRP relevance propagation at the output neuron that classifies it. We cannot minimize unwanted LRP relevance for a single class (e.g., the winning one). Imagine that we have a bias in the image background, associated with class C, and we minimize only the unwanted LRP relevance for class C. In this case, the classifier can negatively associate all other classes with the bias, using it to lower their outputs, making the class C output neuron the winning one. This negative association is expressed as negative relevance in the other classes’ heatmaps. Consequently, the penalization of positive and negative unwanted relevance in all maps is a solution to the problem. Fortunately, using the LRP block to perform relevance propagation provides a simple and efficient strategy for creating multiple heatmaps. They are treated as different samples inside a mini-batch, being processed in parallel, and the relevance propagations for different classes do not interact with each other.

Figure 4 presents a simple ISNet example, whose classifier comprises two convolutional layers, L1 and L2, followed by a linear layer, L3. \(x_i\) indicates the input of classifier layer Li. With LRPi being the layer in the LRP block responsible for performing the relevance propagation through Li, its output, \(R_{i-1}\), is the relevance at the input of layer Li. \(y'\) is the classifier output for one of the classes (with all other outputs set to zero), which will be used to generate the heatmap associated with it. To perform the propagation, LPi shares parameters with Li.

After training the ISNet, using the classifier and LRP block and simultaneously minimizing the respective loss functions, the LRP block can be removed. The classifier will continue to automatically perform implicit segmentation during run-time. As an additional benefit, the LRP heatmaps may be used before removal for their original function, that is, to interpret the classifier’s decisions.

In this work, we used a DenseNet121 as the LRP classifier, because it is a modern and very deep model (121 layers) with strong potential in the field of lung disease classification\(^29\). Moreover, the efficiency of Dense Neural Networks makes them good candidates for becoming ISNet classifiers. Even though the DenseNet121 has more than 100 layers, it only uses around 8M trainable parameters, much fewer than a VGG with 16 layers (138M parameters) and even fewer than a U-Net (about 31M parameters). Due to its definition, an ISNet can be created with any classification neural network that is analyzable by LRP and that has enough power for the desired segmentation task. As LRP can be applied to virtually any DNN and already has implementations for most modern classifiers\(^26\) (even for recurrent networks), our approach has strong flexibility.

4.3 ISNet Loss Function

The minimization of unwanted LRP relevance is fundamental for the ISNet, so we begin by explaining the loss function that penalizes it. Two terms comprise the ISNet loss, \(L_{IS}\), a classification loss, \(L_C\), which penalizes the classifier’s classification outputs in relation to labels, and a term that quantifies the amount of unwanted attention present in the heatmaps, created by the LRP Block. We call this term ‘heatmap loss’, \(L_H\).

\[
L_{IS} = (1 - P)L_C + PL_H
\]  

(4)
Figure 4. Representation of an ISNet for a classifier containing 3 layers. The classifier is on the left, and the equivalent LRP block on the right.

The hyperparameter $P$ in the above equation balances the influence of the two losses in the gradient and parameters update. It must be valued between 0 and 1, with larger values increasing the strength of the heatmap loss. If $P$ is too small, the network will not minimize $L_H$ and implicit segmentation will not be performed. However, if $P$ is too high, the model may ignore the classification task and move towards a zero solution, minimizing its parameters to produce zero valued heatmaps, a simplistic manner of reducing $L_H$. To tune $P$, we advise a search through different values, followed by evaluation of validation error, as the ideal choice may change for different models and tasks. Nevertheless, fine adjustment of the parameter does not seem to strongly affect model performance. Although a long-term increase of $L_C$ indicates that $P$ is too high, we observed its increment during a small number of epochs in the beginning of the training process, followed by the two losses dropping simultaneously, even with adequate values of $P$.

$L_C$ is a standard loss function for classification. Here, we used cross-entropy or binary cross-entropy (for a single-label or multi-label task, respectively), averaged across the mini-batch samples. We will now explain the calculation of the second term, $L_H$. First, some definitions are needed: we denote as $H$ a tensor containing the heatmaps for all classes, created for a single input image or for a mini-batch of $B$ images (in this paper we use bold letters to indicate tensors). For a multiclass classification problem with $K$ classes and input images of shape $(C,Y,X)$, where $C$ is the number of channels, $Y$ the image height, and $X$ its width, $H$ assumes the following shape: $(B,K,C,Y,X)$. The second dimension (class) indicates at which output neuron the relevance propagation began. So in the $k$-th heatmap, the classifier associated positive relevance with the class $k$. The mini-batch and classes dimensions were merged inside the LRP block, but we separate them before loss calculation. $M$ is a tensor containing the segmentation targets (masks) for each figure in the mini-batch. A mask separates the image foreground and background. It is defined as a single-channel image with shape $(Y,X)$, valued 1 in the regions of interest and 0 in areas of undesired attention. Having $B$ masks, one for each mini-batch image, we just repeat them in the channels and classes dimensions for $M$ to match the shape of $H$.

Heatmap loss calculation starts by a normalization of the heatmaps tensor, dividing it by the square root of its elements’ variance plus a small $e$ value (e.g., $10^{-4}$). $e$ enforces the relationship $0/0 = 0$, avoiding indeterminate results. Afterwards, we substitute the tensor elements by their absolute values, $\text{abs}(\cdot)$, since the training procedure must minimize both the positive and the negative undesired relevance in the heatmaps:

$$H' = \text{abs}\left(\frac{H}{\sqrt{\text{Var}(H)} + e}\right)$$

The normalization’s objective is to account for relevance absorption. Although LRP performs an almost conservative propagation, relevance is absorbed by the stabilizer element in LRP-$\varepsilon$ and by the layers’ bias parameters. However, it is important to use a normalization procedure that does not shift the zero value in the heatmaps.

The next step in heatmap loss calculation is zeroing the relevance inside the regions of interest in the $H'$ heatmaps. This is because the $L_H$ minimization process should not affect the classifier’s ability to analyze these regions. With $I$ being a tensor whose elements are 1, with the same shape as $H$ and $M$, we create the inverted masks tensor, $I - M$, whose values are 1 in the region of undesired attention and 0 in the important image areas. Thus, an element-wise multiplication of $H'$ with the inverted
masks generates a new tensor, $UUUHHH'$, containing heatmaps that show only the undesired relevance:

$$UUUHHH' = (1 - M) \odot H' = (1 - M) \odot abs\left(\frac{H}{\sqrt{Var(H)} + e}\right) = [UH'_{bk,cyx}]$$ (6)

Summing the elements in $UUUHHH'$ for all image-related dimensions $(C,Y,X)$ reduces the 5-dimensional tensor to a 2-dimensional one, $uh'$. Therefore, the positive real value $uh'_{bk}$ is a measurement of the total undesired relevance in the heatmap of the mini-batch image $b$, starting the LRP relevance propagation at class $k$.

$$uh'_{bk} = \sum_{c=1}^{C} \sum_{y=1}^{Y} \sum_{x=1}^{X} UH'_{bk,cyx}$$ (7)

Cross-entropy is the most common error function for image segmentation. It therefore seemed a sensible choice for the $uh'_{bk}$ penalization. As the cross-entropy’s target for the undesired relevance, the natural choice is zero, which represents a model not considering background features during classification. Under this specific condition, the cross-entropy function, $CE(\cdot)$, for a scalar, $x$, can be expressed as:

$$CE(x) = -\log(1 - x)$$ (8)

Which evaluates to 0 when $x = 0$, and to infinity when $x = 1$. Consequently, we must choose a function, $f(\cdot)$, to map the $uh'_{bk}$ elements to the interval $[0, 1]$ before applying cross-entropy. Some design requirements for $f(\cdot)$, considering positive arguments ($uh'_{bk} \geq 0$), are: being monotonically increasing, differentiable, limited between 0 and 1, and to map 0 to 0. The sigmoid function, a common activation function for image segmentation, cannot be used, since it evaluates to 0.5 for a null input. After practical tests with some candidates, we decided to use the following function:

$$f(x) = \frac{x}{x + E}$$ (9)

Where $E$ is a hyperparameter controlling the function’s slope and how fast it saturates to 1. The ISNet does not appear to be very sensitive to this parameter. Instead of a computationally expensive grid search for the ideal $E$ and $P$ hyperparameters, we therefore suggest the following rule of thumb, which was effective for different tasks and ISNet classifiers during our tests: obtain the average $uh'_{bk}$ value for a few images at the start of the training process, and set $E$ for about this value divided by 10 to 100. This choice makes $f(uh'_{bk})$ high at the start of the training procedure (about 0.9 to 0.99), but not exaggerately saturated, facilitating the gradient descent. Figure 5a shows the behavior of $f(x)$ for $E = 1$, and the red dot marks the projection for a possible early $uh'_{bk}$ value, according to the proposed rule.

Finally, we can calculate the $g(uh'_{bk})$, the cross-entropy between $f(uh'_{bk})$ and 0, and average it for all $uh'_{bk}$ elements in $uh'$, resulting in the scalar $L_H$. Considering the B mini-batch images and K classes, we have:

Figure 5. Behavior of $f(x) = x/(x + E)$ (a) and $g(x) = -\log(1 - f(x))$ (b), with hyperparameter $E=1$. The red dot marks the position $x=10$.
$$g(uh'_{bk}) = CE(f(uh'_{bk})) = -\log(1 - \frac{uh'_{bk}}{uh_{bk} + E})$$  \hspace{1cm} (10)$$

$$L_H = \frac{1}{B \cdot K} \sum_{b=1}^{B} \sum_{k=1}^{K} g(uh_{bk}) = \frac{1}{B \cdot K} \sum_{b=1}^{B} \sum_{k=1}^{K} -\log(1 - \frac{uh'_{bk}}{uh_{bk} + E})$$  \hspace{1cm} (11)$$

Figure 5b plots $g(uh'_{bk})$, for $E = 1$ and positive $uh'_{bk}$ values, ensured by the absolute value operation in equation 5. As expected, the function monotonically increases with the undesired relevance in a heatmap, presenting a minimum at 0, when there is an absence of focus on the image background. As $g(uh'_{bk})$ diverges if $uh'_{bk} = 1$, we clamp $uh'_{bk}$ between 0 and $1 - 10^{-4}$ for numerical stability.

### 4.4 LRP Block Layers

Besides defining the LRP relevance propagation rules as neural network layers, we also need to modify their execution methodology to train an ISNet. For each layer in the classifier, a corresponding LRP layer is added to the LRP block to perform the relevance propagation through it. All LRP layers are based on an efficient LRP implementation in four steps. Considering the propagation of relevance through a classifier convolutional or dense layer $L$ with ReLU activation, the implementation is defined as:

1. Forward pass the layer $L$ input, $x_L$, through layer $L$, generating its output tensor $z$ (without activation).

2. For LRP-$\varepsilon$, modify each $z$ tensor element (z) by adding $sign(z)\varepsilon$ to it. Defining the layer $L$ output relevance as $R_L$, perform its element-wise division by $z$: $s = R_L/z$.

3. Backward pass: backpropagate the quantity $s$ through the layer $L$, generating the tensor $c$.

4. Obtain the relevance at the input of layer $L$ by performing an element-wise product between $c$ and the layer input values, $x_L$ (i.e. the output of layer $L-1$): $R_{L-1} = x_L \odot c$.

In the ISNet, we apply the following changes to the algorithm above:

In Step 1, we perform a forward pass through a copy of layer $L$, whose weights are shared with its clone (i.e. they have the same parameters). The input for the operation is layer $L$’s original input, $x_L$ (i.e. the output of layer $L-1$), captured by a skip connection between the LRP block and the classifier. Although the bias parameters are also shared, we do not directly optimize them for the $L_H$ minimization (in PyTorch we accomplish this with the detach function on the LRP layer shared bias). Knowing that the biases are responsible for relevance absorption during its propagation\(^1\), this choice prevents the training process from increasing them to enforce an overall reduction of relevance. As in the original method, we do not use an activation function in this step.

Step 2 is unchanged. We note that LRP-$0$ can be unstable for ISNet training. Therefore, we opted to base the LRP layers we implemented on LRP-$\varepsilon$, having chosen $\varepsilon$ as $10^{-2}$.

For Step 3, we propose a fundamental modification because using a backward pass during the forward propagation through the LRP block could cause conflicts in common deep learning libraries, complicating the subsequent backpropagation of the heatmap loss. Consequently, we substitute the backward pass in Step 3 with an equivalent operation: the forward propagation of $s$ through a transposed version of layer $L$. If $L$ is fully connected, its transposed counterpart is another linear layer, whose weights are the transpose of the original ones. Thus, Step 3 becomes: $c = W^T \cdot s$. Similarly, convolutional layers have transposed convolutional layers as their counterpart, using the same padding, stride, and kernel size. As in Step 1, the transposed layer share weights with layer $L$, but this time it does not use bias parameters.

Step 4 is unchanged and, as in Step 1, $x_L$ is carried by the skip connection. The obtained relevance value, $R_{L-1}$, is forwarded to the next LRP block layer, which will perform the relevance propagation for the classifier layer $L-1$.

Using italics to emphasize our changes to the original algorithm, we summarize the relevance propagation through the LRP layer corresponding to a convolutional or fully connected classifier layer $L$, which uses ReLU activation, as follows:

1. Forward pass the layer $L$ input, $x_L$, through a copy of layer $L$, generating its output tensor $z$ (without activation). Use parameter sharing, use the detach() function on the biases. Get $x_L$ via a skip connection with layer $L$.

2. For LRP-$\varepsilon$, modify each $z$ tensor element (z) by adding $sign(z)\varepsilon$ to it. Defining the layer $L$ output relevance as $R_L$, perform its element-wise division by $z$: $s = R_L/z$.

3. Forward pass the quantity $s$ through a transposed version of the layer $L$ (linear layer with transposed weights or transposed convolution), generating the tensor $c$. Use parameter sharing, but set biases to 0.

4. Obtain the relevance at the input of layer $L$ by performing an element-wise product between $c$ and the layer input values, $x_L$ (i.e. the output of layer $L-1$): $R_{L-1} = x_L \odot c$.

The above-defined convolutional and fully connected LRP layers are equivalent to the traditional LRP-$0$/LRP-$\varepsilon$ propagation rules. Therefore, the proposed execution methodology is not detrimental to the explanatory ability of LRP. In the following subsections, we explain the LRP layers for other common operations in DNNs, namely, pooling layers and batch normalization. Finally, we explain the implementation of the LRP $Z^B$ rule for the first convolutional or fully connected layer in the classifier.
4.4.1 LRP Layer for Pooling Layers

As linear operations, sum pooling and average pooling can be treated like convolutional layers, whose corresponding LRP layer was defined in section 4.4. However, since pooling operations do not have trainable parameters, no weight sharing is needed. Furthermore, the \( z \) tensor in Step 1 can be explicitly defined as the pooling output (obtained via a skip connection). We represent convolutional kernels with a tensor \( K \) of shape \((C1,C2,H,W)\), where \( C1 \) is the number of input channels, \( C2 \) of output ones, \( W \) the kernel width, and \( H \) its height. Then, for a convolution to be equivalent to pooling, \( C1 \) and \( C2 \) become the pooling number of channels, while \( W \) and \( H \) are defined by its kernel size. Naturally, the two layers use the same padding and stride parameters. To represent sum pooling, the convolutional kernel elements, \( k_{c1,c2,h,w} \), are defined as constants:

\[
k_{c1,c2,h,w} = \begin{cases} 1, & \text{if } c1 = c2 \\ 0, & \text{otherwise} \end{cases}
\]

And considering average pooling with kernel size of \((H,W)\), we have:

\[
k_{c1,c2,h,w} = \begin{cases} \frac{1}{H \times W}, & \text{if } c1 = c2 \\ 0, & \text{otherwise} \end{cases}
\]

For max pooling, the equivalent LRP layer adopts a winner-takes-all strategy, distributing all the relevance to the layer inputs that were chosen and propagated by the pooling operation. Therefore, we change the LRP four-step procedure for convolutional layers, substituting the forward pass of \( s \) through a transposed convolution (Step 3) with its propagation through a MaxUnpool layer. This operation, available in the PyTorch library, uses the indices of the maximal values propagated by max pooling to calculate its partial inverse, which sets all non-maximal inputs to zero. As in average/sum pooling, no parameter sharing is needed, and \( z \) in Step 1 can be directly obtained via a skip connection.

4.4.2 LRP Layer for Batch Normalization

Batch normalization (BN) layers are linear operations that can be fused with an adjacent convolutional or linear layer during relevance propagation, creating a single equivalent convolutional/fully connected layer. Thus, we calculate the parameters of the equivalent layer and create its corresponding LRP layer with the methodology presented in section 4.4.

One study\(^{25}\) presented equations to fuse batch normalization with an adjacent convolutional or linear layer. To analyze a convolution, followed by BN, and then by a ReLU activation function (a configuration used in DenseNets), we define: \( K \) of shape \((C1,C2,H,W)\) as the convolution weights/kernels; \( B \) as the convolutional bias, of shape \((C2)\); \( \gamma \) as the batch normalization weights, of shape \((C2)\); \( \beta \) as the BN bias, also of size \((C2)\); \( \mu \) the per-channel \((C2)\) mean of the batch-normalization inputs, and \( \sigma \) its standard deviation, defined as the square root of the input’s variance plus a small value for numerical stability (a parameter of the BN layer). After replicating the tensors \( \gamma \) and \( \sigma \) in the \( C1 \), \( W \), and \( H \) dimensions to match the shape of \( K \), we can define the equivalent convolutional layer weights, \( K' \), with element-wise divisions and multiplications:

\[
K' = \frac{K \circ \gamma}{\sigma}
\]

While the equivalent convolution bias, \( B' \), is given by the following element-wise operations:

\[
B' = \beta + \gamma \odot \frac{B - \mu}{\sigma}
\]

Dense neural networks also present BN layers between pooling and ReLU layers, thus an adjacent convolutional or linear layer is not available. Observing that the preceding pooling operation is also a linear operation (not followed by any activation function), we can fuse it with batch normalization. If we have average or sum pooling, we simply calculate its equivalent convolution, as explained in Section 4.4.1, and perform the fusion according to equations 14 and 15.

In the case of max pooling, we start by imagining a convolutional layer performing identity mapping before the batch normalization. This imaginary layer has no bias or padding, its stride is unitary, and its weights, \( K' \), have shape \((C,C,1,1)\), where \( C \) is the max pooling number of channels. An element \( k_{c1,c2,h,w} \) is then given by equation 12. Thus, we can fuse the batch normalization with the identity convolution according to the equations 14 and 15, creating a BN equivalent convolution. The LRP layer for relevance propagation through the sequence of MaxPool, BN, and ReLU is then defined by the 4-step procedure below:

1. Forward pass the MaxPool output through the BN equivalent convolution, generating its output tensor \( z \) (without activation). Use parameter sharing, use the detach() function on the biases, and obtain the MaxPool output via a skip connection.
2. For LRP-\( \varepsilon \), modify each \( z \) tensor element \((z)\) by adding \( \text{sign}(z)\varepsilon \) to it. Define the output relevance of the ReLU activation as \( R_z \), perform its element-wise division by \( z \): \( s = R_z/z \).
3. Forward pass the quantity \( s \) through a transposed version of the BN equivalent convolution (transposed convolution), generating a quantity \( t \). Employ parameter sharing, but set biases to 0. Then perform a MaxUnpool operation, using \( t \) as input, creating the tensor \( c \).
4. Obtain the relevance at the input of the MaxPooling layer by performing an element-wise product between $c$ and the MaxPool input values, $x_L$ (i.e. the output of layer L-1): $R_{L-1} = x_L \odot c$.

This methodology generates the same result as propagating the relevance first in the BN equivalent convolution and then in the max pooling LRP layer, but it has a lower computational cost.

4.4.3 LRP Layer for $Z^b$ LRP Rule

The LRP $Z^b$ rule considers the DNN input range in its formulation, and is used for the first layer in the neural network. For a convolutional or fully connected layer, it begins with a separation of the positive and negative elements of its weights, $W$, and bias, $B$. With $0$ being a tensor of zeros:

$$W^+ = \max(0, W)$$  \hspace{1cm} (16)

$$B^+ = \max(0, B)$$  \hspace{1cm} (17)

$$W^- = \min(0, W)$$  \hspace{1cm} (18)

$$B^- = \min(0, B)$$  \hspace{1cm} (19)

$$B^-$$  \hspace{1cm} (20)

The parameters define three new layers: weights $lW^+$ and biases $lB^+$ define layer $L^+$, where $l$ is the lowest pixel value possible (in the common case where $l = 0$, $L^+$ is ignored); weights $hW^-$ and biases $hB^-$ produce layer $L^-$, where $h$ is the highest allowed input. The original parameters, $W$ and $B$ define a copy of the original layer $L$. The four-step procedure for convolutional/fully connected layers (defined in Section 4.4) is then changed to:

1. Forward pass the L layer input, $x_L$, once for each of the tree layers ($L^+, L^-$, and a copy of L), producing $\mathbf{z}^+, \mathbf{z}^-$ and $\mathbf{z}^{\text{original}}$ (without activation). The classifier layer $L$ shares parameters with the three instances. The bias values are not optimized to reduce the heatmap loss. Use a skip connection with layer $L$ to obtain $x_L$. Combine the three results in the following manner: $\mathbf{z} = \mathbf{z}^{\text{original}} - \mathbf{z}^+ - \mathbf{z}^-$. 

2. Modify each $\mathbf{z}$ tensor element by adding $\text{sign}(\mathbf{z})\epsilon$ to it. Defining the layer $L$ output relevance as $R_L$, perform its element-wise division by $\mathbf{z}$: $s = \frac{R_L}{\mathbf{z}}$.

3. Forward pass the quantity $s$ through a transposed version of $L$, $L^+$ and $L^-$, creating the tensors $c^{\text{original}}$, $c^+$ and $c^-$, respectively. Use parameter sharing, but set biases to 0.

4. Being $x_L$ the layer L input values, obtain the relevance at the input of layer L, $R_{L-1}$, with: $R_{L-1} = x_L \odot c^{\text{original}} - c^+ - c^-$.

If batch normalization follows the first classifier layer, placed before the ReLU nonlinear activation, the above procedure can implement the $Z^b$ rule for the equivalent convolutional/fully connected layer (the result of fusing the convolutional/linear layer with BN), explained in Section 4.4.2.

4.4.4 Dropout

The dropout operation can be defined as the random removal of layer input elements during the training procedure, and LRP block layers can automatically deal with it. For a layer $L$ preceded by dropout, its inputs $x_L$ shall have some zero values caused by the operation. According to the element-wise multiplication $R_{L-1} = x_L \odot c$ (see Step 4 of the procedure in Section 4.4), these null values will also make their respective relevances zero, thus being accounted for in the LRP propagation.

4.5 Explaining the Implicit Segmentation Precision

Having defined the LRP layers of the LRP block, we can visualize the entire ISNet structure. For a common feed-forward classifier (without skip connections in its structure), a layer in the middle of the corresponding LRP block, propagating relevance through the classifier layer $L$, has 3 connections: one with the previous LRP layer, the second with the following LRP layer, and a skip connection with the classifier layer $L$. The first carries the relevance that can be seen at the layer $L$ output, $R_L$; the second brings the LRP layer result, $R_{L-1}$ (equivalent to the relevance at the layer $L$ input), to the next layer in the LRP block; and the last carries, from the classifier layer $L$, the information required for relevance propagation e.g. L’s input, $x_L$, if it is convolutional, or output, $x_{L-1}$, in the case of pooling. The Figure below exemplifies the ISNet architecture for a simple classifier, comprising the following sequence of layers: $L1 \rightarrow L2 \rightarrow L3 \rightarrow L4 \rightarrow L5$, where $L1$ and $L3$ are defined by $\text{Convolution} \rightarrow \text{ReLU}$, while $L2$ and $L4$ are max pooling operations, and $L5$ is fully connected. The LRP layer propagating relevance through layer $Li$ is named $LRPi$. In Figure 6, $y'$ is the classifier output $y$ for one class, with all other outputs made zero. One map shall be created for each class, with each LRP propagation being executed in parallel, in a strategy analogous to mini-batch processing.

As expected, the ISNet structure has a contracting path, defined by the classifier, and an expanding path, the LRP block. We see that the LRP layers for pooling (in red) increase the size of their relevance inputs. Afterwards, an LRP correspondent for convolution (in green) reduces the number of channels in the relevance signal, and combines it with information from an earlier feature map containing higher resolution, which is brought from the classifier. As a consequence of the LRP rules, we naturally have a structure that combines context information from later classifier feature maps with high resolution from the earlier maps, which suffered less down-sampling by pooling operations. Thus, skip connections between an expanding and a contracting path
Figure 6. Representation of an ISNet. The classifier is on the left, and the equivalent LRP block on the right. Pooling layers and their LRP counterparts are in red, convolutional in green, and linear in yellow.

are used to combine information from later and earlier feature maps; this idea is behind state-of-the-art architectures used for image segmentation and object detection, such as the U-Net, Filter Pyramid Networks, and the YOLOv3. Therefore, the concept should also allow the ISNet’s implicit image segmentation to be precise.

4.6 Challenges and Implementation Notes
Fundamentally, the ISNet architecture restrains the classifier during training, conditioning it to analyze only the relevant part of the image. This is the case even when there are undesired background features that could allow the model to reduce the classification loss rapidly and easily. Therefore, any change in the ISNet architecture needs to be cautiously implemented, because it can create a new and unintended way for the classifier to minimize the heatmap loss. Namely, as a flexible model, the classifier will look for the easiest way to reduce $L_{LRP}$, which may be by “cheating” and finding a strategy to “hide” the undesired relevance in the heatmaps.

We encountered an interesting example of this behavior when testing a preliminary version of the ISNet architecture. If we use a heatmap loss formulation that penalizes a ratio of the undesired relevance to the relevance inside the region of interest, the classifier may minimize $L_{LRP}$ by artificially boosting the desired relevance. The alternative loss was still based on the absolute value of the heatmaps (to minimize positive and negative undesired relevance), so the model created a fine and regular chessboard pattern inside the zones of interest, alternating between strong positive and strong negative relevance. The artificial pattern allows the region of interest to equally affect each class in the classifier, making it meaningless for the classification task. However, the high absolute values in the pattern strongly reduce the ratio in the alternative heatmap loss, allowing its minimization while the classifier focuses on the image background with a comparatively small quantity of absolute relevance.

Regarding implementation, we need to be careful with in-place operations inside the classifier, since they may change values that will later be processed by the LRP block. Moreover, we explicitly chose to use deterministic operations during training in PyTorch. This was to ensure better correspondence between the LRP block and the classifier (which can be accomplished with a single command, use_deterministic_algorithms(True)). Finally, using PyTorch’s forward hooks, we can easily store all relevant variables during the forward propagation through the classifier, allowing their later access by the LRP block. This is an effortless way to create the skip connections between the classifier and the block, even if the classifier is already defined and instantiated. To improve training stability, gradient clipping may be utilized, and too high learning rates should be avoided.
4.7 DenseNet based ISNet and Classifier Skip Connections

The Densely Connected Convolutional Network (DenseNet) is characterized by the dense block, a structure where each layer receives, as input, the concatenation of the feature maps produced by every previous layer in the block (i.e. the block presents skip connections between each layer and all preceding ones). For LRP relevance propagation, all the connections must be considered. Therefore, a mirror image, inside the ISNet LRP block, of a dense block with S skip connections will also have S internal skip connections, now propagating relevance. Naturally, we also have skip connections between the classifier and the LRP block, which carry information from layer L-1, but rather as the concatenated outputs of all layers preceding L in its dense block.

To understand the relevance skip connections, imagine that, in the classifier, a layer L propagates its output to layers L+i, with \(i \in \{1, ..., N\}\). We define \(R_{\text{inp}(L+i), L}\) as the relevance at the input of layer L+i, considering only its input elements (or channels) connected to L. We define relevance at the input of layer L+i, considering all input channels, as \(R_{\text{inp}(L+i)}\). \(R_{\text{inp}(L+i), L}\) can be obtained with relevance propagation through L+i. Then, to further propagate the relevance through layer L, we set the relevance at L’s output as \(R_L\), given by equation 21. Proceeding with the propagation rules explained in section 4.4, we can obtain \(R_{\text{inp}(L)}\), the relevance at the input of layer L.

\[
R_L = \sum_{i=1}^{N} R_{\text{inp}(L+i), L}
\] (21)

In a DenseNet, a layer L inside one of its dense blocks is defined as a nonlinear mapping \(H_L(\cdot)\), which in turn comprise a standard feed-forward sequence of other layers, e.g. ReLU activation, convolution, and batch normalization. Since there is an absence of skip connections inside one layer L, we propagate relevance through the sequence in the standard manner. Equation 21 is useful if \(H_L(\cdot)\) is defined as sequences in the form \(C_L(\text{convolution}) \rightarrow BN_L \rightarrow \text{ReLU}_L, C_L \rightarrow \text{ReLU}_L, C_L \rightarrow \text{Pool}_L \rightarrow \text{ReLU}_L\), or combinations of the previous sequences. Figure 7 presents a simple example, considering a sequence of 4 convolutional layers in the classifier, L0, L1, L2, and L3, each defined as \(C \rightarrow \text{ReLU}\) and receiving the outputs of all previous layers. The flow of relevance is observable in the LRP block. Note that different connections carry different channels of the \(R_{\text{inp}(L+i)}\) tensor, \(R_{\text{inp}(L+i), L}\). In the Figure, we define the input of layer Li as \(x_i\), which is a concatenation (in the channels dimension) of the previous layers’ outputs, \(y_j\).

**Figure 7.** Classifier and LRP block for 4 convolutional layers with skip connections in the classifier

The most common proposal for \(H_L(\cdot)\) in DenseNets is: \(BN_L \rightarrow \text{ReLU}_L \rightarrow C_L \rightarrow BN_L \rightarrow \text{ReLU}_L \rightarrow C_L\). In Section 4.4, we defined LRP layers to propagate relevance in sequences ending with ReLU activation. As such, they cannot be directly applied to \(H_L(\cdot)\), and nor can we rely on equation 21. Consider a dense block layer L, followed by N other layers, L+i (\(i \in \{1, ..., N\}\),
which receive L’s output, \( R_{ReLU}^{L+1} \) is the relevance at the output of the first ReLU inside layer \( L+i \) (ReLU\(^+_{L+i} \)), considering only channels that came from layer \( L \). Layer \( L+i \) starts by processing layer \( L \)’s output through the sequence \( BN^{L+1}_{L+i} \rightarrow \) ReLU\(^{L+1}_1 \), producing a tensor that we shall call \( y_{L+i}^{L+1} \). We can define \( y_L \) as the concatenation, in the channels dimension, of the \( N \) \( y_{L+i}^{L+1} \) feature maps, one for each \( L+i \) layer. With these definitions, the below procedure calculates the relevance at the output of layer \( L \)’s first ReLU activation, \( R_{ReLU}^L \), based on the N relevances \( R_{ReLU}^{L+i,L} \):

1. Fuse layer \( L \)’s second convolution, \( C_{L}^{2} \), with the first batch normalization operation in each of the \( N \) layers \( L+i \), \( BN_{L+i}^{1} \), obtaining equivalent convolutional kernels and biases according to equations \( 14 \) and \( 15 \), respectively. Concatenate these parameters in their output channel dimension, creating the kernels and biases that define a single equivalent convolutional layer, which generates \( y_{L} \) from layer \( L \)’s second ReLU output. Use it to recreate \( y_{L} \), but detach the bias parameters.

2. Concatenate the N relevances \( R_{ReLU}^{L+i,1} \) (for each \( L+i \) layer) in the channel dimension, producing \( R_{same}^{L} \). This tensor can be seen as the relevance at the output of the equivalent convolution created in step 1.

3. Propagate the relevance \( R_{same}^{L} \) through equivalent layer from step 1, using the usual four-step procedure for convolutional layers (section 4.4). The operation returns \( \bar{R}_{ReLU}^{L+2} \), the relevance at the output of the second ReLU activation in layer \( L \).

4. Obtain \( R_{ReLU}^{L+1} \) by propagating \( \bar{R}_{ReLU}^{L+2} \) through layer \( L \)’s sequence of operations \( C_{L}^{1} \rightarrow BN^{2}_{L} \rightarrow \) ReLU\(^{2}_{L} \), using the procedure in Section 4.4.2. \( R_{ReLU}^{L+1} \) can be propagated through earlier layers in the dense block by repeating these four steps.

A DenseNet transition layer \( L \) can be formed by the following sequence: \( BN^{1}_{L} \rightarrow \) ReLU\(^{1}_{L} \rightarrow C_{L}^{1} \rightarrow \) ReLU\(^{2}_{L} \rightarrow \) AvgPool\(_{L} \). ReLU\(^{1}_{L} \) is not part of its original configuration, but we added it to simplify the relevance propagation, as our rules are defined for layers with ReLU activation. It did not seem to have a detrimental effect on the model. The transition layer sits between 2 dense blocks. It receives all outputs from the layers in the first block, \( B \), and propagates its own result to every layer of the next one, \( B+1 \). Therefore, a layer in block \( B \) naturally considers the transition layer among its consecutive \( N \) layers during Step 1 of the above procedure; the same is true for the \( BN \rightarrow \) ReLU sequence following the last dense block in a DenseNet (every layer in the last layer shall consider it in Step 1). The relevance propagation through the transition layer \( L \) must take into account all its skip connections with the block \( B+1 \), so we use a 4-step procedure similar to the one above. There are only two changes. Since \( L \) ends in an average pooling instead of a convolution, we modify the fusion process in Step 1 to merge pooling and BN, a technique explained in Section 4.4.2. Furthermore, in Step 4, we obtain \( R_{ReLU}^{L+1} \) after propagating \( R_{ReLU}^{L+2} \) through the transition layer \( C_{L}^{1} \rightarrow \) ReLU\(^{2}_{L} \), using the rules defined in Section 4.4. We can treat the max pooling layer in the beginning of the DenseNet similarly, considering its skip connections with the first dense block. The code to automatically generate an ISNet from a DenseNet, in PyTorch, is available at https://github.com/PedroRASB/ISNet.

In this study, we use an ISNet created with a DenseNet121 classifier. Our main reason for this choice is the DenseNet’s small number of trainable parameters in relation to its depth (fewer than 8M parameters and 121 layers). Moreover, the architecture was successfully employed for detecting COVID-19 and other lung diseases using chest X-rays\(^{32,29} \). We utilized the same DNN for facial attribute estimation, as the DenseNet is known for its robust performance when analyzing natural images\(^3\). We modified the original DenseNet121 classifier by substituting its last layer with a layer containing 3 neurons (representing 3 possible classes, normal, pneumonia and COVID-19, or rosy cheeks, high cheekbones and smiling), preceded by dropout of 50%.

### 4.8 Datasets

#### 4.8.1 COVID-19 Chest X-ray

In this study we employed the Brixia COVID-19 X-ray dataset\(^5 \) as the source of the training and hold-out validation COVID-19 positive samples. It is one of the largest open databases regarding the disease, providing 4644 frontal X-rays showing symptoms of COVID-19 (i.e., samples to which the dataset authors assigned a severity score, the Brixia Score, higher than 0). All images were collected from the same hospital, ASST Spedali Civili di Brescia, Brescia, Italy. The patients mean age is 62.4 years, with a standard deviation of 13.6 years. They are 69.8% male. We randomly assigned 75% of the samples (3483 images) for training, and the remaining for hold-out validation. The two subdivisions were not allowed to have images from the same patients.

The images of healthy and non-COVID-19 pneumonia patients in our training and hold-out validation datasets come from the CheXpert database\(^33 \). It is a large collection of chest X-rays, showing various lung diseases, assembled in the Stanford University Hospital, California, United States of America. The database’s classification labels were created with natural language processing, utilizing radiological reports, and have an estimated accuracy surpassing 90%\(^33 \). Considering class balance, we randomly gathered 4644 images of healthy patients, and 4644 of pneumonia patients. Samples were also randomly divided with 25% for hold-out validation and 75% for training, employing a patient split. Pneumonia patients have a mean age of 62.3 years, with a standard deviation of 18.7 years. They are 57.1% male. Healthy patients have a mean age of 51.7 years, with standard deviation of 18.2, and are 56.3% male.

To avoid the effect of mixed dataset bias in our reported results, and to better understand how segmentation improves the DNNs’ generalization capability, an external test database (with dissimilar sources in relation to the training dataset) was assigned. Regarding the COVID-19 class, we selected the dataset BIMCV COVID-19+\(^34 \) for evaluation. The database is also among the largest open sources of COVID-19 positive X-rays. The data was gathered from health departments in the Valencian healthcare system, Spain. Therefore, it is highly unlikely that this database shares patients with the Brixia dataset\(^5 \). It presents
1515 COVID-19 positive frontal X-rays. The corresponding patients have a mean age of 66 years, with standard deviation of 15.3 years. They are 59.6% male.

We also performed cross-dataset testing for the pneumonia and normal classes. We chose the ChestX-ray14 database as the source of test pneumonia X-rays. We utilized 1295 pneumonia images, corresponding to patients over 18 years old. They present a mean age of 48 years, with standard deviation of 15.5 years, and are 58.7% male. The images were gathered from the National Institutes of Health Clinical Center, Bethesda, United States of America. Normal test images were extracted from a database assembled in Montgomery County, Maryland, USA (80 images), and Shenzhen, China (336 images). The healthy patients have a mean age of 36.1 years (and standard deviation of 12.3 years) and are 61.9% male. The images in this database were manually labeled by radiologists, while the ChestX-ray14 dataset authors labeled their database with natural language processing, according to radiological reports (with an estimated accuracy over 90%). For this reason, we preferred to extract the normal images from the Montgomery and Shenzhen database, and not from ChestX-ray14.

The assembled database was chosen to resemble the characteristics of the most popular COVID-19 datasets: the use of dissimilar sources for different classes is a characteristic that increases the risk of dataset bias, making lung segmentation more important. The Brixia COVID-19 dataset presented only 11 images of patients younger than 20 years old, and the BIMCV COVID-19+ database has a single one. Thus, to avoid bias, patients under 18 were not included in the other classes, which had much higher proportions of pediatric patients in their original datasets.

Both the ISNet and the alternative segmentation-classification pipeline require segmentation masks. The ISNet utilizes them to calculate the heatmap loss during the training procedure, while the alternative models need them to train the segmenter. Thus, we utilized a U-Net, previously trained for lung segmentation, to create the segmentation targets. We applied a threshold of 0.4 on the U-Net output to produce binary masks, which are valued 1 in the lung regions and 0 everywhere else; the chosen threshold maximized the segmenter’s validation performance (intersection over union or IoU). The U-Net was trained using 1263 chest X-rays distributed across the classes COVID-19 (327 images), healthy (327 images), pneumonia (327 images), and tuberculosis (282 images). The model achieved an IoU of 0.864 with the ground truth lung masks during testing. Please refer to for a detailed explanation of the U-Net training procedure.

Finally, to create an extreme test for the ISNet’s segmentation performance, we artificially biased the described dataset: all COVID-19 images were marked with a triangle in their upper left corner, the normal class was marked with a square, and pneumonia with a circle. See figure 2 for an example. Our data augmentation procedure can sometimes remove the shapes from the training samples (with rotations and translations), which may attenuate the background bias effect. However, the same is true for many natural background features (e.g., text and markers close to the X-ray borders), and the effect of the artificial bias should still be very solid. A common classifier, analyzing unsegmented images, can easily learn to identify the shapes, and to use them for improving classification accuracy. Therefore, we train an ISNet on the biased dataset and evaluate it with test images containing the geometric shapes or not. If it performs equally in both cases, we can conclude that adequate segmentation was achieved.

4.8.2 Facial Attribute Estimation

For the task of facial attribute estimation we employed images from the Large-scale CelebFaces Attributes Dataset, or CelebA. The database has 10000 identities, each one presenting 20 images. Binary labels were created by a professional labeling company, according to 40 facial attributes. As the database comprises in the wild images, the displayed faces show a wide variety of positions and sizes, and background clutter is present.

The CelebAMask-HQ dataset presents a subset of 30000 high-quality images from CelebA, cropped and aligned. Furthermore, these images have manually-created segmentation masks, which indicate the face regions. For this study, we selected the CelebA images that originated CelebAMask-HQ. Afterwards, we created their segmentation masks by applying translation, rotation and resizing to the CelebAMask-HQ masks; with this operation we reverted the CelebAMask-HQ crop and align procedure (described in appendix C). We employ the dataset split suggested by the CelebAMask-HQ dataset authors, which assigns 24183 samples for training, 2993 for hold-out validation and 2824 for testing. The proposed splits are subsets of the official CelebA training, validation and test datasets.

We chose to work with unaligned images that have strong background clutter, and whose face segmentation masks have a wide variety of shapes and positions. We think this setting constitutes a challenging test of the ISNet implicit segmentation capability. Moreover, the CelebA dataset authors show that their DNN naturally focuses more on the persons’ faces when more facial attributes are classified. For this reason, we believe that the ISNet implicit segmentation task will be more difficult when it classifies a small number of attributes. Thus, we chose to work with 3 classes, to better assess the ISNet segmentation potential, and to better visualize the architectures’ benefits. If we choose attributes that also have features outside of the face (e.g., gender), the ISNet architecture will change the natural classification strategy for one that ignores features outside of the face. This effect can be desirable or not, depending on the researcher’s objective. Here we opted to classify three attributes that are exclusively present in the face: rosy cheeks, high cheekbones and smiling. The first two are considered identifying attributes, i.e., they can be used for user identification. Therefore, avoiding bias in their classification is a security concern.

As in COVID-19 detection, we also created an artificially biased dataset for facial attribute estimation. The second application constitutes a multi-label problem. Thus, we marked our images with a square in the lower-right corner, a circle in the lower-left, and a triangle in the upper-left, to indicate rosy cheeks, high cheekbones and smiling, respectively. See figure 2 for an example of an image with three positive labels.

4.9 Alternative Models: U-Net Followed by DenseNet121 and Standalone DenseNet121

In this study, we compare the ISNet with a more traditional methodology to segment and classify images: a U-net (segmenter) followed by a DenseNet121 (classifier). The Densely Connected Convolutional Network is configured exactly as the one inside the ISNet (including a 50% dropout before the output layer), while the U-Net uses its original architecture proposal,
which can be seen in Figure 1 of 16. We trained the segmenter beforehand, using the same datasets that would be latter used for classification, and employing the segmentation masks as targets. The same targets were used in the ISNet training. Analyzing the U-Net validation performance, we found the best threshold to binarize its outputs. For COVID-19 detection the optimal value was 0.4, and for facial attribute estimation, 0.5.

The U-Net architecture 16 and its variants are a common model choice for biomedical image segmentation, including lung segmentation 9,10,39. It was created with this type of task in mind and so is designed to obtain strong performances even in small datasets. The U-net is a fully convolutional deep neural network with two parts. First, a contracting path, created with convolutions and pooling layers, captures the image’s context information. Then, an expanding path uses transposed convolutions to enlarge the feature maps and perform precise localization. Skip connections between the paths allow the expanding path access to earlier high-resolution feature maps (before down-sampling), which carry accurate positional information. Its combination with the context information in the later feature maps results in the U-net’s ability to conduct precise image segmentation.

As is commonly done, the U-Net parameters are kept frozen when training the DenseNet121 classifier. The pipeline for segmentation followed by classification can be summarized as follows:

1. The U-Net processes the X-ray and segments the lung region.
2. We threshold the U-Net result and create a binary mask, where lungs are represented as 1 and the remaining image parts as 0.
3. We use an element-wise multiplication between the X-ray and the mask to erase the image background.
4. The DenseNet121 classifies the segmented image.

Finally, we also train the classifier (DenseNet121) without segmentation, as a baseline.

4.10 Data Processing and Augmentation
For COVID-19 detection, we loaded the X-rays as grayscale images to avoid the influence of color variations in the dataset. We then performed histogram equalization to further reduce dataset bias. Next, we re-scaled the pixel values between 0 and 1, reshaped the images (which can assume various sizes in the database) to 224x224 and repeated them in 3 channels. The shape of (3,224,224) is the most common input size for the DenseNet121, reducing computational costs while still being fine enough to allow accurate segmentation and lung disease detection 9,29,22. Single-channel images can be used but, without a profound change in the classifier architecture, they would provide marginal benefits in the training time. Test images were made square (by the addition of black borders) before resizing to avoid any bias related to aspect ratio. The technique was not used for training because the model without segmentation could learn to identify the added borders.

Since the DenseNet121 classifier is a very deep model and our dataset is small, we used data augmentation in the training dataset to avoid overfitting. The chosen procedure was: random translation (up to 28 pixels up/down, left/right), random rotations (between -40 and 40 degrees), and flipping (50% chance). Besides the overfitting prevention, the process also makes the DNN more resistant to natural occurrences of the operations. We used online augmentation and substituted the original samples by the augmented ones.

For facial attribute estimation, we loaded the photographs in RGB, re-scaled between 0 and 1, and reshaped the images to 224x224, which is a common input size for the DenseNet121 and for natural image classification. We employed the same online image augmentation procedure used in the COVID-19 detection task.

4.11 Training Procedure
To train the ISNet, we first set the heatmap loss $E$ hyperparameter to 10, according to the procedure described in Section 4.3. We then analyzed the hold-out validation accuracy to find the ideal $P$ parameter for the ISNet loss function. During this tuning process, we trained on the artificially biased datasets to ensure that $P$ was high enough to produce adequate segmentation even in extreme cases. We looked for a model resulting in equal accuracy in the original validation set and in its artificially biased version. The $P$ value that better balanced the two loss terms and provided the best results was 0.7 for both tasks, and we found little performance variation with similar values (e.g., 0.5, 0.6 and 0.8). During the hyperparameter tuning procedure, we noticed that using weight decay makes it much more complicated to find $P$. This is because it seems to strongly favor a zero solution (i.e. the network minimizing its parameters to generate null heatmaps, optimizing the heatmap loss and L2 penalization, but ignoring the classification task). Thus, we did not use L2 regularization.

We employed a similar training procedure for the 3 DNNs, to allow a fair comparison. It started from a random parameter initialization. The chosen optimizer was stochastic gradient descent, using momentum of 0.99, and a mini-batch of 10 images. We employed gradient clipping to limit gradient norm to 1, making the training procedure more stable. Learning rate was set to $10^{-3}$. For the COVID-19 detection task we utilized cross-entropy as the classification loss, and for facial attribute estimation we used binary cross-entropy, as it is a multi-label problem. The network later evaluated on the test dataset is the one achieving the best validation performance during training. We trained all DNNs for 48 epochs in COVID-19 detection. We used 144 epochs for the ISNet in facial attribute estimation, but in this task we trained the alternative methodologies for 96 epochs, because they were already showing overfitting at that point. The classification thresholds for facial attribute estimation were chosen to maximize validation mAP 1 in the trained DNNs.

For the alternative segmentation-classification pipeline, we began by training the U-net for segmentation. We employed the same dataset, data augmentation and preprocessing steps used in the ISNet training procedure. We used stochastic gradient descent, with mini-batches of 10 samples and momentum of 0.99. Learning rate was set to $10^{-4}$ and we utilized the cross-entropy loss function. We trained using hold-out validation, until overfitting could be observed. Accordingly, we employed
600 epochs in the lung segmentation task, and in face segmentation, 400. The model achieved a test IoU of 0.893 in lung segmentation, and 0.925 in face segmentation. In both cases, the resulting segmentation masks seemed adequate upon visual inspection.

**Data Availability**

In support of the findings of this study, the X-ray data from healthy and/or pneumonia positive subjects are available from the corresponding author of20 upon reasonable request; in ChestX-ray14, https://paperswithcode.com/dataset/chestxray14; and in CheXpert, https://stanfordmlgroup.github.io/competitions/chexpert/. COVID-19 radiography data that support the findings of this study are available in The BrixIA COVID-19 project, https://brixia.github.io/; and in the BIMCV-COVID19+ database, http://https://bimcv.cipf.es/bimcv-projects/bimcv-covid19/. Images for facial attribute estimation, along with their segmentation masks, are available in the Large-scale CelebFaces Attributes (CelebA) Dataset, https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html.

**Code Availability**

The code containing the ISNet PyTorch implementation is available at https://github.com/PedroRASB/ISNet.

**References**

1. Bach, S. et al. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE* **10**, 1–46, DOI: 10.1371/journal.pone.0130140 (2015).
2. Geirhos, R. et al. Shortcut learning in deep neural networks. *Nat. Mach. Intell.* **2**, 665–673, DOI: 10.1038/s42256-020-00257-z (2020).
3. Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. Densely connected convolutional networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2261–2269, DOI: 10.1109/CVPR.2017.243 (2017).
4. Cohen, J. P., Morrison, P. & Dao, L. Covid-19 image data collection. *arXiv* 2003.11597 (2020).
5. Signoroni, A. et al. Bs-net: learning covid-19 pneumonia severity on a large chest x-ray dataset. *Med. Image Analysis* **71**, 102046, DOI: 10.1016/j.media.2021.102046 (2021).
6. Maguolo, G. & Nanni, L. A critic evaluation of methods for covid-19 automatic detection from x-ray images (2020). *arXiv*:2004.12823.
7. López-Cabrera, J., Portal Diaz, J., Orozco, R., Lovelle, O. & Perez-Diaz, M. Current limitations to identify covid-19 using artificial intelligence with chest x-ray imaging (part ii). the shortcut learning problem. *Heal. Technol.* **11**, DOI: 10.1007/s12553-021-00609-8 (2021).
8. DeGrave, A. J., Janizek, J. D. & Lee, S.-I. Ai for radiographic covid-19 detection selects shortcuts over signal. *Nat Mach Intell* **3**, 610–619, DOI: 10.1038/s42256-021-00338-7 (2021).
9. Bassi, P. R. A. S. & Attux, R. Covid-19 detection using chest x-rays: is lung segmentation important for generalization? (2021). 2104.06176.
10. Teixeira, L. O. et al. Impact of lung segmentation on the diagnosis and explanation of covid-19 in chest x-ray images. *Sensors* **21**, DOI: 10.3390/s21217116 (2021).
11. Guan, W.-j. et al. Clinical characteristics of coronavirus disease 2019 in china. *New Engl. J. Medicine* **382**, 1708–1720, DOI: 10.1056/NEJMoa2002032 (2020). https://doi.org/10.1056/NEJMoa2002032.
12. Kim, E. A. et al. Viral pneumonias in adults: Radiologic and pathologic findings. *RadioGraphics* **22**, S137–S149, DOI: 10.1148/radiographics.22.suppl_1.g02oc15s137 (2002). PMID: 12376607.
13. Liu, Z., Luo, P., Wang, X. & Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)* (2015).
14. Kumar, N., Berg, A. C., Belhumeur, P. N. & Nayar, S. K. Attribute and simile classifiers for face verification. In 2009 IEEE 12th International Conference on Computer Vision, 365–372, DOI: 10.1109/ICCV.2009.5459250 (2009).
15. Rajnoha, M., Burget, R. & Povoda, L. Image background noise impact on convolutional neural network training. In 2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 1–4, DOI: 10.1109/ICUMT.2018.8631242 (2018).
16. Ronneberger, O., P.Fischer & Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, vol. 9351 of *LNCS*, (Springer, 2015).
17. Zhang, D., Wang, J. & Zhao, X. Estimating the uncertainty of average f1 scores. *Proc. 2015 Int. Conf. on The Theory Inf. Retr.* DOI: 10.1145/2808194.2809488 (2015).
18. Salvatier, J., Wiecki, T. & Fonnsebeck, C. Probabilistic programming in python using pymc3. *PeerJ Comput. Sci.* DOI: 10.7287/PEERJ.PREPRINTS.1686V1 (2016).
19. Homan, M. D. & Gelman, A. The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.* **15**, 1593–1623 (2014).
Acknowledgments
This study was financed by the Italian Institute of Technology (IIT).

Author Contributions
Pedro R.A.S. Bassi developed the concept, implemented the neural networks, and analyzed the results. Andrea Cavalli supervised and reviewed the work.

Competing Interests
The authors declare no competing interests.

Corresponding Author
Correspondence to Pedro R.A.S. Bassi and Andrea Cavalli.