UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Elétrica e de Computação

PEDRO RICARDO ARIEL SALVADOR BASSI

A STUDY OF DEEP NEURAL NETWORKS FOR IMAGE RECOGNITION IN BCIs AND COVID-19 DETECTION

UM ESTUDO DE REDES NEURAIS PROFUNDAS APLICADAS A RECONHECIMENTO DE IMAGENS EM BCIs E DETECÇÃO DE COVID-19

CAMPINAS

2021

PEDRO RICARDO ARIEL SALVADOR BASSI

A STUDY OF DEEP NEURAL NETWORKS FOR IMAGE RECOGNITION IN BCIs AND COVID-19 DETECTION

UM ESTUDO DE REDES NEURAIS PROFUNDAS APLICADAS A RECONHECIMENTO DE IMAGENS EM BCIs E DETECÇÃO DE COVID-19

Dissertation presented to the School of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Electrical Engineering, in the area of Computer Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e de Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Elétrica, na área de Engenharia de computação.

Supervisor/Orientador: ROMIS RIBEIRO DE FAISSOL ATTUX

Este trabalho corresponde à versão final da dissertação defendida pelo aluno Pedro Ricardo Ariel Salvador Bassi, orientado pelo Prof. Dr. Romis Ribeiro de Faissol Attux.

CAMPINAS

2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

B294s

Bassi, Pedro Ricardo Ariel Salvador, 1996-
A study of deep neural networks for image recognition in BCIs and COVID-19 detection / Pedro Ricardo Ariel Salvador Bassi. – Campinas, SP : [s.n.], 2021.

Orientador: Romis Ribeiro de Faissol Attux.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Interfaces cérebro-computador. 2. Aprendizado profundo. 3. Aprendizado por transferência. 4. COVID-19. 5. Imagem de raio-X. I. Attux, Romis Ribeiro de Faissol, 1978-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Um estudo de redes neurais profundas aplicadas a reconhecimento de imagens em BCIs e detecção de COVID-19
Palavras-chave em inglês:
Brain-computer interfaces
Deep learning
Transfer learning
COVID-19
X-rays
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Romis Ribeiro de Faissol Attux [Orientador]
Levy Boccato
Everton Zaccaria Nadalin
Data de defesa: 17-08-2021
Programa de Pós-Graduação: Engenharia Elétrica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-8995-9423
- Currículo Lattes do autor: http://lattes.cnpq.br/2499986311212223

COMISSÃO JULGADORA – DISSERTAÇÃO DE MESTRADO

Candidato(a): Pedro Ricardo Ariel Salvador Bassi RA: 157007
Data da defesa: 17 de agosto de 2021, 10 horas.
Título da Tese: “Um Estudo de Redes Neurais Profundas Aplicadas a Reconhecimento de Imagens em BCIs e Detecção de COVID-19”

Prof. Dr. Romis Ribeiro de Faissol Attux (Presidente)
Prof. Dr. Levy Boccato (Titular)
Dr. Everton Zaccaria Nadalin (Titular Externo)

A Ata de Defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no SIGA (Sistema de Fluxo de Dissertação/Tese) e na Secretaria de Pós-Graduação da Faculdade de Engenharia Elétrica e de Computação.

Gostaria de dedicar esse trabalho a aqueles que me ajudaram a chegar até aqui. Dentre eles, devo mencionar o meu orientador, professor Romis Attux, que me deu suporte em toda pesquisa conduzida no meu mestrado e iniciações científicas. Outra pessoa importante foi Willian Rampazzo, que me ajudou com a programação durante minhas iniciações e nos trabalhos com SSVEP. Menciono também a UNICAMP, por toda sua qualidade e estrutura, que desfrutei durante minha graduação e mestrado.
Gostaria também de dedicar esse trabalho aos meus pais, Adalberto Bassi e Cleonice Bassi, por terem me proporcionado a educação que me levou até aqui. Por fim, à minha namorada, Luíza Pinheiro, por estar comigo e me fazer rir durante todo meu mestrado.

AKNOWLEDGEMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

RESUMO

Neste trabalho utilizamos redes neurais profundas (DNNs, *deep neural networks*) para reconhecimento de imagens nos contextos de interfaces cérebro-computador (BCIs, *brain-computer interfaces*) e detecção de COVID-19. Com BCIs, classificamos sinais de eletroencefalografia (EEG) em uma interface de um canal, baseada em Potenciais Visualmente Evocados em Regime Estacionário, a qual não requer calibração no usuário final. Convertemos sinais de EEG em espectrogramas e utilizamos *transfer learning*. Modificamos e aplicamos uma técnica de *data augmentation*, SpecAugment, criada para reconhecimento de fala. Alcançamos 82,2% de acurácia de teste média e F1-Score média de 0,825 usando o eletrodo Oz e sinais de 0,5 segundos. Nesta configuração, nossa DNN superou FBCCA e SVMs. O uso de *transfer learning* não melhorou as acurácias, mas deixou o treinamento mais rápido. SpecAugment criou um pequeno aumento de acurácia e pode ser combinada com janelamento para gerar melhor performance. No nosso estudo com COVID-19, classificamos raios-X torácicos como COVID-19, pneumonia ou normal, utilizando redes neurais densas (DenseNets). Utilizamos *transfer learning* duplo, treinando as redes em três bases de dados, ImageNet, NIH ChestX-ray14 e então o *dataset* de COVID-19. Nós propusemos uma mudança na técnica de *transfer learning* duplo, a qual chamamos de *output neuron keeping* (ONK), que se baseia na manutenção de neurônios de saída treinados. Atingimos 100% de acurácia de teste e constatamos que *transfer learning* duplo e ONK melhoraram performances, principalmente no início do treinamento. Analisamos as redes com *Layer-wise Relevance Propagation* (LRP), que gerou mapas de calor, os quais indicaram que palavras nos raios-X podem influenciar os classificadores. Entretanto, descobrimos que essa influência pouco mudava a acurácia. Na última parte de nosso trabalho, estudamos como bases de dados pequenas e mistas, como as disponíveis na época deste estudo, poderiam afetar a generalização de redes neurais, e como isso poderia ser melhorado com segmentação de pulmões. Treinamos uma DenseNet e uma DNN empilhada (formada por uma U-Net, um módulo intermediário original e uma DenseNet) que realiza segmentação e classificação. Novamente, classificamos raios-X torácicos como COVID-19, pneumonia ou normal. Avaliamos as redes utilizando um *dataset* externo (criado em localizações distintas). Com inferência Bayesiana estimamos distribuições de

probabilidades de métricas de performance. A rede empilhada alcançou uma AUC de teste de 0,917, e a rede sem segmentação, 0,906. Com segmentação, inferência Bayesiana indicou acurácia média de teste de 76,1%, com HDI de 95% (intervalo em que se encontra 95% da massa de probabilidade) de [0,695, 0,826], e, sem segmentação, de 71,7% e [0,646, 0,786]. Comparamos mapas de calor criados com LRP com raios-X avaliados por radiologistas com a pontuação de Brixia. Isso indicou que áreas nas quais os radiologistas encontraram fortes sintomas de COVID-19 também eram aquelas que mais tinham influenciado a rede empilhada. Validação externa indicou acurácias menores que interna, mostrando a existência de viés do conjunto de dados, que é diminuído com segmentação. Nossas performances no *dataset* externo mostram que redes neurais podem ser treinadas em *databases* pequenas e mistas e ainda sim detectarem COVID-19. Porém, uma base grande, aberta e de boa qualidade seria muito benéfica.

Palavras-chave: Interfaces cérebro-computador, Aprendizado profundo, Aprendizado por transferência, COVID-19, Imagem de raio-X

ABSTRACT

In this work, we utilized deep neural networks (DNNs) for image recognition in two problems: brain-computer interface (BCI) classification and COVID-19 detection. In the context of BCI, we proposed new strategies for classifying electroencephalography (EEG) signals in a single-channel BCI based on steady-state visually evoked potentials (SSVEP). Our BCI did not require calibration by the final user. We converted the EEG signals to spectrograms and used transfer learning. We modified and applied SpecAugment, a data augmentation technique, originally created for speech recognition. We reached 82.2% mean test accuracy and 0.825 mean F1-Score, using the Oz electrode and a small data window length (0.5 s). In this configuration, our DNN surpassed FBCCA and SVMs. Transfer learning did not improve accuracies, but made training faster, while SpecAugment created a small performance increase and could be successfully combined with window slicing to create higher accuracies. In the first part of our COVID-19 detection study, we utilized dense convolutional networks (DenseNets) and transfer learning to classify chest X-rays as COVID-19, pneumonia or normal. We applied twice transfer learning, training the networks on three datasets, ImageNet, NIH ChestX-ray14 and then the COVID-19 database. We proposed a novel modification to twice transfer learning, output neuron keeping (ONK), which is based on keeping trained neurons of the output layer. We achieved 100% test accuracy, twice transfer learning and output neuron keeping improved performances, mainly in the beginning of the training process. We analyzed the networks with Layer-wise Relevance Propagation (LRP), which generated heatmaps, showing that words on the X-rays can influence the classifiers. But we discovered that this influence only slightly changed accuracy. In the last part of our work, we studied how the utilization of small and mixed datasets, like the ones available at the time of this study, could affect generalization, and if lung segmentation could improve it. We trained a dense neural network and a stacked DNN, composed of a U-Net (for segmentation), an original intermediate module and a DenseNet (for classification). Again, we classified chest X-rays as pneumonia, normal or COVID-19. To understand the generalization capability, we evaluated the networks on an external dataset (from distinct localities) and used Bayesian inference to estimate the probability distributions of performance metrics.

The stacked DNN achieved 0.917 test AUC, and the DenseNet, 0.906. With segmentation, Bayesian inference indicated a mean test accuracy of 76.1% and [0.695, 0.826] 95% high density interval (HDI, an interval with 95% of the probability mass) with segmentation and, 71.7% and [0.646, 0.786] without it. We also proposed a novel DNN evaluation technique, comparing LRP heatmaps with COVID-19 X-rays analyzed by radiologists with the Brixia score. It indicated that areas where radiologists found stronger symptoms of COVID-19 were also the most important for the stacked DNN classification. External validation showed smaller accuracies than internal validation, indicating dataset bias, which is reduced by segmentation. Our results indicate that DNNs can be trained in small and mixed datasets and still detect COVID-19, but a large, high quality and open database would be highly beneficial.

Keywords: Brain-computer interfaces, Deep learning, Transfer learning, COVID-19, X-rays

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Our main objective in this study is to utilize deep neural networks (DNNs) for image analysis and classification. In this general field, we focused on two specific biomedical applications: analyzing spectrograms, created from electroencephalography (EEG) signals, in the context of brain-computer interfaces (BCIs), and COVID-19 detection using lung X-Rays.

In the work with BCIs, we utilized the convolutional DNN called VGGish (HERSHEY et al., 2017), pretrained with AudioSet (GEMMEKE et al., 2017), a large database for audio classification. We used transfer learning to classify EEG signals, which were created using a steady-state visually evoked potentials (SSVEP) based BCI. This BCI utilized a single electrode, making it more practical and wearable, and less expensive. Furthermore, it used a small data length, making it faster. Before classification, we transformed the signals into spectrograms, using the short-time Fourier transform (STFT), because convolutional neural networks have been known to achieve good results in image analysis (GOODFELLOW; BENGIO; COURVILLE, 2016).

As DNNs have a tendency to overfit when trained with relatively small datasets (GOODFELLOW; BENGIO; COURVILLE, 2016) - like our EEG dataset -, we used two data augmentation techniques, alongside transfer learning, to avoid this problem. The first one is window slicing (WS), a technique that was already successfully used for SSVEP classification in other studies (KWAK; MÜLLER; LEE, 2017). The second method is called SpecAugment, a technique created for data augmentation in speech recognition, which we adapted to the context of SSVEP classification. As far as we could verify, this technique had never been used in the context of BCIs.

Our objectives in the BCI study were: to transform a signal classification problem into an image classification task, using STFT to create spectrograms; to adapt a data augmentation technique used in another area, SpecAugment, to the context of BCIs, and to analyze its efficiency; contribute to the advance of assistive technologies, finding techniques that can improve the BCI classifier accuracy. The results of the study were recently published (BASSI; ATTUX, 2021b).

In April 2020, we started working with COVID-19 detection in chest X-Ray images, in the beginning of the COVID-19 pandemic. We utilized deep neural networks and twice transfer learning. Furthermore, we proposed an original technique, which

changes the twice transfer approach, and called it Output Neuron Keeping (ONK) (BASSI; ATTUX, 2020). The main objective in this work was to classify patients as healthy, pneumonia or COVID-19. In our twice transfer learning approach, the networks were first trained with ImageNet (DENG et al., 2009), an image classification dataset containing millions of figures, then on ChestX-ray14 (WANG et al., 2017a), a database with more than 100000 chest radiographs, and, finally, on the COVID-19 database, which contained hundreds of images.

Our method, ONK, consists in exploring similarities between the last two datasets (ChestX-ray14 and COVID-19 databases), keeping the output neurons that classify the pneumonia and healthy classes present in both datasets, instead of replacing all neurons in the output layer.

The results of this first work with COVID-19 classification were published firstly as preprints and, finally, as (BASSI; ATTUX, 2021a).

We continued the study with COVID-19 X-rays, in the second semester of 2020, exploring attention mechanisms to avoid sources of bias in the images. We utilized two stacked neural networks, a U-net (RONNEBERGER; FISCHER; BROX, 2015), which segments the lung regions in the X-rays, and a DenseNet201 (HUANG et al., 2017), which classifies the regions as COVID-19, pneumonia or healthy. Using this methodology, we can classify the images based only on the important regions and ignore biases that can be present in the rest of the image. We applied external validation to understand the effects of segmentation and of small and mixed datasets, like the COVID-19 databases available at the time of this study, on DNN generalization. To further analyze the networks, we used Bayesian inference to create interval estimates of the performance metrics and we proposed a novel evaluation method, comparing layer-wise relevance propagation (LRP) heatmaps to X-rays analyzed by radiologists with the Brixia score (BORGHESI; MAROLDI, 2020). This part of our study is also available as a preprint (BASSI; ATTUX, 2021b).

Our objectives in the study of COVID-19 classification were: researching if X-rays can become an auxiliary method for COVID-19 diagnosis; introducing and checking the efficacy of the ONK technique; studying attention mechanisms in feedforward neural networks.

In the next section of this dissertation, we will perform a literature review, highlighting the main concepts that served as the basis for our work, as fundamentals

of neural networks and deep learning, the DNN architectures that we used, transfer learning, data augmentation and LRP. Afterwards, we will have three sections, each one referring to one part of our work (EEG classification, COVID-19 detection and COVID-19 detection with segmentation). They will be divided into the subsections, such as related work, data analysis (where we talk about the datasets and data processing), methodology (where we discuss the neural networks, along with their training and evaluation procedures), results and discussion, and conclusion. In the end, we will have a section for the general conclusion.

2 LITERATURE REVIEW

In this section we explain the basic concepts of neural networks and the DNN architectures that we used.

2.1 NEURAL NETWORKS

An artificial neural network (NN) is a highly parallelized structure composed of interconnected simple processing units. These units are inspired by certain elements of the operation of the biological neuron and are called artificial neurons (GOODFELLOW; BENGIO; COURVILLE, 2016).

The most common artificial neuron model performs a linear combination of its inputs, defined by a set of synaptic weights, and integrates it with a bias. The result of this operation is fed, as an input, to an activation function, which is generally non-linear. An example commonly used in DNNs is the rectified linear unit function (ReLU), defined in equation 1.

$$y = max(0, x) \tag{1}$$

It is usual to organize an NN in layers, and, in a fully-connected (or dense) layer, every neuron receives, as inputs, the outputs of the neurons in the previous layer, and sends its output to every neuron in the next layer.

A NN is considered feedforward if there are no feedback cycles in the neuron connections. It is defined by an input layer, hidden layers (if any) and an output layer. A NN is considered deep if it has a number of layers that is considered “large”, but there is no consensus about this number (GOODFELLOW; BENGIO; COURVILLE, 2016).

Deep convolutional neural networks (DCNNs) are important tools for image classification and computer vision problems. They are neural networks that have one or more layers that process the input information using discrete convolution kernels. The operation is illustrated in figure 1. We observe that the kernel, composed of weights that are optimized during the training process, moves across the input. In each position, one output element is generated by multiplying each kernel weight by an input element and summing the results. A convolutional layer also has an activation function,

applied to each element of the convolution output. Furthermore, these layers can perform multiple convolutions in parallel, using different kernels and generating multiple output channels.

Since the convolutional kernel moves across the input, the convolutional layers can apply the same filters over many regions of an image, generating a certain degree of invariance to translations. A single convolution can be seen as a single artificial neuron, with its synaptic weights defined by the kernel, being shifted over the input. Alternatively, we can understand each output element of a convolution (point in a feature map) as the output of a neuron; the parameters of these neurons are shared, because they are defined by the kernel. Therefore, a convolutional layer generally has less parameters than a fully-connected one.

Figure 1. Illustration of a two-dimensional convolution, at the left we have the input, in the center the kernel and at the right the feature map being formed.

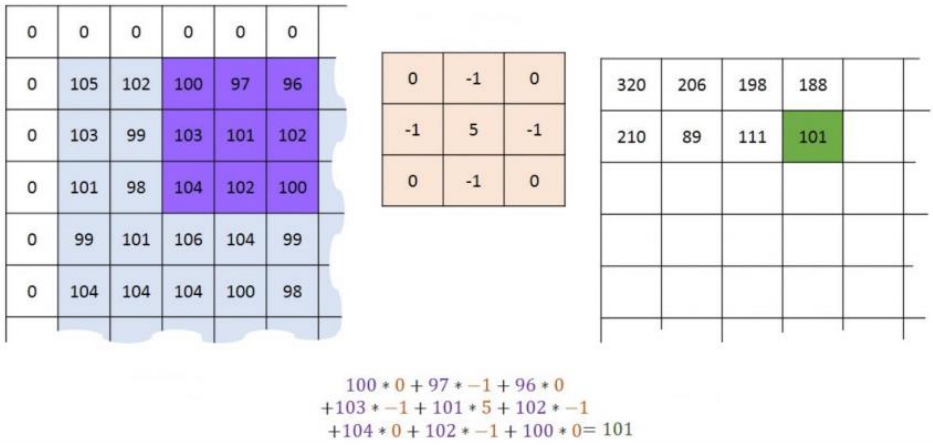
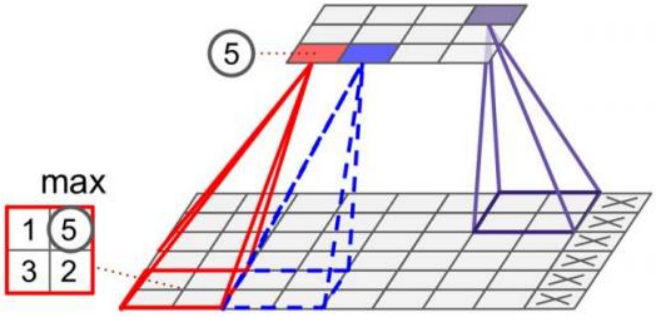


Image extracted from Zhang (2020).

The convolutional layers are commonly followed by maxpooling layers. A pooling layer subsamples the output of the convolution, which is also known as a feature map. In the case of maxpooling, it divides the feature map in rectangles, defined by a kernel size, without superposition and, for each rectangle, it selects its maximum value. Figure 2 illustrates this concept.

Figure 2. Illustration of maxpooling, with 2x2 kernel, the structure below represents the input and the one above the output.



Adapted from Géron (2019).

The neural network training process is defined by the optimization of its trainable parameters (kernels, synaptic weights and biases). The optimization procedure aims to reduce a loss value, which, in supervised training, is defined by a function that compares the NN output to the desired output (or label) for the given input. As a rule, the available data is divided into three sets: training, validation and test. The training dataset is used to optimize the network adjustable parameters. The test database is used to simulate the NN generalization capability, i.e., how it performs with unseen data. Finally, the validation loss is used to estimate the test loss during training and to adjust hyperparameters.

Two common problems related to the training process of neural networks are called overfitting and underfitting. To solve a task, a NN must model a statistical distribution, which defines the relationship between inputs and outputs. When a model is too flexible, it may capture noise and slight variations in training data, which do not reflect the ideal statistical distribution. In this case, the network will have a small training loss, but a high test loss and a poor generalization. This phenomenon is called overfitting, and it can be seen, for instance, when a large NN, with many parameters, is trained using a small dataset. The opposite case is underfitting, defined by a model with small flexibility trying to approximate a complex statistical distribution. An underfitted NN has large loss values in all datasets. We can find underfitting when training small networks to solve complex tasks in large datasets or when training time was insufficient.

A well-established approach to mitigate the risk of overfitting consists in using regularization techniques. An example is L2 normalization, a method that adds to the loss function a penalty term proportional to the L2 norm of the weight vector, thus

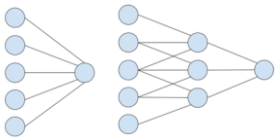
reducing the model flexibility. Another technique is called dropout, it consists in randomly removing neurons during training, replacing their output values by zero (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.1.1 VGG

The VGG architecture was first proposed in Simonyan and Zisserman (2014) in the context of image classification. At the time, it achieved state-of-the-art performance on the ImageNet dataset (DENG et al., 2009).

The structure is deep, with 19 layers in the configuration E (SIMONYAN; ZISSERMAN, 2014). It is also characterized by the utilization of many convolutional layers, which employ small kernels (3x3). Stacking many of these layers enables the last ones to have a large receptive field, i.e., each element of their output will be influenced by a large number of the DNN input elements, as would be obtained with less layers using larger kernels (SIMONYAN; ZISSERMAN, 2014). For example, in a one-dimensional convolution, using a kernel with size 5, each output will be influenced by 5 elements in the input. But, if we stack two convolutional layers with a kernel of size 3, the outputs of the second layer will also be influenced by 5 elements of the input. Figure 3 illustrates this example: the circles represent the convolution inputs and outputs, whereas the lines indicate which input elements influence an output.

Figure 3. Receptive field illustration. On the left, a convolution with a kernel size of 5. On the right, two stacked convolutions with kernel size of 3.



An advantage of using more convolutions with smaller kernels instead of less convolutions with larger kernels is the reduction of the total number of parameters in the network (considering two-dimensional convolutions). For example, one 5x5 kernel has 25 weights, whereas two 3x3 kernels have 18 weights (SIMONYAN; ZISSERMAN, 2014).

Another advantage is that the DNN becomes more nonlinear, as each convolutional layer has a nonlinear activation function (SIMONYAN; ZISSERMAN, 2014). This makes the model more flexible.

In (HERSHEY et al., 2017), the authors changed the VGG network and used it in audio classification, classifying the AudioSet dataset (GEMMEKE et al., 2017). The database is composed of more than one million 10 seconds audio clips from YouTube. They are labeled by humans into hundreds of classes, like ambient sounds, human made sounds and instruments (GEMMEKE et al., 2017). The modified VGG architecture can be called VGG or VGGish (we will call it VGGish), and its differences from the original architecture are the adoption of batch normalization and the structure of the last layer.

The VGGish architecture is shown in Table 1. All layers have ReLU activation, except for the last, which uses softmax activation. All convolutions are two-dimensional, with 3x3 kernels, 1x1 stride and zero padding. The maxpooling layers are also 2D, with 2x2 kernel, stride of 2 and dilation of 1. The DNN has 62 million trainable parameters.

Table 1. VGGish architecture.

Convolution, 64 channels
Maxpool
Convolution, 128 channels
Maxpool
Convolution, 256 channels
Convolution, 256 channels
Maxpool
Convolution, 512 channels
Convolution, 512 channels
maxpool
Fully connected, 4096 neurons
Fully connected, 4096 neurons
Fully connected, 128 neurons

.

We modified this pretrained network to use transfer learning in our study with BCIs. Our hypothesis was that using a large DCNN, pretrained for audio classification using spectrograms, would be beneficial to our problem of EEG spectrogram classification.

2.1.2 Dense Neural Networks

A densely connected neural network, dense neural network or DenseNet is a feedforward DNN architecture presented in HUANG et al., (2017).

The distinct characteristic of the DenseNet is that its layers are divided into dense blocks. A dense block is a structure composed of a sequence of (generally convolutional) layers, in which every layer propagates its output to all subsequent layers. Therefore, every layer in the block receives the outputs of every previous layer in the block (HUANG et al., 2017). Figure 4 illustrates a dense block with 5 convolutional layers, and the arrows show how they are connected.

Figure 4. Illustration of a dense block. Colored rectangles indicate feature maps (outputs of convolutional layers).

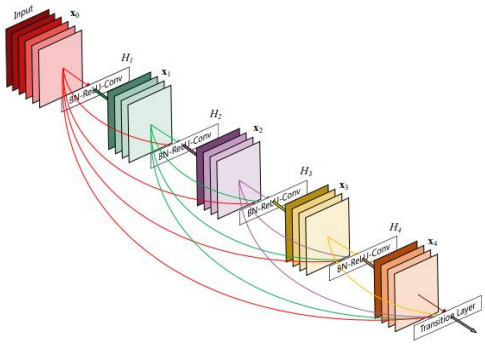


Image from Huang et al. (2017).

A typical dense neural network has many dense blocks, with transition layers between them. These layers consist of convolutions and pooling operations. The blocks are connected in a typical feedforward way, in which a block only sends its outputs to the next one (HUANG et al., 2017). Therefore, a DenseNet has a very large number of layers (even more than 100), hence can be considered a very deep NN.

In very deep neural networks, information is lost when the signal propagates from the input to the output or when the loss backpropagates. The additional connections in a DenseNet create additional direct paths to help signal and gradient propagation. This reduces information loss and allows efficient training of very deep networks (HUANG et al., 2017).

Furthermore, large DNNs tend to have redundancy in the features learned by each layer. In ResNets, another architecture with many layers, research has shown that layers can have a very small contribution to the networks and can be randomly removed during training for regularization (HUANG et al., 2017). Since a DenseNet allows layers to have direct access to many previous layers' outputs, redundancy is avoided. Therefore, convolutions in a dense network can have fewer channels (e.g.,

12), creating a DNN with fewer parameters in comparison to other structures with the same number of layers (HUANG et al., 2017).

A dense neural network with 121 layers (DenseNet121), called CheXNet (RAJPURKAR et al., 2017), achieved high accuracy classifying the NIH ChestX-Ray14 database. The dataset used to train it contains more than 100,000 chest X-rays, which can be related to 14 different lung diseases and healthy patients (WANG et al., 2017). This inspired us to also use DenseNets for our COVID-19 detection problem. We tested networks with 121 and with 201 layers.

2.1.3 U-Net

The U-Net architecture was proposed in (RONNEBERGER; FISCHER; BROX, 2015) to segment biomedical images, in small datasets.

A segmentation problem means that the neural network must indicate the class to which every pixel in an image belongs, segmenting the image into different classes. For example, in lung segmentation, we may input a chest X-ray to the network and expect it to create a grayscale image called a mask. In this mask, the whiter a pixel, the higher the probability that it belongs to the lungs' region.

The U-Net is a fully convolutional architecture, i.e., all of its layers are convolutional. It has two symmetrical paths, like an autoencoder. Figure 5 illustrates the architecture. The path on the left is the contracting path, where each consecutive layer produces smaller outputs than the previous one, with a larger number of channels. This path extracts context information from the image. The path on the right is the expanding one: each of its layers produces an output that is larger and with fewer channels than the previous layer. This path allows better localization of the classes on the images. There are multiple connections between the two paths (in grey on the image).

Figure 5. U-Net architecture. Blue rectangles represent feature maps, with their size indicated at their side and the number of channels in the top. White rectangles are copied feature maps.

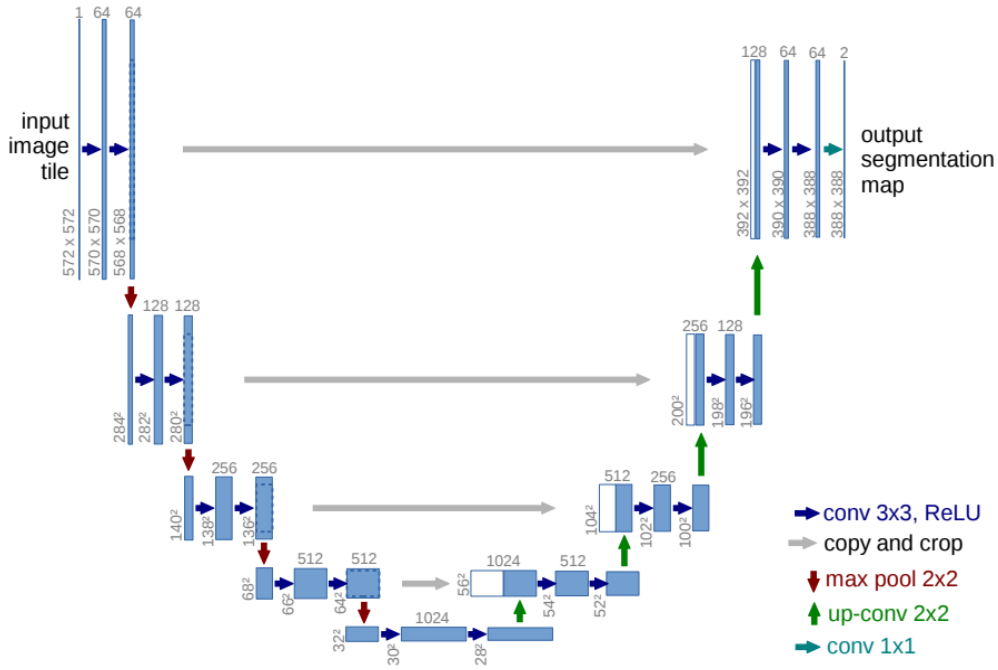


Image extracted from Ronneberger; Fischer; Brox (2015).

Medical segmentation datasets are normally created manually, hence are generally small, and an important characteristic of U-Nets is their capability to be trained in small databases.

U-Nets were already used to segment lungs in chest X-rays (HEO et al., 2019), creating masks that would be used to select only regions of interest to a classifier DNN.

We decided to study the difference that segmentation creates on the generalization capability of DNNs that classify COVID-19. Therefore, we stacked a U-Net with a DenseNet, which would classify only the lung regions in the radiographs. To better evaluate the generalization capability and real-world performance of the DNNs, we used external validation and test datasets in our study with segmentation.

2.2 TRANSFER LEARNING

When we have a high-dimensional input in a neural network, the input space becomes larger and the data sparser. Therefore, with larger inputs, we need more data points to create a good statistical model of the input and label distributions. This is a consequence of the so-called “curse of dimensionality” (TRUNK, 1979).

Deep neural networks are very flexible models, which can represent complex statistical relations in terms of potentially elevated number of parameters. Due to this flexibility, when we do not have enough data, they will also model small variations and noise in the training dataset, which do not reflect the ideal data distribution. Hence, there is a tendency of attaining a high accuracy when classifying inputs that were used during training, but a low performance with unseen data, i.e., bad generalization (GOODFELLOW; BENGIO; COURVILLE, 2016).

Therefore, DNNs tend to overfit when trained with small datasets and large inputs, like images with high resolution. The transfer learning technique has the objective of improving generalization and avoiding overfitting.

Using transfer learning, we train a neural network twice and sequentially, in two datasets with similar data and tasks. The second task is the one we wish to solve, and, if the procedure is successful, it will improve generalization in the second task. Transfer learning is more useful if the first dataset is much larger than the second one (GOODFELLOW; BENGIO; COURVILLE, 2016).

The technique is based on the concept of representation learning (BENGIO; COURVILLE; VINCENT, 2013), which states that the layers in a DNN learn to extract different features of the data, while mapping its input onto a feature space. Layers nearer the output of the network learn more abstract features (GOODFELLOW; BENGIO; COURVILLE, 2016). For instance, in image classification, a layer near the input may learn to recognize borders, whereas one near the output, to recognize cats. The objective of transfer learning is using features learned in the first dataset, with a large amount of data, to solve the second task more easily.

If the two tasks are very similar, it is more likely that highly abstract features learned in the first task will be useful for solving the second one. Thus, more layers will have learned useful information and transfer learning will bring more benefits. Therefore, the ideal case is when we have a very large first dataset, with a task that is similar to that of the second database.

Nowadays, there are many pretrained networks available for download, and, therefore, on many occasions we do not need to train the network in the first dataset.

Nonetheless, a pretrained DNN still has to be modified before carrying out the second training phase. The most usual approaches are: adding new layers at the end of the DNN, substituting its last layers for new ones, or substituting the last layer. In all

those cases, the new layers have random parameters. The more similar the two tasks in transfer learning, the more useful will be the features learned by the last layers.

We used the transfer learning techniques in both our work with EEG and with COVID-19 classification. In The COVID-19 problem we utilized the twice transfer learning technique, as explained in section 2.2.1.

In SSVEP classification, we used transfer learning by employing a DNN pretrained over a large audio classification dataset. Our hypothesis was that SSVEP classification could be related to the problem of trying to identify a specific sound in a noisy environment. Hence, we hypothesized that our task shared similarities with audio classification using spectrograms.

2.2.1 Twice Transfer Learning

Sometimes we may not find a large dataset with a task that is very similar to the one we are trying to solve. In image classification tasks, it is common to choose ImageNet (DENG et al., 2009) as the first dataset for transfer learning. But, in our work with COVID-19 detection, we did not believe that the ImageNet classification task, with a thousand classes, was similar enough to X-ray classification.

Another dataset, NIH ChestX-ray14 (WANG et al., 2017a), presents a much more akin task: classifying chest X-rays as 14 different lung diseases or healthy individuals. Furthermore, the dataset has more than 100,000 images. But starting with an ImageNet pretrained DNN may make training in ChestX-ray14 faster, and features learned on ImageNet may be carried through ChestX-ray14 training and improve generalization on the COVID-19 classification task.

Therefore, we decided to use a twice transfer learning procedure for COVID-19 detection. A DNN was first trained on ImageNet, then on ChestX-ray14 and finally on the COVID-19 dataset, assembled by us. Looking for other works that used this technique we found Cai, Liu and Guo (2018) where the authors successfully used twice transfer learning to classify mammograms.

2.3 DATA AUGMENTATION

In addition to transfer learning, a way to avoid overfitting is to increase the size of the dataset being used, providing a better statistical representation of the task we

wish to solve. Data augmentation techniques “artificially create” more data from the existing inputs (GOODFELLOW; BENGIO; COURVILLE, 2016).

There are many data augmentation techniques, which depend on the type of data being analyzed, like image recognition or classification of time series. We used data augmentation in both contexts of BCI-SSVEP and COVID-19 classification.

2.3.1 Image Data Augmentation

Image rotation, translation and horizontal flipping are common data augmentation techniques for image recognition, and we chose to use them in the COVID-19 detection task.

These transformations do not modify the class of the X-ray images, being used to generate new images. They also make the DNN more resistant to the natural variances in the images, like the position of the patient.

2.3.2 Window Slicing

Window slicing (WS) (CUI; CHEN; CHEN, 2016) is a data augmentation technique for time series or signals. We used it in our SSVEP classification work, and it was already used in other SSVEP studies, like in Kwak, Müller and Lee (2017), in which it improved the accuracies of neural networks.

In our BCI study, WS was also used to define our data length i.e., the number of samples used to send a command to the interface. To perform WS, we begin with the original signals, with a longer duration, from which we take slices of a fixed size (the data length/window size). The first slice starts at the beginning of the time series, and, for each subsequent slice, we add a fixed displacement considering the starting position from the previous slice. The process stops when a slice does not entirely fit in the time series anymore. With this procedure, for each original time series, we create many smaller slices, used as inputs to our neural network, with the same desired output as the original time series taken as the source for the slices. The window size is the desired data length.

2.3.3 Specaugment

SpecAugment (PARK et al., 2019) is a data augmentation technique that is applied to spectrograms. It was created for speech recognition, and we adapted it to the context of BCIs.

As convolutional neural networks tend to be effective at analyzing images, we decided to convert our SSVEP signals into spectrograms, before sending them to the DCNN classifier. This same conversion is usual in audio classification tasks. For example, the January 2021 LibriSpeech test-other (a large speech recognition dataset) best performing model uses spectrograms and two-dimensional convolutions (ZHANG et al., 2020). This fact allowed us to use transfer learning with an audio classification dataset as the first task, and to hypothesize that techniques used for augmenting audio and speech datasets might also be useful for our EEG study.

SpecAugment was created in 2019 and allowed its authors to achieve state-of-the-art performances in the speech recognition datasets LibriSpeech 960h and Switchboard 300h (PARK et al., 2019), using deep neural networks.

Although we modified the technique to apply it in a BCI context, here we explain the original method. It has three parts: frequency masking, time masking and time warping. Time warping was based on the TensorFlow function sparse image warp to warp a random point in the horizontal line in the center of the spectrogram, by a random distance, to the right or left, modifying other points in its path. According to Park et al., (2019), this technique has a small effect on accuracy, and it is the most computationally expensive part of SpecAugment. Frequency masking consists in replacing random frequency channels (rows) in the spectrogram by a mask. In time masking, we replace random time intervals (columns) in the spectrogram with masks. Normally, every point in a mask is defined as the spectrogram average value. We can have many time and frequency masks in a spectrogram, and they can have different lengths, taking one or many columns/rows. The number of masks and their sizes are defined with a random approach.

Figure 6 shows an example of a speech recognition spectrogram augmented with SpecAugment.

Figure 6. From top to bottom: original spectrogram, spectrogram after time warp, after a frequency mask and after a time mask.

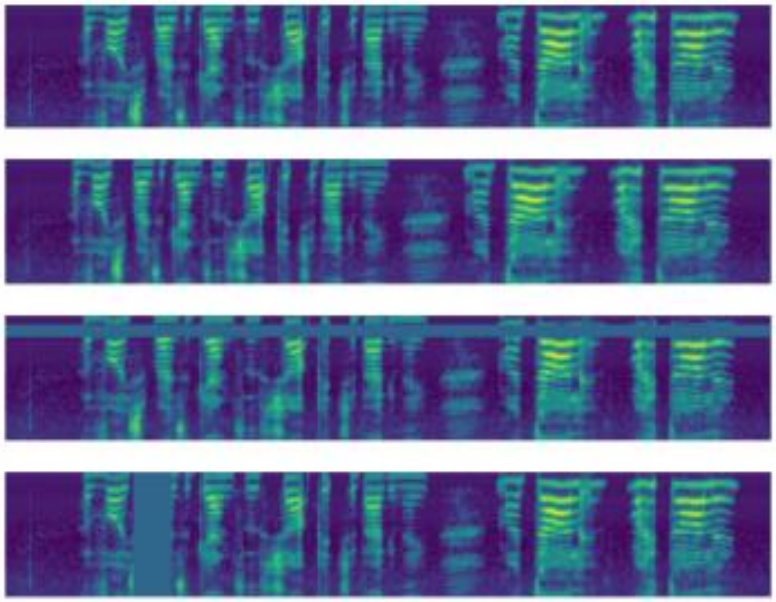


Image from Park et al. (2019).

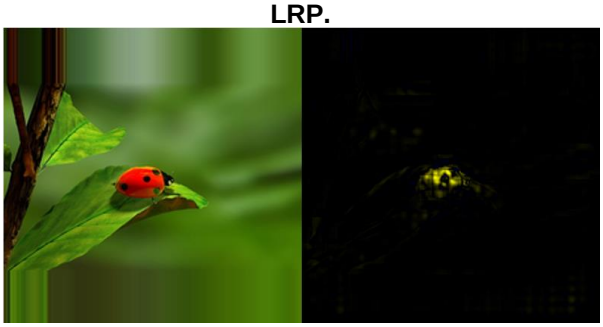
2.4 LAYER-WISE RELEVANCE PROPAGATION

Deep neural networks are complex and non-linear structures, with thousands or millions of parameters and connections. Therefore, interpreting how a network generates its output and why it takes a decision is not a simple task. Layer-wise relevance propagation (LRP) is an explanation technique, created to make DNNs more interpretable by humans. We can use it to decompose a neural network output, creating a heatmap, which shows how each part of the input image contributed to the DNN output (BACH et al., 2015).

We can start relevance propagation from any DNN output neuron, and the meaning of the colors in the resulting heatmap depends on this choice (BACH et al., 2015). For example, in an image classification DNN, if we choose to start the propagation in a neuron that indicates the probability of the image showing a ladybug, yellow colors (high relevance) on the map identify areas that the DNN associated with the class ladybug, while the blue colors (negative relevance) show areas associated with other classes. Dark colors (low relevance) indicate areas with little importance for the DNN decision. In classifiers we generally start propagation from the output neuron with the highest output, i.e., the predicted class. The image below shows an example of an input image, correctly classified by a DNN as a ladybug, and its heatmap. The bright yellow areas in the heatmap reveal the regions of the image that were decisive

for the classification. In this case, they are concentrated in the insect body, as expected.

Figure 7. Picture of a ladybug, correctly classified by a DNN, and its heatmap, created using LRP.



LRP works by propagating the DNN output backwards through the layers, using local propagation rules, which may differ in distinct layers. The signal being propagated is called relevance. All propagation rules have a conservation property: the quantity of relevance received by a neuron from its upper layer is distributed in equal amount to the neurons in the lower layer (MONTAVON et al., 2019). As an illustration, a neuron receives 10 relevance and propagates it to 3 neurons in the lower layer. It could, for example, propagate relevances of 7, 2 and 1 or 5, 3 and 2, but not 7, 5 and 1 or 1, 2 and 3, because these numbers do not add up to 10. The conservation property ensures that the quantity of explanation on the heatmap relates to what can be explained by the DNN output (MONTAVON et al., 2019).

The choice of a propagation rule depends on the DNN architecture and on the specific layer: a good decision generates heatmaps that are more understandable, less noisy and more coherent with the DNN behavior. Open-source implementations of LRP on Python already exist and work with many DNN architectures. An example is a library called iNNvestigate (ALBER et al., 2019).

Three common propagation rules used with DNNs utilizing ReLU activation (e.g., VGG and DenseNet) are the basic rule (LRP-0), the epsilon rule (LRP-ϵ) and the gamma rule (LRP-γ) (MONTAVON et al., 2019). The output of an artificial neuron with ReLU activation is given by the equation below:

$$a_k = max(0, \sum_{j=0}^{J} w_{jk} a_j) \quad (2)$$

In equation 2, a_k is the output of neuron k, in the layer L+1, a_j is the output of neuron j of the previous layer (L), w_{jk} is the synaptic weight of the connection between neurons j and k and J is the number of neurons in the previous layer. a_o is considered 1 and w_{0k} the bias of neuron k.

LRP-0 is then defined as (MONTAVON et al., 2019):

$$R_j = \sum_k \frac{w_{jk}a_j}{\sum_{j=0}^{J} w_{jk}a_j} R_k \tag{3}$$

In equation 3, R_j is the relevance propagated to the neuron j, in layer L, from all the neurons in the next layer (L+1). The summation over k sums over all neurons in the layer L+1. R_k is the relevance of neuron k from layer L+1. We observe that for every neuron k in layer L+1, this rule redistributes the relevance R_k to the neurons in the lower layer according to their contribution for the neuron k activation. If we only apply this rule in LRP, we will produce heatmaps that are equivalent to the gradient times the input. Because the gradient in DNNs is noisy, using only this rule generates noisy heatmaps (MONTAVON et al., 2019).

LRP-ϵ is defined by the following equation:

$$R_j = \sum_k \frac{w_{jk}a_j}{\epsilon+\sum_{j=0}^{J} w_{jk}a_j} R_k \tag{4}$$

In the equation, ϵ is a small positive term, which is used to absorb the relevance of neurons that had a weak or contradictory influence on the activation of neuron k. If the term is large, only the neurons that mostly contributed to the activation will receive a significant amount of relevance. Therefore, this rule changes LRP-0 to create less noisy and sparser heatmaps (MONTAVON et al., 2019).

LRP-γ is defined by the equation below:

$$R_j = \sum_k \frac{a_j\left(w_{jk}+\gamma w_{jk}^{+}\right)}{\sum_{j=0}^{J} a_j(w_{jk}+\gamma w_{jk}^{+})} R_k \tag{5}$$

This rule favors positive contributions to the activation of neuron k, which are indicated by w_{jk}^{+} (a value that is w_{jk} if the contribution of the neuron j is positive, and

zero if it is negative). The positive parameter γ controls how strongly they are favored. This rule makes explanations more stable (MONTAVON et al., 2019).

3 SSVEP CLASSIFICATION WITH DNNS

This section explains in detail our work with SSVEP based brain-computer interfaces.

3.1 SSVEP BASED BCIS

Steady-state visually evoked potentials (SSVEP) are brain activity oscillations, mostly in the visual cortex, which appear when a person gazes at a flickering visual stimulus. Their frequency is that of the stimulating frequency and its harmonics, and the associated signal-to-noise ratio is considered to be high.

To create an SSVEP-based brain-computer interface, we need visual stimuli, flickering at different frequencies, in a stimulation device, like a computer monitor. Furthermore, electroencephalography (EEG) is used to capture the oscillations, and a classifier receives the generated signals and must discover at which stimulus the user was focusing, thus, which command he/she wants to send to the interface (BEVERINA et al., 2003). Since they typically use EEG, SSVEP-based BCIs are non-invasive.

In our work, we created the classifier module of the BCI, working with data from an open dataset. Our classifier is a DNN, which is trained without the data of a test subject. Therefore, we simulate a BCI that does not require calibration on the final user. This aspect makes the classification problem more challenging, increasing the tendency of overfitting and the need of regularization techniques.

3.2 RELATED WORK

Deep neural networks are not the most common SSVEP classification method in BCIs, but they have already been used for this purpose. Here we will analyze three examples: KWAK; MÜLLER; LEE, (2017), NGUYEN; CHUNG, (2019) and LEE; LEE, (2020). As our experiments with BCIs were mainly done in 2019, two of the aforementioned examples are contemporary to it.

In Kwak et al, (2017) an exoskeleton was controlled using a SSVEP based BCI, which used a convolutional neural network as the classifier. The experiment was performed in an ambulatory environment, raising the amount of noise in the BCI. The authors used data from 8 electrodes and analyzed them in the frequency domain (using

FFT). The FFT results created a matrix with 8 rows, one for each electrode, and it was used as input to the NN. The study showed accuracies above 94%, which were above CCA, using a 2 s data length.

In Nguyen and Chung, (2019) the authors used a single-channel (Oz electrode) SSVEP based BCI to create a virtual keyboard, using a DCNN as classifier. Their BCI used a 3D printed headset, containing only two electrodes, which is light and allows simple and fast self-application. Their DNN also analyzed the EEG signals in the frequency domain, applying FFT. Their accuracies were above 97% with 2 s data length, and above 70% with 0.5 s. As in the previous study, the CNN performance surpassed CCA.

In Lee and Lee, (2020) the authors created a NN that has two inputs, a signal in the time domain and its FFT. Their network has a recurrent path, which processes the time signal, and a convolutional path, which analyzes the frequency domain. As KWAK et al, (2017), they employed a multi-channel approach. The NN accuracies surpass CCA when the BCI user was moving.

Some important differences between our BCI study and the aforementioned works are: we simulate a calibration-less BCI, because we do not use the test subject's data for training or validation. Preliminary tests showed us that this makes the classification problem more challenging, as the DNNs' tendency to overfit increases. Kwak, Müller and Lee (2017) and Nguyen and Chung (2019) analyzed the EEG signals in the frequency domain, using FFT, whereas Nguyen and Chung (2019) utilized the time signal and its FFT. We also wanted to provide the DNN frequency and time information, but our approach was different, since we employed spectrograms, created with STFT, to combine this information in a single input image. Finally, we resorted to larger DNNs, while in the studies above the authors utilized a maximum of 5 layers. To avoid the problem of overfitting, more common with deeper architectures, we suggested the usage of transfer learning and SpecAugment, techniques not present in those works.

3.3 DATA ANALYSIS

This section explains the database that we used, the signal and image preprocessing steps and the employed data augmentation.

3.3.1 SSVEP Dataset

The dataset that we used in this part of our study (WANG et al., 2017b) was created in the evaluation of a virtual keyboard. It showed 40 different visual stimuli (with different frequencies and phases), each one corresponding to a different character. We decided to use this database because it is open and relatively large. 35 subjects are present, 8 with previous experience in BCIs, 27 without it. 64-channel whole head electroencephalography (EEG), using the international 10-20 system, was used to capture the data, and the stimulation frequencies ranged from 8 to 15.8 Hz (with 0.2 Hz and 0.5 π frequency and phase differences between adjacent stimuli). The study had 6 blocks of 40 five seconds trials per subject (one for each frequency). More details can be seen in (WANG et al., 2017b).

In our study, we analyzed only two frequencies, 12 and 15 Hz, using only the Oz electrode (based on the international 10-20 system). This electrode stays above the visual cortex and, in a previous study (BASSI; RAMPAZZO; ATTUX, 2019) it created a better classification performance in comparison to other electrodes and combination of electrodes. In that work we utilized triplet deep neural networks to analyze SSVEP spectrograms and we compared results after training just with the Oz electrode data and with data from other electrodes or combinations of multiple electrodes. When using multiple electrodes, the spectrograms were independently created and analyzed, in a single channel approach. The accuracy with the Oz electrode was about 10% better than with the combination of O1, O2, Oz and POz electrodes (BASSI; RAMPAZZO; ATTUX, 2019).

3.3.2 Signal Processing and Window Slicing

Since the dataset consisted of EEG 5 s time signals, with sampling rate of 250 Hz (downsampled from 1000 Hz), each trial had 1250 points. The signals were already filtered, by WANG et al., (2017), with a notch filter at 50 Hz, to remove common powerline noise. We had 420 time series (with 35 subjects, two stimulation frequencies and one electrode), and we started signal processing by applying a common average reference filter (CAR).

The next signal processing step was our first data augmentation technique, window slicing (WS). We used a window size of 0.5 s. This data length simulates a fast

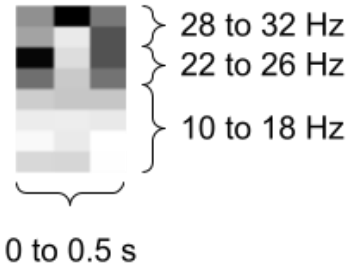
BCI and provides enough frequency resolution to distinguish the 12 and 15 Hz frequencies in our spectrograms. Also, with preliminary tests we discovered that, using longer data lengths, FBCCA (CHEN et al., 2015) generated better results than our DNNs (with 4s windows, FBCCA has about 4% higher accuracy). For this reason, even though they would produce higher accuracies, larger data lengths would have no practical value in our study. That is because a simpler method (FBCCA), which utilizes frequency filters along with canonical correlation analysis and does not require training, could surpass our networks. Other studies (NGUYEN; CHUNG, 2019) show that increasing window size benefits FBCCA more than DNNs.

We utilized two different displacements in WS: 0.5 s and 0.1 s. 0.5 s is the largest value that still allows the exploration of all data, creating 10 windows per signal, without superposition. We used this value in most of our experiments, to minimize training time when comparing methods. 0.1 s displacement generated 46 windows per signal and created a dataset size similar to using 0.5 s and SpecAugment's time masks.

After applying WS we used a short-time Fourier transform (STFT) to create spectrograms (images) from the new signals. The function used rectangular windows, with length of 125 samples (0.5 s) and hop length (number of samples between adjacent STFT columns) of 62 samples (0.248 s). Rectangular windows have a better frequency resolution in the frequencies that we are working at, and, in preliminary tests, they produced better accuracies than Hann and Blackman windows. After applying STFT, we converted its output absolute value to decibels and normalized it between 0 and 1.

We then filtered the spectrograms, removing unimportant frequencies. In SSVEP classification we search for the stimulation frequencies and their harmonics. Therefore, we only maintained the frequencies between 10 and 18 Hz, 22 and 26 Hz and between 28 and 32 Hz. The generated images had size 8x3 (rows x columns), frequency resolution of 2 Hz and time resolution of 0.167 s. Figure 8 shows a spectrogram example. Before sending the spectrograms to the DNNs we resized (as an image, using nearest neighbor interpolation) them to 94x64, the pretrained VGG input size.

Figure 8. Spectrogram example, after filtering and resizing, corresponding to a 12 Hz stimulus.



We compared our DNNs to SVMs, a common SSVEP classification method. We created SVMs that were fed with the spectrograms transformed into vectors, with length 24. Furthermore, we also created a SVM that received the fast Fourier transform (FFT) of the time signals (which were preprocessed in the same way as the signals that we analyzed with STFT and used WS with the displacement of 0.1 s). The FFT also utilized the window length of 0.5 s, and afterwards we converted its results to the decibels scale, normalized them between 0 and 1 and maintained frequencies between 0 and 90 Hz.

3.3.3 Specaugment For BCIs

We changed the original SpecAugment technique (PARK et al., 2019) to adapt it for BCIs. Firstly, we did not use time warp, as the method had small benefits in our preliminary tests but had high computational cost. These findings are in accordance with what Park et al., (2019) discovered in their speech recognition studies. Secondly, we abandoned the original randomized and online approaches to create and position time and frequency masks. Our spectrograms are much smaller than speech recognition spectrograms. Hence, to avoid losing too much information, we decided to only use one time mask, of one row, and one frequency mask, of one column. With this decision, we had only 36 masking options for each spectrogram (4 time masks, one for each of the 3 columns and using no mask, and 9 frequency masks, one for each of the 8 rows and using no mask). Therefore, we generated and saved all 36 options for each spectrogram. In this case, not using a random approach allowed us to save training time and avoid repetition (the same image being created more than once).

In this work we compared using SpecAugment to using only time masks and to not using the technique. We also utilized different window displacements in WS. Therefore, we ended up with 5 datasets of varied sizes:

A. SpecAugment and WS with 0.5 s displacement: 146,880 images.
B. Time masking and WS with 0.5 s displacement: 16,320 images.
C. No SpecAugment and WS with 0.5 s displacement: 4,200 images.
D. No SpecAugment and WS with 0.1 s displacement: 19,320 images.
E. Time masking and WS with 0.1 s displacement: 77,280 images.

We did not use SpecAugment with 0.1 s displacement in WS because we noticed, when using the 0.5 s displacement, that time masks and SpecAugment generated very similar results (as will be shown in section 3.5).

3.3.4 SSVEP Dataset Subdivisions

In this study, we simulated a BCI that does not require calibration by the end-user. Therefore, we did not use any data from the subject selected for testing in the training or validation datasets. To divide the dataset, we first separated the test subject's data (creating the test dataset), then we randomly selected 66.6% of the remaining spectrograms for training and 33.3% for validation (maintaining the class balance). We only applied SpecAugment to the training dataset, but WS was applied to the three datasets, with the same parameters (to utilize the same data length, 0.5 s, in training and testing, simulating a fast BCI). Only the data generated by WS was used.

3.4 SSVEP CLASSIFICATION WITH DNNS: METHODOLOGY

In this section we explain the neural networks that we used and their training procedure.

3.4.1 DNN Architecture

Our DNN was created starting from the AudioSet pretrained VGG, we kept its convolutional layers but replaced the fully-connected (FC) ones (the last three layers) by two new, randomly initialized, FC layers. We chose this architecture after preliminary tests (we also tested not removing layers and changing only the last VGG layer), and it is shown in table 2. The first added FC layer has a ReLu activation function, and the last layer is linear, because we trained the DNN with PyTorch's cross-entropy loss, which adds a Softmax activation to the end of the network.

Table 2. The DNN for SSVEP classification.

Convolution, 64 channels
Maxpool
Convolution, 128 channels
Maxpool
Convolution, 256 channels
Convolution, 256 channels
Maxpool
Convolution, 512 channels
Convolution, 512 channels
Maxpool
Dropout, 50%
Fully connected, 512 neurons
Dropout, 50%
Fully connected, 2 neurons

3.4.2 Training and Transfer Learning

After preliminary tests, we noticed better results when not freezing any layer weights while training (we also tested freezing the convolutional block and then fine tuning all layers, at once or unfreezing layer by layer).

We needed much regularization because we were not using the test subject data for training. Regularization was applied in the forms of dropout, weight decay and early stop. We trained one network for each of the 35 subjects as the test subject.

In this study we utilized the Python library PyTorch, with a NVidia RTX 3080 graphics processing unit. Because of early stopping the DNNs did not achieve the maximum number of training epochs, which we will mention below.

With WS and 0.5 s window displacement we tested our DNN using SpecAugment, with only time masks and without the technique. We used stochastic

gradient descent (SGD) to train the networks, with learning rate (lr) of 0.001, momentum of 0.9, weight decay of 0.001, cross-entropy loss and mini-batches of 128 images. With SpecAugment early stop's patience was 50 and the maximum number of epochs 500. With time masks or without SpecAugment, patience was 500 and maximum number of epochs 5000. Training times varied with SpecAugment utilization, utilizing the technique, each epoch took about 30 s and training took about 1,100 s (time until the optimal DNN, according to the hold-out validation loss, was achieved). With time masks, each epoch took about 3.6 s and training 730 s. Finally, without SpecAugment each epoch took about 1.3 s and training 270 s.

With 0.1 s window displacement in WS, we trained a DNN with SpecAugment's time masks and one without SpecAugment. Again, we used SGD, with momentum of 0.9, learning rate of 0.001, weight decay of 0.01, cross-entropy loss, and mini-batches of 128. Without SpecAugment early stop patience was set to 200 and maximum epochs to 5,000. With time masks, 250 and 2,000. Epochs with time masks took about 15 s and the whole training process required about 5,600 s. Without SpecAugment each epoch took about 6 s and training 2,700 s.

Three other SSVEP classification methods were used for comparison: SVMs, FBCCA and our neural network without transfer learning. SVMs are a common SSVEP classification method (e.g., SUTJIADI; PATTIASINA; LIM, 2018 and SETIONO et al., 2018) and a shallow and linear model, making an interesting comparison to a deep network. We decided to use linear kernels in the SVMs, because in other SSVEP works (e.g., SETIONO et al., 2018) we could not observe a substantial improvement when using non-linear SVMs. FBCCA is the technique used by the authors of our dataset, WANG et al., (2017b), for classification and it surpassed CCA performances in a similar database (CHEN et al., 2015).

The DNN without transfer learning was trained in a similar way to the other DNNs (e.g., same loss functions, optimizer parameters and mini-batch size). We did not use SpecAugment, the maximum number of epochs was 5000, early stop patience was 2000 and WS displacement was 0.5 s. We noticed that these networks took a long time to effectively start reducing the training loss, and it could stay in a high plateau for hundreds of epochs. Therefore, we used a large value for early stop patience. Each epoch took about 1.3 s and training 310 s.

We trained the linear SVMs using spectrograms, with SGD, momentum of 0.9, lr of 0.01, hinge loss, regularization parameter (c) of 0.01, early stop patience of 200, maximum of 5000 epochs, mini-batches of 128 images and WS window displacement of 0.5 s. Without SpecAugment each epoch took about 0.5 s and training time was about 100 s. With SpecAugment each epoch took about 4.3 s and the mean training time was 660 s.

We also trained a linear SVM with the fast Fourier transform of the time signals. Its training parameters were: SGD, momentum of 0.9, lr of 0.001, hinge loss with c=0, early stop with patience of 200 and maximum of 3000 epochs, mini-batches of 16 samples (with larger mini-batches the models took a long time to start decreasing the training loss) and WS with window displacement of 0.1 s. Each epoch took about 0.5 s, and the mean training time was 500 s.

Finally, we classified the dataset with FBCCA. We used the same parameters that the dataset authors used (WANG et al., 2017b) to classify the database: M3 filtering method, Nh=5, a=1.25, b=0.25, N=7. (CHEN et al., 2015) explains these parameters in more detail. With FBCCA we utilized the WS displacement of 0.5 s. We classified only the Oz electrode data, as we did in the rest of our study, but we also classified the data of a combination of 9 electrodes (Pz, PO5, PO3, POz, PO4, PO6, O1, Oz, and O2), as was done by the authors of (CHEN et al., 2015) and (WANG et al., 2017). Our implementation of FBCCA was based on the one available at Alu (2019), which uses Python.

3.5 SSVEP CLASSIFICATION WITH DNNS: RESULTS AND DISCUSSION

We present, in table 3, the test accuracies (in percentage) of our DNNs trained with transfer learning. The columns indicate the different networks and the rows the test subjects. In this table the first three DNNs were trained using WS with 0.5 s window displacement, and the last two using 0.1 s displacement. In the last row of the table, we display the mean test F1-Scores.

Table 4 shows the alternative classification methods (SVM, FBCCA and DNN without SpecAugment) test accuracies. Every method in the table used a window displacement of 0.5 s, except for the SVM with FFT, which used 0.1 s. Except for the last column, all methods used only the Oz electrode data (as in table 3). In this table

we only used SpecAugment in one SVM (column 4). Again, in the last row of the table we display the mean test F1-Scores.

Table 3. Test accuracies and F1-Scores (last row) for networks trained with transfer learning.

Test subject	DCNN with SpecAugment	DCNN without SpecAugment	DCNN with time masks	DCNN with 0.1 s displacement (no SpecAugment)	DCNN with 0.1 s displacement and time masks
1	75.8	72.5	69.2	74.5	74.6
2	78.3	76.7	79.2	80.4	81.7
3	90.8	90.8	90.8	93.5	93.7
4	71.7	70.8	75	72.6	72.6
5	91.7	91.7	91.7	91.7	91.7
6	89.2	90	89.2	90.9	90.9
7	79.2	80	79.2	83.2	84.4
8	93.3	92.5	91.7	92	92.2
9	74.2	73.3	75	78.3	78.4
10	87.5	85	85.8	88.8	88.8
11	66.7	64.2	64.2	63.9	63.2
12	65.8	64.2	66.7	69.7	70.8
13	55.8	56.7	57.5	60	60.9
14	70	68.3	70.8	68.8	69.6
15	87.5	88.3	91.7	88.4	88
16	70	70.8	69.2	75.5	76.3
17	75.8	76.7	75.8	78.4	79.2
18	75.8	77.5	78.3	76.3	76.8
19	75	72.5	74.2	71.7	72.5
20	90.8	90	90.8	89.9	91.3
21	72.5	72.5	69.2	74.8	74.3
22	98.3	96.7	96.7	96.6	96.4
23	74.2	66.7	70	67.4	67.8
24	92.5	92.5	93.3	91.8	91.7
25	86.7	84.2	85.8	84.8	85.3
26	91.7	90.8	90	92.2	92.4
27	89.2	92.5	93.3	92	92.6
28	86.7	87.5	88.3	93.1	93.3
29	60	54.2	57.5	57.6	58.7
30	93.3	94.2	94.2	94.7	93.7
31	89.2	91.7	90.8	91.7	92
32	95	95.8	95	97.6	97.8
33	70	69.2	71.7	72.6	73.4
34	78.3	80.8	80	76.4	77.9
35	86.7	86.7	88.3	89.7	90.8
Mean accuracy	80.8	80.2	80.8	81.8	82.2
Mean F1-Score	0.814	0.806	0.814	0.819	0.825

Table 4. Test accuracies and F1-Scores (last row) for alternative SSVEP classification methods.

Test Subject	SVM	SVM with FFT	SVM with SpecAugment	DCNN without transfer learning	FBCCA	FBCCA with 9 electrodes
1	72.5	80.8	69.2	74.2	75	93.3
2	80	83.2	79.2	78.3	87.5	96.7
3	89.2	93.8	90	90.8	93.3	95.8
4	65.8	69.6	67.5	72.5	68.3	95.8
5	90	87.3	90.8	90	90.8	95.8
6	87.5	80.3	85	88.3	86.7	94.2
7	79.2	77.2	78.3	80.8	77.5	92.5
8	90.8	91.1	90	89.2	85.8	90.8
9	76.7	81.5	76.7	70	70.8	83.3
10	84.2	83.5	83.3	85.8	91.7	94.2
11	62.5	64.3	61.7	60.8	60	85
12	63.3	56.9	64.2	62.5	65	95
13	53.3	58.3	55	57.5	55	95
14	61.7	62.3	64.2	72.5	68.3	95
15	85	86.8	85	89.2	80	87.5
16	67.5	72.8	66.7	70.8	71.7	80.8
17	77.5	71.4	75.8	75.8	71.7	82.5
18	75.8	73.4	76.7	76.7	70.8	90.8
19	69.2	56.2	69.2	71.7	63.3	69.2
20	90	81.3	88.3	89.2	87.5	84.2
21	72.5	75.7	71.7	75	70.8	85.8
22	96.7	86.2	95.8	97.5	88.3	95
23	70	65.0	70	70	67.5	91.7
24	83.3	89.9	82.5	94.2	90.8	96.7
25	83.3	72.3	81.7	82.5	71.7	99.2
26	89.2	87.3	86.7	91.7	71.7	95.8
27	90.8	92.6	90.8	90	92.5	93.3
28	89.2	90.4	88.3	87.5	88.3	95.8
29	59.2	57.6	60	59.2	57.5	81.7
30	91.7	92.4	92.5	94.2	89.2	94.2
31	88.3	86.1	87.5	91.7	82.5	98.3
32	95.8	96.9	95	95	95.8	93.3
33	66.7	73.0	68.3	70.8	68.3	81.7
34	80.8	77.4	82.5	80	65.8	96.7
35	86.7	88.8	89.2	85.8	76.7	91.7
Mean accuracy	79	78.4	78.8	80.3	77.1	91.1
Mean F1-Score	0.793	0.75	0.794	0.805	0.754	0.908

We observe that the DNN that used WS with displacement of 0.5 s and SpecAugment could not surpass the DNN that used just WS with displacement of 0.1 s (even though the dataset sizes were similar). But we note that the two augmentation methods can be successfully combined, and the DNN with WS displacement of 0.1 s and SpecAugment achieved our best mean accuracy and F1-Score (with a single electrode).

When we analyze, in table 3, the DNNs that used 0.5 s window displacement, we observe the same mean test accuracy in the DNN with SpecAugment and the one with just time masks, and a lower accuracy in the network without SpecAugment. In

table 4 we see that the effect of SpecAugment on the SVM was much smaller than that observed with DNNs, as was expected in a shallow and linear model. Moreover, none of the SVMs could surpass the VGGish DNNs.

Observing table 4, we see that FBCCA's accuracy was worse than the neural networks' when also using only one electrode. However, its results are much better when using 9 electrodes (Pz, PO5, PO3, POz, PO4, PO6, O1, Oz, and O2). This large increase in performance is expected, as FBCCA is a multi-channel classification model (it combines data from all electrodes in its input), created for the multi-channel case, unlike our proposed DNNs, which take a single spectrogram as input. In a previous study (BASSI; RAMPAZZO; ATTUX, 2019) we utilized data from many electrodes with DNNs, but the DNNs there were also single-channel, analyzing the electrodes individually, and they performed better with just the Oz electrode data. We also note that training our DNNs with frozen weights did not improve results.

Finally, we should say that, when comparing our results to other studies, it is important to observe if they also did not use the test subject's data for training, as this approach can make the classification problem more challenging, decreasing accuracy, but creating a calibration-less BCI.

3.6 SSVEP CLASSIFICATION WITH DNNS: CONCLUSION

This study showed us that deep neural networks can improve the accuracy in a fast (with small data length, 0.5 s) single-channel SSVEP based BCI (in comparison to FBCCA and SVMs), even without training in the test subject. Using SpecAugment's time masks, WS with 0.1 s window displacement and transfer learning we achieved a mean accuracy 3.2% higher than the SVM's and 5.1% higher than FBCCA's. The single-electrode approach is useful to create lighter, more practical, smaller and less expensive devices, like the 3D printed headset in (NGUYEN; CHUNG, 2019).

The classification method proposed here is a single-channel approach, and even though this method has its benefits, better accuracies can be achieved with FBCCA and multiple electrodes. In a future study we wish to modify the methods proposed here to also explore them in a multi-channel BCI.

In this study, transfer learning had a very small effect on performances (increasing mean F1-Score by 0.001 and decreasing mean accuracy by 0.1%). However, it made training about 15% faster and more predictable, as, without it, the

DNNs can stay, for hundreds of epochs, in a plateau of high training loss during the beginning of the training process. The small performance change and the ineffectiveness of freezing layers may indicate that audio classification is not similar enough to SSVEP classification to fully take advantage of transfer learning. In this case, pretraining on a large SSVEP dataset could improve performances.

SpecAugment showed a positive result, increasing mean accuracy by 0.6% and mean F1-Score by 0.008 (in the DNNs with 0.5 s WS window displacement). But not using SpecAugment and utilizing a smaller displacement, of 0.1 s, in WS surpassed SpecAugment's accuracies (with 0.5 s displacement) by 1%. However, both methods were successfully combined, and the DNN with time masks and 0.1 s WS window displacement showed our best single-electrode results, with 82.2% mean accuracy and 0.825 mean F1-Score. Therefore, SpecAugment can be combined with other augmentation methods and improve their performances. As expected, SpecAugment has a very small effect in the shallow and linear SVM, actually reducing its accuracy by 0.2%.

The utilization of SpecAugment's time and frequency masks or just time masks provided almost the same accuracies. We theorize that the main reason for this is the small size of our spectrograms. Therefore, using more masks may remove too much information from the image, and frequency information is very important for our classification task.

This study shows the potential of DNNs in BCI-SSVEP systems, and we intend to expand these results to the multi-channel case. Furthermore, we think that the creation of larger and open SSVEP datasets can improve DNN's results even more, allowing better generalization with more effective transfer learning. This part of our work can also be seen in our paper (BASSI; RAMPAZZO; ATTUX, 2021).

4 COVID-19 DETECTION WITH DNNS

In this section we explain the first part of our work with COVID-19 detection using chest X-rays and deep neural networks.

4.1 RELATED WORK

The field of COVID-19 detection using X-rays and DNNs is quickly evolving, and there are many reviews which are tracking the new papers, like (SHOEBI et al., 2020). It is complicated to directly compare the results of these studies, as many different datasets are used, with different classification methods and different probabilities of bias. Therefore, we compared our proposed classification methods (like DNN with twice transfer learning and ONK) with alternative methods (like DNN without transfer learning) that we implemented ourselves, in the same dataset.

Our work with COVID-19 detection started in January 2020, when there were only very few articles about the topic. Our paper (BASSI; ATTUX, 2020), covering our study of twice transfer learning and ONK for COVID-19 detection, was submitted still in the first semester of 2020.

The second part of our work with X-rays, which analyzed the effect of segmentation in the DNN generalization capability, started in the second semester of 2020, and the paper covering it was submitted in 2021 (BASSI; ATTUX, 2021).

Nonetheless, for context, we will quickly talk about another paper in the context of COVID-19 detection with X-rays, (WANG et al., 2020). It is contemporary to our first article (BASSI; ATTUX, 2020), as it was first published (as a preprint) in April of 2020, 1 month before the first publication of our first preprint. We also chose to talk about this paper because it is one of the most influential early COVID-19 articles (it currently has 700 citations and its final version was published by Nature, in November of 2020). In the study, the authors proposed a convolutional DNN architecture, created with a machine-driven design exploration. As in our study, they also trained their DNNs with a mixed dataset that contained a relatively low number of COVID-19 images (less than 400), but they used a larger number of pneumonia and normal images. They created a test dataset in a random manner, as we did in (BASSI; ATTUX, 2020), and obtained a high test accuracy, 93.3%. Furthermore, their data augmentation technique in the

COVID-19 dataset was very similar to ours, as they also employed rotation, translation and horizontal flipping, but they used zoom and intensity shift as well.

4.2 DATA ANALYSIS

This section explains the datasets that we used, the image preprocessing steps and data augmentation.

4.2.1 The Databases

ChestX-ray14 is a very large chest X-ray dataset, consisting of 112,120 images from 30,805 patients, obtained from the US National Institutes of Health (NIH). It involves 14 different pulmonary diseases, along with healthy patients. The dataset is multi-label, as a single patient can present more than one condition. Labels were originally created with natural language processing, analyzing radiological reports. The labels have an estimated accuracy above 90% (Wang et al., 2017a). The dataset is unbalanced. State-of-the-art deep neural networks were trained in it, such as CheXNet (RAJPURKAR et al., 2017), a DenseNet with 121 layers. This dataset was used in the first step of our twice transfer learning approach.

In the last twice transfer learning step we utilized a database assembled by us, which we will call the COVID-19 database. It is formed by images from three databases: “Covid-19 image data collection” (COHEN et al., 2020), containing COVID-19 images, CheXpert (IRVIN et al., 2019), containing pneumonia images and Montgomery and Shenzen databases (JAEGER at al., 2014), containing normal images.

From “Covid-19 image data collection” (COHEN et al., 2020) we obtained 439 X-rays. These images are all COVID-19 frontal Chest X-rays, removing images with arrows. This database was the largest COVID-19 dataset we could find until October 2020. Furthermore, it is one of the best documented databases, as many images contain information of patient gender and age, image source and disease severity. COHEN et al., (2020) collected the images from public sources or indirectly from hospitals and physicians.

The CheXpert database contains 224316 chest X-rays, from 65240 patients, showing 13 different lung diseases or no findings (IRVIN et al., 2019). From the dataset

we obtained 1255 pneumonia images, selecting only patients that were older than 18 years old (to avoid bias, as the youngest patient with COVID-19 in our dataset was 20 years old). Except for 8 images, all of the other pneumonia X-rays were classified with natural language processing, like in NIH ChestX-ray14, and labels' accuracy is also estimated to be over 90%. The 8 exceptions were manually labeled by three board-certified radiologists (these images are part of the original CheXpert test dataset). All CheXpert X-rays were obtained from the Stanford University Hospital (USA).

The Montgomery and Shenzen databases (JAEGER at al., 2014) contain 370 X-rays of healthy adults, which we used. The database also has tuberculosis images, but they were not used in this part of our study. All images were obtained from the Department of Health and Human Services, Montgomery County, Maryland, USA and Shenzhen No. 3 People's Hospital in China (JAEGER at al., 2014).

4.2.2 COVID-19 Database Details

There are 370 healthy patients in our COVID-19 database, as the dataset from (JAEGER at al., 2014) does not contain multiple images from a single patient. 61.9% of them are male and the mean age is 36.1 years, with 12.3 years of standard deviation.

There are 1047 pneumonia patients in our COVID-19 dataset, 58.2% male and with mean age of 61.8 years and an age standard deviation of 19.3 years.

We have 268 COVID-19 patients in the dataset; 246 of them contained gender information and were 64.2% male; 199 had age information, with a mean age of 42.8 years and 16.4 years of standard deviation; 81 of the patients had survival information, 81.5% of them survived; 69 patients had information about intubation, 59.4% of them were intubated; 94 patients had information about supplemental oxygen, and 59.6% needed it. Finally, 112 patients had information about ICU and 59.8% of them were treated in the intensive care unit.

4.2.3 Data Processing And Augmentation

We designed our data processing and augmentation strategies for the ChestX-ray14 dataset following the procedure described in Rajpurkar et al., (2017), as they also trained a DenseNet (CheXNet) in this dataset.

Firstly, we resized the images to 224x224x3, the ImageNet pretrained DenseNets input size. We then normalized the images, with the means and standard deviation of their ImageNet pretraining. We utilized the ChestX-ray14 suggested test dataset and hold-out validation, randomly splitting the remaining images, with 80% for training and 20% for validation. Different datasets (training, testing and validation) had no images from the same patient. We utilized 15 labels, one for each disease and one for healthy.

As data augmentation, we only utilized image flipping, horizontally, with 50% chance. The augmentation was performed online, and new images substituted the old ones in the mini-batch.

We will now specify the data processing and augmentation processes in the COVID-19 database. Firstly, we divided it into three parts, for training, testing and validation (hold-out). The test dataset has 50 randomly selected images of each class (normal, pneumonia and COVID-19). We splitted the remaining images with 90% for training and 10% for validation, maintaining in both sets the same class proportions. Again, two distinct sets do not have images from the same patient. We started by loading the X-rays in greyscale, reshaped them to 224x224x3, where 3 meaning that the grayscale image was converted to the BGR colors pace, still using only shades of grey. Images were converted to BGR because it is the color space that had been used to train the DNNs on ImageNet. After reshaping, we normalized the images with the same mean and standard deviation from the ImageNet pretraining. The 8 manually labeled CheXNet pneumonia images are in our test dataset.

Many images in the COVID-19 dataset contained words or letters, which sometimes were exclusive for some classes. For example, the Italian word “SEDUTO” (seated) is present in many COVID-19 images and does not appear in any other class. Fearing that this could be a source of bias, we removed the words in the test dataset, manually covering them with a black rectangle (this process was not automated because there were words in different locations in the X-rays). As they were not over the lungs, no important information was lost. This technique was only used in the test

dataset, as we feared that using it for training could teach the DNNs to identify the rectangles, creating a potentially bigger source of bias.

The COVID-19 dataset is small, so we decided to explore data augmentation in a more extensive manner. Furthermore, data augmentation was used to balance our datasets. For this reason, we applied it to the training and validation datasets. Three methods were used: image rotation (between -40 and 40 degrees), translation (up to 28 pixels left and right or up and down) and flipping (horizontal, 50% chance). The techniques could augment our data and also make our DNNs more resistant to rotations and translations, and they are common in the context of image classification. Furthermore, they were also used in other papers about COVID-19 detection with X-rays, such as in (WANG; LIN; WONG, 2020). The augmentation was online and new images were added to the mini-batch, along with the original X-rays. To balance the databases the normal class was augmented 30 times, pneumonia 8 times and COVID-19 24 times. Therefore, we had 8,280 COVID-19 images and 8,640 images of normal and pneumonia. At each epoch, 360 of the augmented pneumonia and normal images were left out of training and the DNN was fed a completely balanced dataset of 16,560 images. Every epoch we randomly selected the images to be left out, so every X-ray was used during training.

4.3 COVID-19 DETECTION WITH DNNS: METHODOLOGY

In this section we explain our proposed technique, output neuron keeping, the DNNs that we used, their training procedures and our implementation of LRP.

4.3.1 Output Neuron Keeping

In this part of our study, we proposed an original modification in the twice transfer learning method, which we called output neuron keeping (ONK).

When we perform three-step transfer learning we search for a third task that is very similar to the second. Thus, it is possible for the second and third datasets to share classes. For example, ChestX-ray14 and the COVID-19 dataset share two classes, healthy and pneumonia.

In this case we suggest keeping the output neurons that classify the shared classes, i.e., not changing their learned weights and biases after the second transfer

learning step, and just replacing the other output neurons. Doing so, the representations learned by these neurons in the second step will be maintained and this may improve training speed or performance in the final task.

To keep the neurons, using PyTorch, we can begin by copying their learned parameters after the second step of twice transfer learning. Then, in the beginning of the third step, we can replace the DNN output layer for one with the correct number of neurons, using randomly initialized parameters, find the neurons that will classify the shared classes and substitute their parameter for the ones previously copied.

4.3.2 The DNNs For COVID-19 Detection

In this part of our study, we trained 5 DNNs, which we shall call A, B, C, D and E. All of them are dense neural networks. The reason for this was CheXNet's (a DenseNet121) excellent result classifying ChestX-ray14, even surpassing 4 radiologists in pneumonia detection (RAJPURKAR et al., 2017).

DNN A is a DenseNet with 201 layers, downloaded pretrained on ImageNet, and trained on the COVID-19 dataset. Therefore, here we applied a simple transfer learning.

DNN B is also a 201 layers DenseNet pretrained on ImageNet, but we trained it on ChestX-ray14 and then on the COVID-19 dataset. Therefore, we trained it with twice transfer learning (even though the first step, on ImageNet, was not performed by us).

DNN C is the same as DNN B but exploring the output neuron keeping. After training on ChestX-ray14, we kept the neuron that classified normal and pneumonia. The neuron that would classify COVID-19 was randomly initialized in the beginning of the third step.

DNN D is a dense neural network with 121 layers. We downloaded a pretrained CheXNet, i.e., a DenseNet121 trained on ImageNet and then on ChestX-ray14 and trained it on the COVID-19 dataset. Therefore, it was trained with twice transfer learning.

Finally, DNN E is the same as DNN D, but with output neuron keeping. From the downloaded CheXNet output layer we maintained the neuron that classified pneumonia. Only one neuron was kept, because the CheXNet did not have a neuron to classify the healthy class.

The five DNNs are summarized in table 5.

Table 5. DNNs for COVID-19 detection.

Name	DNN architecture	Training process
A	201-layers DenseNet	Transfer learning
B	201-layers DenseNet	Twice transfer learning
C	201-layers DenseNet	Twice transfer learning + output neuron keeping
D	121-layers DenseNet (CheXNet)	Twice transfer learning
E	121-layers DenseNet (CheXNet)	Twice transfer learning + output neuron keeping

4.3.3 Twice Transfer Learning: ChestX-ray14

The DNNs B and C are the only DNNs that we will need to train on ChestX-ray14 ourselves. But they have the same structure and training procedure in this dataset. Therefore, we trained a single DenseNet201 on ChestX-ray14, which would generate DNNs B and C.

We began by downloading the official PyTorch version of the ImageNet pretrained DenseNet201. We then substituted its output layer by one with 15 neurons (for the 14 diseases and normal/"no findings" class) and sigmoid activation.

We trained the DNN in PyTorch, using binary cross-entropy loss, stochastic gradient descent with momentum of 0.9, mini-batches of 16 images and hold-out validation. Two NVidia GTX 1080 GPUs were used. Firstly, we trained only the output layer, for 20 epochs, with a learning rate of 0.001. We then trained the whole model, with lr of 0.0001, for 90 epochs. By the end of this process the network was already overfitting.

4.3.4 Twice Transfer Learning: COVID-19 Dataset

All the five networks were trained on the COVID-19 dataset. For DNN A, we began with an ImageNet pretrained PyTorch version of the DenseNet201 and we substituted its output layer for one with 3 neurons and softmax activation (because this dataset is single label), randomly initialized. DNN B started as the network we trained on ChestX-ray14, and we also substituted its last layer for one similar to DNN A's. DNN C also started as the DNN we trained on ChestX-ray14, and we also substituted its output layer, but we kept the neurons that classify normal and pneumonia.

To create DNNs D and E, we started by downloading a pretrained CheXNet (on ImageNet and ChestX-ray14) from (ZECH, 2018) and we substituted its final layer for one with 3 neurons and softmax activation. For DNN D, they were randomly initialized. For DNN B we kept the parameters of the neuron that classified pneumonia and randomly initialized the other two, performing output neuron keeping. All five DNNs have similar architectures and we want to compare them during and after the training process on the COVID-19 dataset. Therefore, they were trained in the same way. We used PyTorch, cross-entropy loss, stochastic gradient descent with momentum of 0.9, mini-batches of 9 images and hold-out validation. We again used two NVidia GTX 1080 GPUs. We determined most training hyperparameters with preliminary tests, e.g., we discovered that utilizing weight decay in the beginning of the training process, to avoid fast overfitting, and removing it in the end, when training loss stagnated, improved accuracy.

The training procedure had 4 phases. In the first we trained only the output layer, for 10 epochs, with a learning rate of 0.001 and weight decay of 0.01. Afterwards, we trained the entire network for 48 epochs, with early stop and patience of 20, the same weight decay, and differential learning rate (it started as 0.0001 in the last layer and decreased by a factor of 10 for each previous dense block and its transition layer). We then set the learning rate to 0.00001 in every layer and trained for 48 epochs more, with the same parameters of early stop and weight decay. Finally, we removed the weight decay and trained for 48 epochs more.

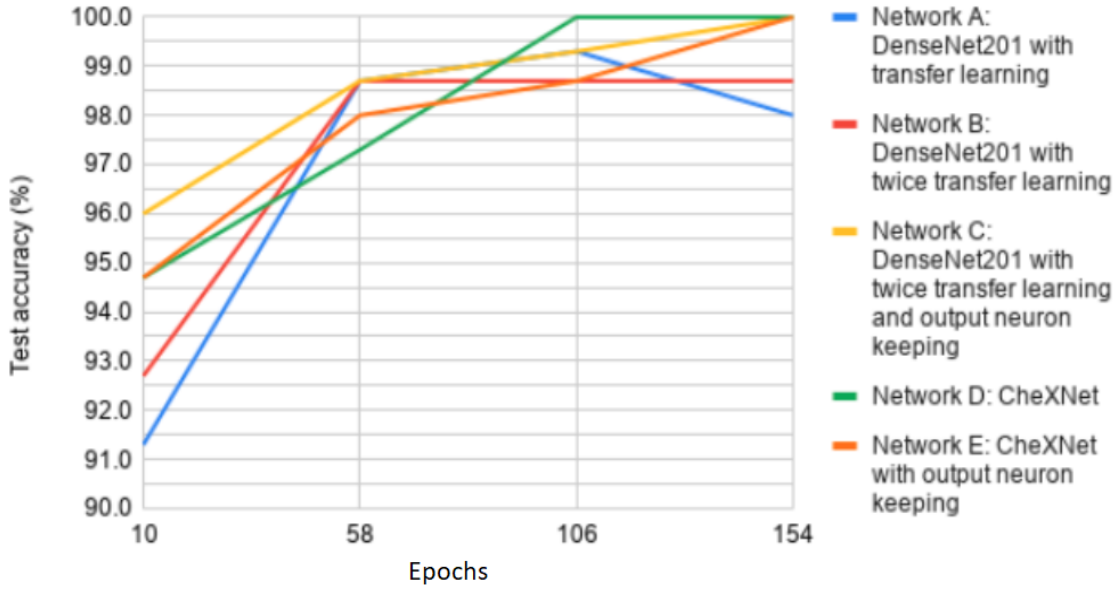
4.3.5 Applying LRP

We applied LRP with the iNNvestigate (ALBER et al., 2019) Python library. As it works with models trained in Keras (another Python library for deep learning) and not in PyTorch, we needed to convert our networks. For this purpose, we used the py2keras library (MALIVENKO, 2018). We tested different LRP presets, and chose INNvestigate's "LRP-PresetAFlat", because it produced more interpretable and coherent heatmaps.

4.4 COVID-19 DETECTION WITH DNNS: RESULTS AND DISCUSSION

In figure 9, we show the test accuracy during training on the COVID-19 dataset, for all five DNNs. These values are the losses for the best networks, according to validation loss, in all the four training phases described in section 4.3.4, which end in epochs 10, 58, 106 and 154. Both CheXNets and the DenseNet201 with twice transfer learning and output neuron keeping achieved the maximum accuracy, of 100%.

Figure 9. Test accuracy during training on the COVID-19 dataset.



Tables 6 and 7 show DNNs A and B confusion matrices (the DNNs that did not have 100% accuracy), created with the test dataset. DNN B made two mistakes, classifying COVID-19 as pneumonia, and DNN A made one mistake, classifying normal as COVID-19.

Table 6. DNN A confusion matrix.

		Predicted class		
		Normal	Pneumonia	COVID-19
Real Class	Normal	49	0	1
	Pneumonia	0	50	0
	COVID-19	0	0	50

Table 7. DNN B confusion matrix.

		Predicted class		
		Normal	Pneumonia	COVID-19
Real Class	Normal	50	0	0
	Pneumonia	0	50	0
	COVID-19	0	2	48

We will start our analysis by comparing the DenseNet201s (DNNs A, B and C) in figure 9. In the first epochs, we can clearly observe a higher accuracy with DNN C (twice transfer learning and ONK), followed by DNN B (twice transfer learning) and then by DNN A (simple transfer learning). During the training process their performance differences decreased and, at the end, DNN C had 100% accuracy, DNN A 99.3% and DNN B 98.7%.

We will now compare the CheXNets (DNNs D and E) in figure 9. Their accuracies were very similar: in fact, we can notice that they started and ended at the same values. Our idea is that ONK with a single neuron had a smaller effect, as differences were larger in the DenseNet201s. We can also observe that the CheXNets's starting accuracies were higher than those of DNNs A and B, but lower than DNN C, which used ONK with two neurons.

If we look at the state-of-the-art in COVID-19 detection using deep learning we see that most studies achieved accuracies in the 90% to 100% range (SHOEBI et al., 2020). Therefore, this work is on par with the state-of-the-art.

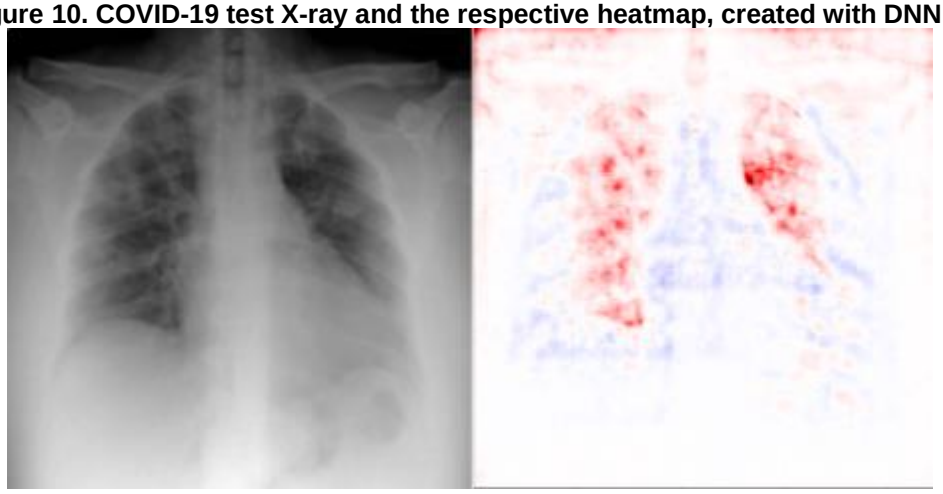
But, although we achieved test accuracies of 100% and employed patient split, we must note that our test database had only 150 images, and that the COVID-19 dataset is small and mixed (i.e., images from different classes come from different sources). These dataset limitations decrease the statistical significance of our performance metrics, and possibly reduces generalization capability of our DNNs. These aspects will be explored in detail in section 5 of this dissertation.

4.4.1 Analysis Using LRP And Words On X-Rays

In figure 10, we show an example of a correctly classified COVID-19 X-ray test image and its heatmap, created with DNN C. The redder the region on the map, the more important it was for the DNN classification as COVID-19. The bluer, the more that region is related to other classes (i.e., somewhere the DNN found healthy or where

it found pneumonia signs). In the heatmap we see the DNN encountered signs of COVID-19 in both lungs.

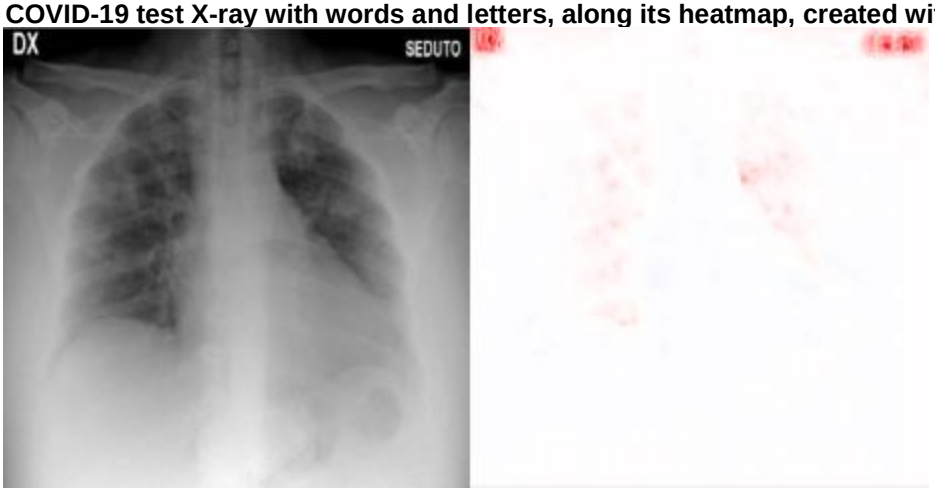
Figure 10. COVID-19 test X-ray and the respective heatmap, created with DNN C.



LRP showed us that our black rectangles, which cover words in the test dataset, can generate artifacts (lines of relevance) around them, but the effect is small. Also, as we only utilized them for testing and with all classes, we do not think that this can affect our accuracy or create any bias.

LRP also allowed us to analyze the effect of words on the X-rays and we feared they could be a source of bias if the DNN was used in a dataset containing them. In figure 11 we analyze the same X-ray shown in figure 10, but with the words “SEDUTO” and the letters “DX”, both originally present in the image. The red colors in the word locations in the heatmap makes clear that they were associated with the COVID-19 class.

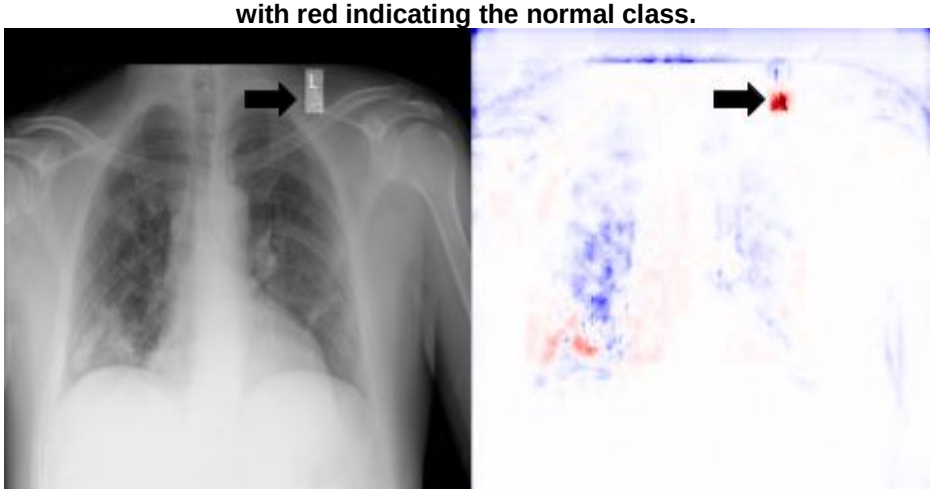
Figure 11. COVID-19 test X-ray with words and letters, along its heatmap, created with DNN C.



To understand the magnitude of this effect we tested our DNNs with the unedited test dataset (showing all words). We observed only small accuracy changes. They decreased or increased by 1.33% at most on any fully trained DNN. Our best networks, C, D and E, maintained their 100% accuracy. DNNs in early training stages had accuracies changing by up to 3.33%.

The last LRP test to understand the words' effects was trying to “fool” our DNNs. We added the letters “L PA”, copied from a healthy image (and present in many of the X-rays in this class), to a COVID-19 X-ray. The network still classified the X-rays as COVID-19, the disease’s predicted probability just decreased from 99.94% to 99.91%, while the normal probability increased from 0.036% to 0.055%. Figure 12 shows the edited image and its heatmap, created with DNN C. This relevance propagation started on the output neuron that classifies the normal class, thus, red colors indicate normality and blue colors indicate diseases. As the PA letters are red, they influenced the DNN to choose the normal class. The letter L, above PA, is blue. We believe that it was not associated with the normal class because it was common in other classes too. Although the heatmap suggests an interference, we see that, in this case, the letters influence in the DNN output was minimal.

Figure 12. COVID-19 X-ray edited with letters “L PA”, from a normal X-ray, and its heatmap, with red indicating the normal class.



4.5 COVID-19 DETECTION WITH DNNS: CONCLUSION

In dense neural networks with 201 layers, the proposed method of output neuron keeping surpassed the accuracies of just twice transfer learning and simple transfer learning. With this in mind, and also observing the great results that twice

transfer learning had in (CAI; LIU; GUO, 2018), we think that the technique and our proposed method, ONK, are promising and could improve performances in other classification problems.

CheXNets kept up with and even surpassed most of the DenseNet201 DNNs. We think that the main reason for this is that the larger network is more susceptible to overfitting in the small COVID-19 dataset, even with the use of regularization and data augmentation.

LRP was effective in showing what most captured the attention of the networks and affected their decisions. We hope that this indicates a possible tool to help radiologists, allowing them to better understand the decision provided by a machine learning model, and provides a better interaction between specialists and the neural networks.

LRP also allowed us to analyze how words affect the neural networks, showing that words and letters do indeed influence them. However, small accuracy changes when testing the DNNs on a dataset containing or not words (1.33% at most in fully trained DNNs, 3.33% at most in DNNs trained for a small number of epochs) indicate that such influence is small.

Even though we explored the largest and most well documented COVID-19 dataset we could find in October 2020, it only has 439 coronavirus X-rays. Furthermore, images from other classes needed to be gathered from other datasets, making the dataset small and mixed. This presents challenges to generalization capability and the small test dataset size decreases the statistical significance of our measured accuracy. These aspects will be explored in section 5 of this dissertation. But we can already say that a very large, open and high-quality database, with all classes collected from the same sources, would be extremely beneficial for the field of COVID-19 detection with neural networks. Furthermore, clinical studies are needed to ensure real-world performance of COVID-19 detection methods.

Even with these limitations, this study and other initiatives (SHOEIBI et al., 2020) indicate that deep neural networks have the potential of making chest X-ray a fast, accurate, cheap and easily available auxiliary method for COVID-19 diagnosis. The models we trained here are available for download at (BASSI; ATTUX, 2020) and this study can also be seen in our paper (BASSI; ATTUX, 2021b).

5 COVID-19 DETECTION WITH LUNG SEGMENTATION

This section explains the last part of our work, which involved COVID-19 detection with lung segmentation and an analysis of the generalization capability of DNNs trained in small and mixed COVID-19 datasets.

5.1 DATA ANALYSIS

This section explains our datasets, data processing stages and data augmentation.

5.1.1 The Source Databases

In this part of our work, we used the same source datasets as in section 4, namely NIH ChestX-Ray14, the Montgomery and Shenzen databases, CheXpert and the COVID-19 image data collection. We utilized these databases to create three datasets, a segmentation dataset, a classification training dataset and external classification testing and validation datasets (for holdout validation).

925 images, showing healthy patients, were extracted from the NIH ChestX-ray14 dataset (WANG et al., 2017a) and used in our classification training dataset. Those images correspond to 925 different patients, with mean age of 46.8 years (with 15.6 years of standard deviation), being 54.3% of them male. Additionally, 1295 ChestX-Ray14 images, showing patients with pneumonia, were also included in our classification training dataset. They correspond to 941 patients, with a mean age of 48 years (standard deviation of 15.5 years), and 58.7% of them are male.

The Montgomery images came with segmentation masks, created under the supervision of a radiologist (JAEGER at al., 2014). The dataset authors segmented the images excluding the lung part behind the heart, and following some anatomical landmarks, such as the ribs, the heart boundary, aortic arc, pericardium line and diaphragm (JAEGER at al., 2014). Stirenko at al., (2018) created segmentation masks for most of the Shenzen database. They are similar to the Montgomery's (e.g., they also exclude the lung part behind the heart). The healthy patients in the Montgomery and Shenzen database have a mean age of 36.1 years (with standard deviation of 12.3

years) and are 61.9% male. Their X-rays were used in our classification training dataset.

From the COVID-19 image data collection (COHEN et al., 2020), we obtained 475 COVID-19 X-rays (all the frontal COVID-19 X-rays). The images we utilized correspond to 295 COVID-19 patients, with a mean age of 42.5 years (with standard deviation of 16.5 years), having 64.5% male of them as male. We have information about disease severity on some of them: from 87 patients, 79.3% survived; from 118 patients, 61.9% needed ICU; from 77 patients, 61% were intubated; from 107 patients, 62.6% needed supplemental oxygen.

We used part of the CheXPert (IRVIN et al., 2019) database in our classification dataset, as part of the external validation. 79 pneumonia and 79 healthy images were used, including the ones manually labeled by three radiologists. The normal images are associated with 73 patients, with a mean age of 49.5 years (with standard deviation of 18.5 years), being 56.2% male. The pneumonia images come from 61 patients, with mean age of 61.9 years (standard deviation of 18.1 years), and who are 60.7% male.

We did not utilize images from patients under 18 years old to avoid bias, because the youngest COVID-19 patient in our database is 20 years old. Again, only frontal X-rays were used.

5.1.2 The Segmentation Dataset

We explored the segmentation dataset to train our segmentation module, an U-Net, to segment lung regions in chest X-rays. The dataset contains 327 COVID-19 images, 327 pneumonia images, 327 normal images and 282 tuberculosis images.

The normal and tuberculosis images are all the X-rays with segmentation masks in the Montgomery and Shenzhen database. Pneumonia and COVID-19 images were randomly selected among the classification dataset images (thus, they come from NIH ChestX-Ray14, and the COVID-19 image data collection, respectively).

The dataset also has a segmentation mask for each of its X-rays. The masks for the healthy and tuberculosis classes were created by the authors of the Montgomery database and by Stirenko at al., (2018). The pneumonia and COVID-19 masks were created by us, as will be described in section 5.2.2.

The segmentation database was divided in three parts, for training, validation (hold-out) and testing. The process was random, but we kept class balance and utilized a patient split (two parts of the dataset could not contain images from the same patient).

The segmentation test dataset has 150 images, 50 normal, 50 pneumonia and 50 COVID-19. Tuberculosis images were not included here, because, although we thought that they could improve our network training process, the U-Net performance with them was not relevant, as this class is not present in the classification dataset. After creating the test dataset, we divided the remaining images, selecting 80% of them as training and 20% as validation.

5.1.3 The Classification Training Dataset

We used the classification training dataset to train our classifiers, to identify COVID-19, pneumonia and normal. It has 1295 pneumonia X-rays, 1295 healthy X-rays and 396 COVID-19 X-rays. Unlike the segmentation dataset, this one only has simple labels, identifying the image class.

The COVID-19 images are all frontal X-rays showing the disease in the database from (COHEN et al., 2020), except for the ones from Hannover Medical School (Hannover, Germany), as they will be used in the external classification database.

Pneumonia images were obtained from NIH ChestX-Ray14, and healthy images were all (excluding pediatric patients) normal X-rays from the Montgomery and Shenzen databases, along with 925 normal images from ChestX-Ray14.

5.1.4 The Classification External Dataset

This dataset was used for external validation and testing for the classification task. The purpose for using an external database is to avoid any bias in the DNN evaluation, a concern that was brought up in other studies (MAGUOLO; NANNI, 2021).

In this dataset, the pneumonia and normal images were obtained from the CheXPert database. Hence, they come from a distinct location when compared to the training dataset, the Stanford University Hospital. 79 images for each of the two classes were randomly chosen, but necessarily included the images that were manually labeled by 3 radiologists.

We did not risk taking another COVID-19 dataset as an external database, as, due to the small availability of COVID-19 X-rays today, many open datasets have images in common. Our approach was to select all images from the Hannover Medical School (Hannover, Germany) in (COHEN et al., 2020) as our external dataset. This location was selected because it provided a reasonable (considering the small dataset size) number of X-rays, 79, and because, besides them, only 3 other images came from Germany (making it unlikely that a patient took an X-ray in Hannover and another in another hospital from our database).

The external classification dataset was divided in two parts, for validation and test. The test dataset contains 50 images from each class, while the validation one, 29. The division was random and with a patient split.

5.1.5 Data Processing

All X-rays (from all datasets in this part of our study) were loaded in greyscale, resized to 224x224x3 (U-Net and DenseNet input size), histogram equalization was applied, and the images were normalized between 0 and 1.

In the external classification dataset and the segmentation dataset, the images were made square with the addition of black borders before being resized to 224x224. This was done to avoid changing the lung aspect ratio. This technique was not used in the classification training dataset to avoid creating bias (a DNN that learns to identify the black borders), mostly in the DNN without segmentation.

5.2 COVID-19 DETECTION WITH LUNG SEGMENTATION: METHODOLOGY

In this section we explain our proposed stacked DNN, how we created segmentation masks, the training procedures, how we applied LRP, and the Bayesian model used for evaluation.

5.2.1 Training For Segmentation With The Montgomery And Shenzen Databases

Initially, we only had the segmentation masks for the Shenzen and Montgomery datasets, so we started by training a U-Net (with the original architecture, depicted in

figure 5 and in (RONNEBERGER; FISCHER; BROX, 2015) with them, so that we could use it to help us to create the COVID-19 and pneumonia masks.

To train this DNN, we first randomly divided the Shenzen and Montgomery X-rays that had masks in three datasets, for training (70%), validation (hold-out, 10%) and testing (20%).

In the training dataset we utilized data augmentation, multiplying the number of images by 8 (the original images were not removed), with random rotations (between -40 and 40 degrees), translations (with a maximum of 28 pixels up or down and also 28 left or right) and horizontal flipping (50% chance).

Again, we used the Python library PyTorch to train the DNN, as we did in the rest of this study. In this section of the dissertation (5), all training procedures were conducted in a NVidia RTX 3080 GPU, utilizing mixed precision (to take advantage of the card's tensor cores and accelerate training).

The U-Net was trained with cross-entropy loss, SGD with momentum of 0.99 and mini-batches of size 8. We trained with a learning rate of 10^{-4} for 200 epochs. Then we reduced it to 10^{-5} and activated a reduce on plateau lr scheduler, which reduced the learning rate by a factor of 10 when validation loss stopped decreasing for 20 epochs. We trained for 200 epochs more with the scheduler.

To evaluate the U-Net, we measured the mean intersection over union (IoU, a measurement of similarity between two binary images) between the DNN outputs and the test segmentation masks. We turned the DNN output binary by applying a 0.5 threshold: values over 0.5 became 1, below it, 0. IoU is calculated by dividing the intersection of the images (area where both are 1) by their union (area where one or the other is 1). After training our mean test IoU was 0.927 in the Montgomery and Shenzen datasets. In a qualitative analysis we found the U-Net outputs adequate.

5.2.2 Creating Segmentation Masks

After training the U-Net in the Montgomery and Shenzen databases, we employed it to generate the pneumonia and COVID-19 masks for our segmentation dataset.

Firstly, we used the DNN to generate the masks and transformed them in binary (with a threshold of 0.5). We then manually checked every mask, comparing them with the X-rays and editing where necessary. If a mask was not good enough, we recreated

it manually. As in the Montgomery and Shenzen masks, we did not include the lung region behind the heart, and we utilized anatomical landmarks (such as the ribs and the diaphragm) to recreate and edit masks.

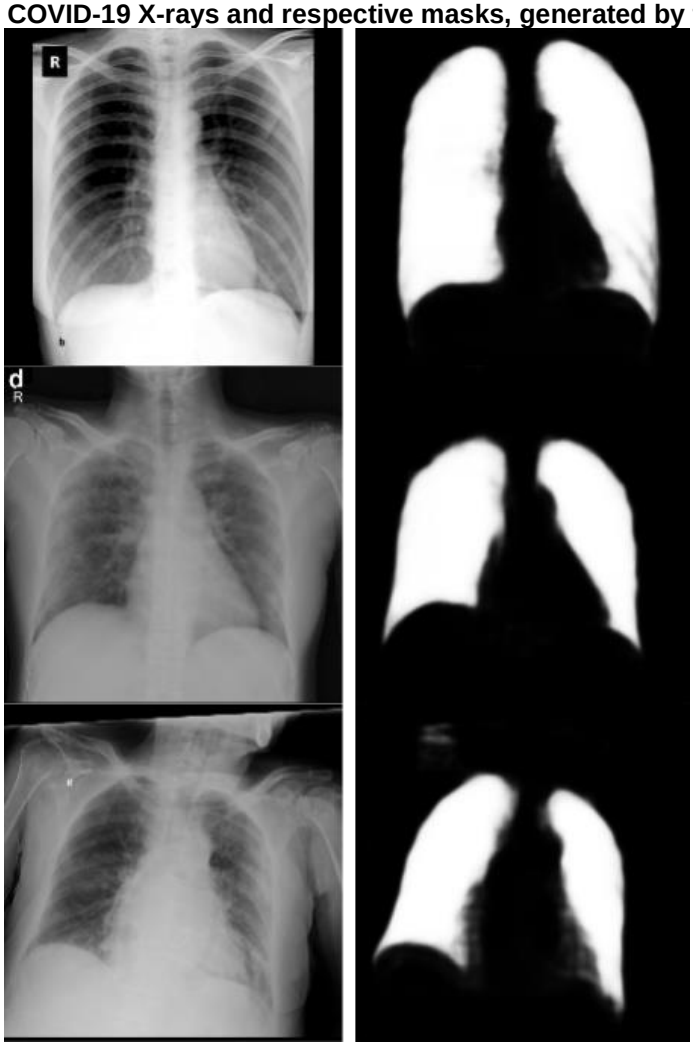
5.2.3 Training For Segmentation

With all the targets for our segmentation dataset, we were able to train a new U-Net on it. We utilized data augmentation (online) in the training segmentation dataset. It created new images and added them to the database, alongside the original ones (making the training database 15 times bigger). New images were created with random rotation (between -40 and 40 degrees), translation (maximum of 28 pixels up or down and 28 left or right) and horizontal flip (50% chance).

The U-Net was trained with cross-entropy loss, stochastic gradient descent with momentum of 0.99 and mini-batches of 5 images. We trained for 400 epochs with a learning rate of 10^{-4} and we could already observe overfitting at the end of this process.

We obtained a test mean IoU of 0.864 and the generated masks seemed appropriate in a qualitative analysis. Most of the DNN mistakes were generating whiter regions in the gastric bubble area and in the lung region behind the heart. Examples of generated COVID-19 masks are shown in figure 13.

Figure 13. COVID-19 X-rays and respective masks, generated by the U-Net.



5.2.4 The Stacked Deep Neural Network

The U-net we trained in section 5.2.3 was used as the segmentation module of our stacked deep neural network. This architecture stacks the U-Net (the segmentation module), an original intermediate module and a DenseNet20 (the classification module). We compared the stacked DNN with a simple DenseNet201 to understand the effect of segmentation on generalization with small and mixed COVID-19 chest X-ray datasets. We trained the proposed DNN and the DenseNet201 in the same manner (described in sections 5.2.5 and 5.2.6), to better compare them.

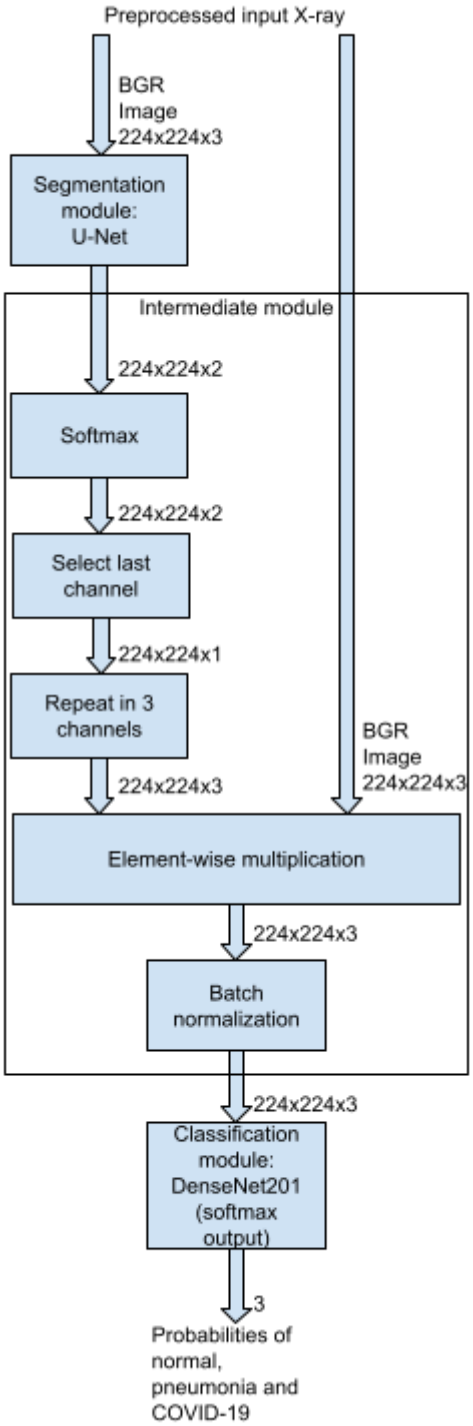
In this section we explain the stacked DNN architecture. The segmentation module is the already trained U-Net, and its parameters will be frozen (they shall not change during training for classification). It receives an X-ray and outputs its segmentation mask.

The intermediate module takes, as input, the U-Net output (with two channels) and the original X-ray. It firstly applies a softmax function to the U-Net output, restricting its values between 0 and 1, and takes only the last dimension of the softmax result (it indicates the lung regions with values close to 1 and unimportant areas with values close to 0). It then replicates this image in three channels and multiplies it (element-wise) with the input X-ray. The multiplication removes irrelevant areas of the X-ray and keeps lung regions. Finally, to improve generalization, the module performs batch normalization.

The output of the intermediate module serves as input for the classification module, a DenseNet201, which is trained to predict the probabilities of COVID-19, pneumonia or normal. We decided to use DenseNets in this module because we had good results with them in section 4 of this dissertation.

Figure 14 shows the stacked DNN structure and explains the intermediate module in detail.

Figure 14. The stacked deep neural network architecture.



5.2.5 Pretraining For Classification On Chestx-Ray14

We trained our DNN using a twice transfer learning approach, similarly to the procedure described in section 4. We began by downloading two ImageNet pretrained DenseNet201s, which would become the classification module in our stacked DNN

and the DNN without segmentation (we only substituted its output layer for one with 15 neurons, linear). The second step in twice transfer learning (described in this section) was training on the ChestX-ray14 dataset, and the third was on the COVID-19 classification dataset. Analogously to section 4, we expect training on ChestX-ray14 to improve generalization, as it is a large dataset, with a task that is similar to COVID-19 detection.

Again, we only applied random horizontal flipping (50% probability) as data augmentation in this dataset. Unlike in the other datasets, this online augmentation substituted the original images.

We utilized the original ChestX-ray14 test dataset and randomly selected the remaining images, with 80% for training and 20% for validation (hold-out). We utilized a patient split, avoiding images from the same patient in more than one of the created datasets.

Classifying ChestX-ray14 is a multi-label classification task. Therefore, we trained with PyTorch's binary cross-entropy loss with logits. We utilized SGD, with momentum of 0.9 and mini-batches of 64 images. We trained the DNNs' last layer for 20 epochs, with a learning rate of 10^{-3}. Afterwards, we trained the complete DNNs (except for the segmentation module, in case of the stacked network) for 80 epochs, with lr of 10^{-4}. In the end the networks were already overfitting.

5.2.6 Training With The Classification Dataset

Finally, we trained the stacked DNN and the DenseNet201, which we pretrained in section 5.2.5, in the classification training dataset, to classify X-rays as COVID-19, pneumonia or normal.

We began by replacing the pretrained DNNs last layer with one with 3 neurons and softmax activation, as the last task is multi-class but single-label. We also added a dropout of 50% before the output layer, because it improved the accuracy in the external validation dataset in preliminary tests.

In the training classification dataset, we utilized data augmentation to avoid overfitting and to balance the database. We used online data augmentation, new images were added to the dataset, along with the original ones, and they were created with random rotations (between -40 and 40 degrees), translations (up to 28 pixels up or down, left or right), and flipping (50% chance). To balance the database, different

numbers of images were generated for different classes: the number of normal and pneumonia images was multiplied by 3, and of COVID-19 images by 10. They did not create exactly the same number of images in all classes, we ended up with 3885 normal and pneumonia training images, and 3960 COVID-19 training images. Therefore, to perfectly balance the dataset, some of the images were left out of the training process (90 COVID-19 augmented images and 15 pneumonia and normal images), but they were randomly selected each epoch. Thus, all X-rays were used during the training procedure. Every epoch the DNN received 11610 training images (3870 for each class). We did not apply data augmentation to the external test and validation datasets.

We trained the two neural networks with cross-entropy loss, SGD using momentum of 0.9, mini-batches of 30 images and we used hold-out validation (using the external validation dataset). We first trained the DNNs' last layer, for 20 epochs, with a learning rate of 10^{-5} and weight decay of 0.01. Afterwards, we trained the whole DNNs (except for the segmentation module in the stacked network) for 240 epochs, using weight decay of 0.05 and differential learning rates (lr started as 10^{-5} in the last layer was divided by 10 for each dense block before it, achieving the smallest value in the DenseNet first layer) (HOWARD; RUDER, 2018). Each epoch took about 200s in our NVidia RTX 3080 GPU and at the end of the training process both DNNs were overfitting.

5.2.7 Performance Evaluation: Bayesian Model And LRP

In this part of our work, we applied LRP in a similar way to section 4, using the Python library iNNvestigate (ALBER et al., 2019) and using py2keras (MALIVENKO, 2018) to convert our models from PyTorch to Keras. As we were still working with DenseNet classifiers (LRP was only applied to the classifier module in the stacked DNN), we used the same iNNvestigate preset, LRP-A-flat.

We will now describe the Bayesian model that we used to create interval estimates for our performance metrics and understand the effect of our small test dataset in the statistical significance of our results.

The Bayesian model proposed in (ZHANG; WANG; ZHAO, 2015) can be used to estimate the probability distributions of F1-Scores in multi-class single-label classification problems (when there are more than two classes and a single label per

image, like in our COVID-19 datasets). It can be summarized by equations 6 to 10 (ZHANG; WANG; ZHAO, 2015):

$$\boldsymbol{\mu} \sim \text{Dir}(\boldsymbol{\beta}) \quad (6)$$

$$\mathbf{n} \sim \text{Mult}(N,\boldsymbol{\mu}) \quad (7)$$

$$\boldsymbol{\theta}_j \sim \text{Dir}(\boldsymbol{\alpha}_j) \quad \text{for } j=1,...,M \quad (8)$$

$$\mathbf{c}_j \sim \text{Mult}(n_j,\boldsymbol{\theta}_j) \quad \text{for } j=1,...,M \quad (9)$$

$$\psi = f(\boldsymbol{\mu},\boldsymbol{\theta}_1,...,\boldsymbol{\theta}_n) \quad (10)$$

N is the test dataset size (150), M is the number of classes (3), the Dirichlet distribution is represented by Dir() and the multinomial by Mult().

The random vector $\mathbf{n}$, with M elements n_j, estimates the number of samples in class j if we sample a test dataset with size N. The random vector $\boldsymbol{\mu}$, containing M elements μ_j, indicates the probability of a sampled element belonging to class j. The vector $\boldsymbol{\beta}$ contains the hyperparameters of the prior distribution of $\boldsymbol{\mu}$. $\boldsymbol{\beta}$=[1,1,1] creates a uniform prior, which was chosen by Zhang, Wang and Zhao (2015) and by us in this study.

The random vector $\mathbf{c}_j$, of size M has elements c_{jk}, which estimate the number of class j elements classified as class k. Therefore, the c_{jk} values create an expected confusion matrix, for the classifier and a new sampled test dataset. The random vector $\boldsymbol{\theta}_j$ has size M, and its θ_{jk} elements indicate the estimated probability of classifying a sample from class j as class k. The vector $\boldsymbol{\alpha}_j$ has the M hyper parameters that define the $\boldsymbol{\theta}_j$ prior distribution. We chose $\boldsymbol{\alpha}_j$=[1,1,1], creating a uniform prior, as did Zhang, Wang and Zhao (2015) in his implementation of the Bayesian model.

ψ indicates a function, deterministically calculated, using the posterior probability distributions of $\boldsymbol{\mu}$ and $\boldsymbol{\theta}$. In (ZHANG; WANG; ZHAO, 2015) we see the functions for many performance metrics: class precision P_j, class recall R_j, macro-averaged F1-Score (maF1) and micro-averaged F1-Score (miF1). In a multi-class single-label classification problem, like classifying the COVID-19 datasets, miF1 and overall accuracy have the same value (SAKAI, 2006). In a balanced test dataset, like ours, it is also identical to overall accuracy. Therefore, we will use these names interchangeably in sections 5.3 and 5.4.

We expanded the Bayesian model created by Zhang, Wang and Zhao (2015), to also estimate the probability distributions of the mean specificity and the classes' specificity. For this purpose, we needed to express specificity as a function (ψ) of **μ** and **θ**, allowing us to calculate it with their estimated posterior probability distributions.

In (ZHANG; WANG; ZHAO, 2015) the authors define equations for the estimated number of true negatives and false positives in the class j contingency table (tn_j and fp_j):

$$tn_j = \sum_{u \neq j} \sum_{v \neq j} N \mu_u \theta_{uv} \quad (11)$$

$$fp_j = \sum_{u \neq j} N \mu_u \theta_{uj} \quad (12)$$

Using equations 11 and 12, along with the definition of specificity, we can deduce the equations that calculate class specificity (s_j) and mean specificity (S) in terms of **μ** and **θ**:

$$s_j = \frac{tn_j}{tn_j + fp_j} = \frac{\sum_{u \neq j} \sum_{v \neq j} \mu_u \theta_{uv}}{\sum_{u \neq j} \sum_{v=1}^{M} \mu_u \theta_{uv}} \quad (13)$$

$$S = \left(\frac{1}{M}\right) \sum_{j=1}^{M} s_j \quad (14)$$

The Bayesian model uses only the classifier confusion matrix to calculate the likelihoods for $\mathbf{c}_j$ and **n**. We computed the posteriors utilizing Markov chain Monte Carlo (MCMC) and the No-U-Turn Sampler (HOFFMAN et al., 2011), with 4 chains, 10000 tuning samples and 100000 samples after tuning. The model was implemented with the Python library PyMC3 (SALVATIER; WIECKI; FONNESBECK, 2016).

5.3 COVID-19 DETECTION WITH LUNG SEGMENTATION: RESULTS AND DISCUSSION

We show, in tables 8 and 9, the test confusion matrices for the stacked DNN and the DenseNet201, respectively.

Table 8. Stacked DNN confusion matrix.

		Predicted Class		
		Normal	Pneumonia	COVID-19
Real Class	Normal	38	7	5
	Pneumonia	8	32	10
	COVID-19	2	0	48

Table 9. Confusion matrix for the DNN without segmentation.

		Predicted Class		
		Normal	Pneumonia	COVID-19
Real Class	Normal	43	0	7
	Pneumonia	14	24	12
	COVID-19	6	0	44

In table 10, we show the performance metrics for the stacked DNN, and, in table 11, for the DNN without segmentation. The “Score” column displays the traditional point estimate of the scores, deterministically calculated. The other columns display the statistics of the metrics' posterior distributions, estimated using Bayesian inference. These statistics are: mean, standard deviation (std), Monte Carlo error, and 95% high density interval (HDI). The 95% HDI is the interval that contains 95% of the metric’s probability mass and any point in that interval must have a probability higher than that of any point outside the HDI.

Table 10. Performance metrics for the stacked DNN.

Metric	Score	Mean	std	MC error	95% HDI
Mean Accuracy or miF1	0.787	0.761	0.034	0.0	[0.695,0.826]
Macro-averaged F1-Score	0.781	0.754	0.034	0.0	[0.687,0.82]
Macro-averaged Precision	0.791	0.764	0.034	0.0	[0.697,0.829]
Macro-averaged Recall	0.787	0.761	0.032	0.0	[0.698,0.823]
Macro-averaged Specificity	0.893	0.88	0.017	0.0	[0.848,0.912]
Normal Precision	0.792	0.765	0.059	0.0	[0.648,0.877]
Normal Recall	0.76	0.736	0.06	0.0	[0.617,0.85]
Normal F1-Score	0.776	0.748	0.048	0.0	[0.654,0.839]
Normal Specificity	0.9	0.887	0.031	0.0	[0.825,0.943]
Pneumonia Precision	0.821	0.786	0.063	0.0	[0.66,0.902]
Pneumonia Recall	0.64	0.623	0.066	0.0	[0.493,0.75]
Pneumonia F1-Score	0.719	0.692	0.054	0.0	[0.586,0.795]
Pneumonia Specificity	0.93	0.915	0.027	0.0	[0.861,0.964]
COVID-19 Precision	0.762	0.742	0.054	0.0	[0.636,0.844]
COVID-19 Recall	0.96	0.925	0.036	0.0	[0.854,0.985]
COVID-19 F1-Score	0.85	0.822	0.038	0.0	[0.746,0.894]
COVID-19 Specificity	0.85	0.84	0.035	0.0	[0.769,0.906]

Table 11. Performance metrics for the DNN without segmentation. Score values are point estimates, other values were created with Bayesian estimation.

Metric	Score	Mean	std	MC error	95% HDI
Mean Accuracy or miF1	0.74	0.717	0.036	0.0	[0.646,0.786]
Macro-averaged F1-Score	0.729	0.705	0.037	0.0	[0.632,0.776]
Macro-averaged Precision	0.794	0.758	0.032	0.0	[0.696,0.82]
Macro-averaged Recall	0.74	0.717	0.033	0.0	[0.653,0.781]
Macro-averaged Specificity	0.87	0.858	0.017	0.0	[0.825,0.891]
Normal Precision	0.683	0.667	0.058	0.0	[0.553,0.779]
Normal Recall	0.86	0.83	0.051	0.0	[0.728,0.924]
Normal F1-Score	0.761	0.738	0.045	0.0	[0.647,0.824]
Normal Specificity	0.8	0.792	0.039	0.0	[0.714,0.867]
Pneumonia Precision	1.0	0.926	0.05	0.0	[0.829,0.998]
Pneumonia Recall	0.48	0.472	0.068	0.0	[0.34,0.605]
Pneumonia F1-Score	0.649	0.622	0.063	0.0	[0.497,0.743]
Pneumonia Specificity	1.0	0.981	0.013	0.0	[0.955,1.0]
COVID-19 Precision	0.698	0.682	0.057	0.0	[0.569,0.792]
COVID-19 Recall	0.88	0.849	0.049	0.0	[0.752,0.938]
COVID-19 F1-Score	0.779	0.755	0.044	0.0	[0.667,0.839]
COVID-19 Specificity	0.81	0.802	0.039	0.0	[0.726,0.876]

Utilizing the external test dataset, we also calculated the multi-class AUC (area under the ROC curve), with micro averaging and the pairwise comparisons introduced in (HAND and TILL, 2004). Our stacked DNN achieved a test AUC of 0.917, and the DNN without segmentation, 0.906. We do not provide interval estimates for AUC, since defining them would not be a simple task, as HAND and TILL (2004) pointed in their paper. They also suggest the utilization of bootstrapping to create interval estimates, but we cannot use this technique in this study, because we are considering an external test dataset and the amount of COVID-19 chest X-rays is very limited.

Figures 15 and 16 show the Bayesian estimates of mean accuracy (miF1) and of macro-averaged F1-Score, for the stacked DNN and the DesneNet201, respectively. The corresponding trace plots, for a single MCMC chain, are displayed in figures 17 and 18 (excluding the tuning samples).

Figure 15. Probability density distributions for accuracy (left) and maF1 (right), for the DNN with segmentation.

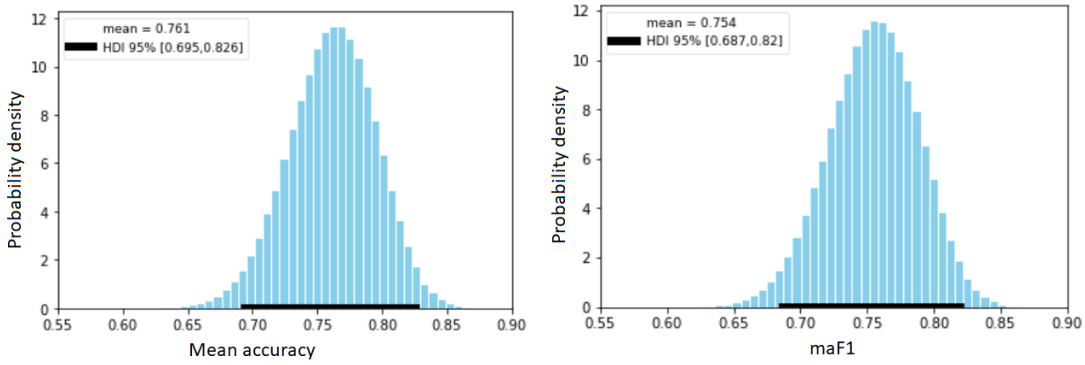


Figure 16. Probability density distributions for accuracy (left) and maF1 (right), for the DNN without segmentation.

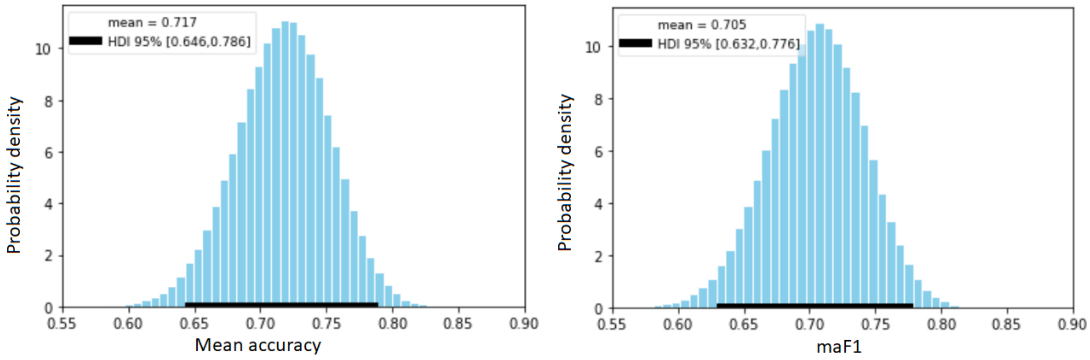


Figure 17. Trace plots for accuracy (left) and maF1 (right), for the DNN with segmentation.

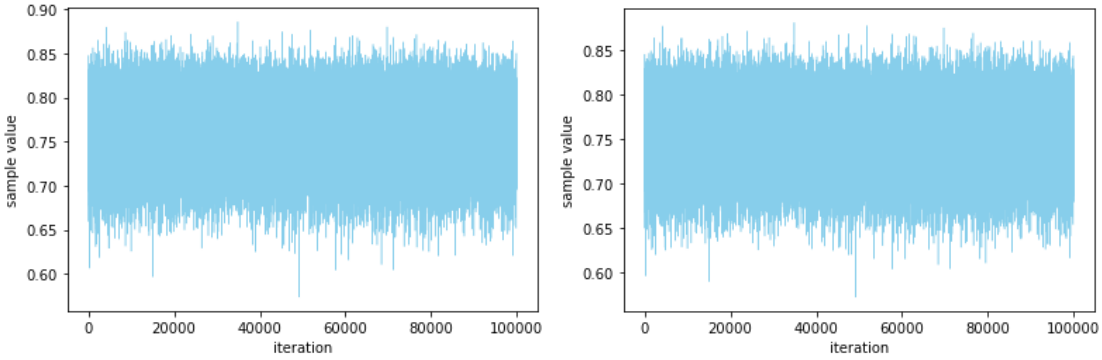
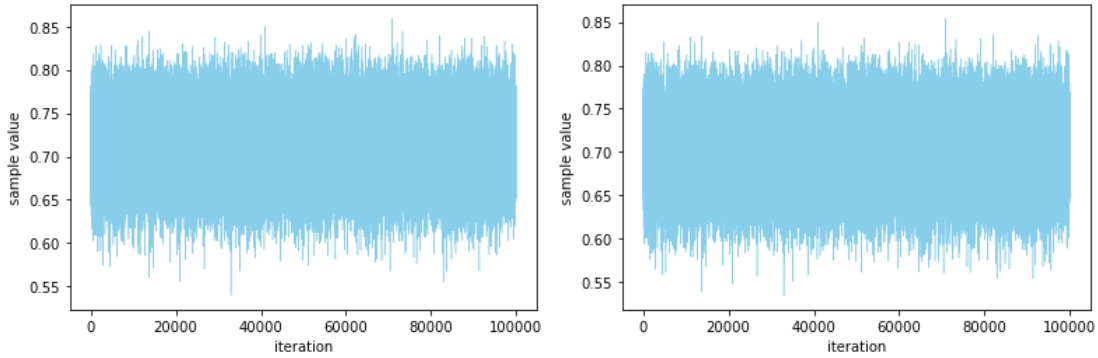


Figure 18. Trace plots for accuracy (left) and maF1 (right), for the DNN without segmentation.



In section 4 of this dissertation, we did not perform validation and testing on an external dataset, and we were able to achieve test accuracies above 90%, even reaching 100%. These high values are also very common in many COVID-19 studies that also employ internal validation (randomly splitting a single dataset for training, validation and testing), as can be seen in (SHOEBI et al., 2020). Preliminary tests with our stacked DNN and internal validation also showed accuracies above 90%, even separating the patients when creating the test dataset (as we did in section 4). Therefore, we see that evaluating the DNNs in an external dataset shows much smaller accuracies, indicating that the small and mixed datasets can cause bias and artificially improve performance metrics.

When we compare the results of our stacked DNN to the DenseNet201, we see that segmentation improves generalization, increasing the mean accuracy score on the external test dataset by 4.7%, and the Bayesian estimation mean by 4.4%.

We can see that we achieve a specificity of 90% (or 88.7%, according to the Bayesian estimate) in the normal class, with the stacked DNN. This high value is important, as it indicates that a relatively small number of sick patients were classified as normal by the DNN.

Finally, we observe, in figures 15 and 16, as well as in tables 10 and 11, that our 95% HDIs are relatively large, for example, it has a length of 0.131 in the stacked DNN mean accuracy. We think that this happens due to the small size of the test dataset (150 images), which reduces the statistical significance of our metrics.

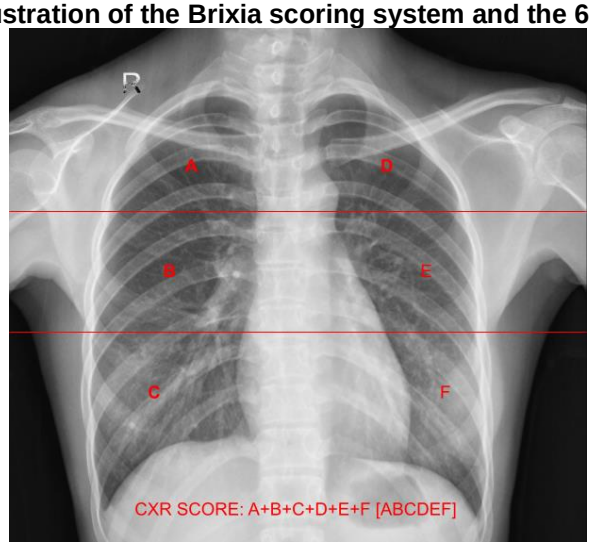
5.3.1 LRP Analysis And Comparison With Radiologists, Using The Brixia Score

In this section, we present a new method to compare DNN analysis to radiologist's, which is based on LRP and the Brixia score, detailed in (BORGHESI; MAROLDI, 2020).

The Brixia scoring system was created to allow radiologists to grade the severity of COVID-19 symptoms in different parts of the lungs, using chest X-rays, and Borghesi and Maroldi, (2020) showed that higher overall scores were associated with higher death probability. The system starts by dividing the lungs in six regions, A, B, C, D and E. Two lines are used to create the regions, the upper one is at the inferior wall of the aortic line, and the lower at the level of the right pulmonary vein. For each region, the radiologist assigns a score, from 0 to 3. 0 indicates no abnormalities, 1 indicates

interstitial infiltrates, 2 interstitial and alveolar infiltrates, with interstitial predominance, and 3 interstitial and alveolar infiltrates, with alveolar predominance. Summing the 6 partial scores we obtain the overall brixia score, which ranges from 0 to 18. In the X-ray the score is presented with the overall score followed by the partial scores between square brackets (from A to F, [ABCDEF]). We show, in figure 19, the six regions and the score presentation.

Figure 19. Illustration of the Brixia scoring system and the 6 lung regions.

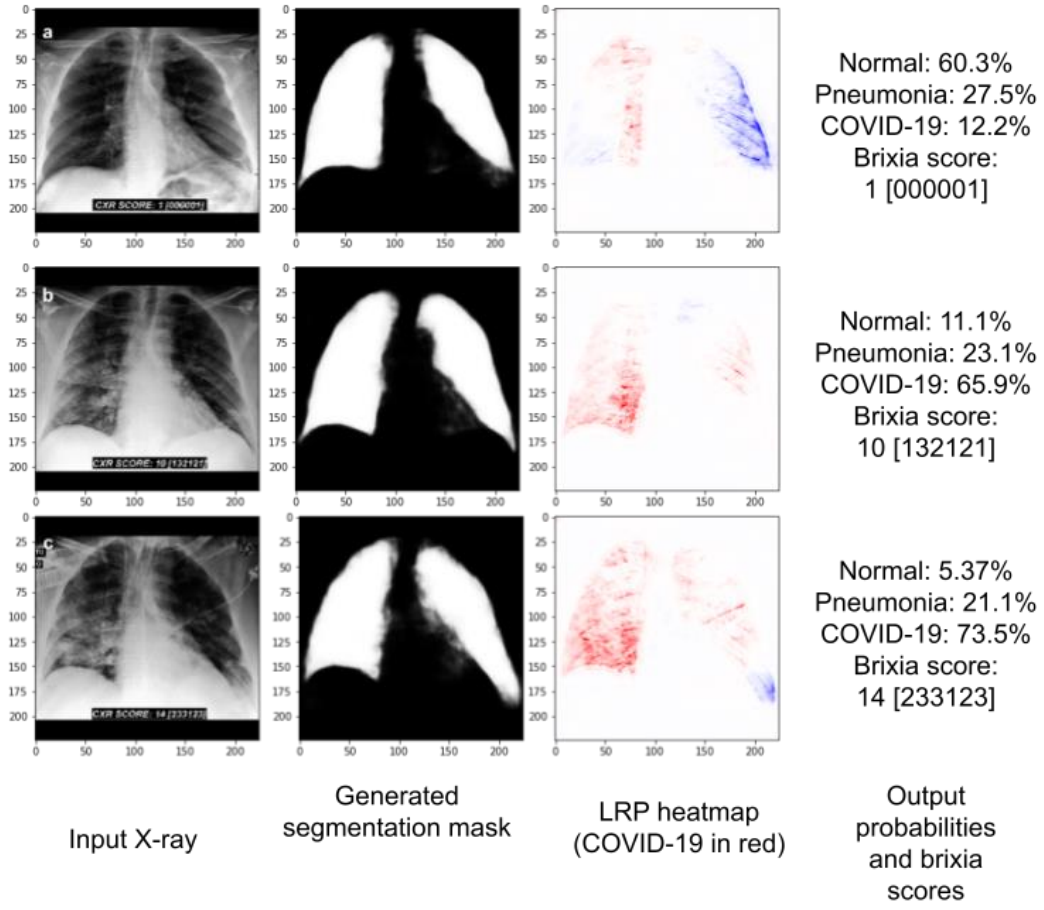


Our proposed analysis method consists in comparing X-rays analyzed by radiologists, with the Brixia score, with LRP heatmaps of the same X-rays. Starting the relevance propagation in the neuron that classified COVID-19, the heatmap will indicate, in red, regions that the DNN associated with COVID-19 symptoms. The larger and darker the red region, the stronger the signs of COVID-19, according to the network. If we compare the colors of the six lung regions in the heatmap with their partial brixia score we can understand if the DNNs and the radiologists look for the same signs of coronavirus. Furthermore, stronger symptoms lead to higher overall brixia scores and could also increase the COVID-19 probability predicted by the DNN. Thus, we may also look for a correlation between overall brixia score and the COVID-19 predicted probabilities.

Borghesi and Maroldi (2020) contains examples of COVID-19 X-rays already scored by radiologists. These images are also part of the COVID-19 image data collection and of our training dataset. In figure 20, we show three of them, their Brixia scores, LRP heatmaps (according to the stacked DNN, propagating from the output

neuron that classifies COVID-19) and the stacked DNN output for them. We chose these three X-rays because, unlike the other radiographies in Borghesi and Maroldi (2020), they had nothing written over the lungs. As we started LRP in the neuron that classifies COVID-19, red areas indicate the disease, while blue areas indicate pneumonia or healthy areas.

Figure 20. COVID-19 X-rays analyzed with the stacked DNN and with the Brixia score, by radiologists.



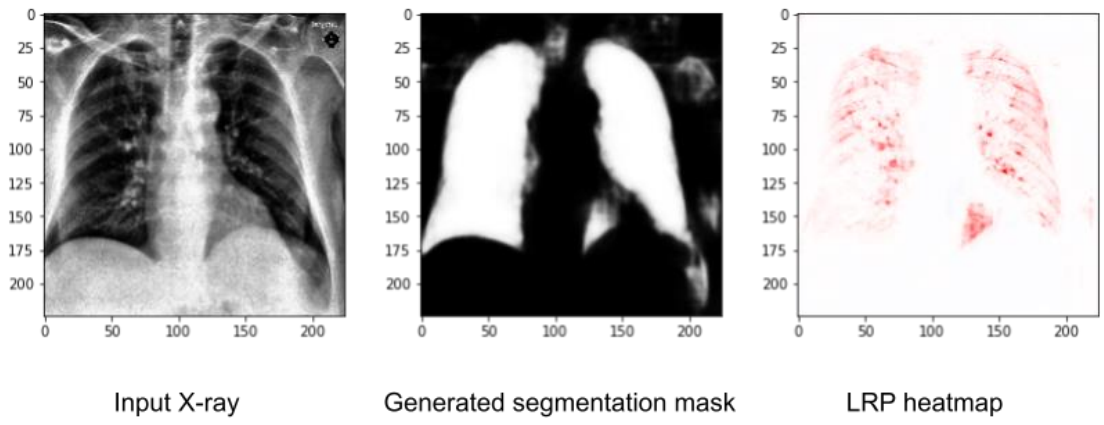
All the X-rays in figure 20 are from the same patient, a 72-year-old man with COVID-19. The first row has an image from his day of admission in the hospital, one day after the onset of fever (BORGHESI; MAROLDI, 2020). The analysis with Brixia scores shows little signs of COVID-19, with an overall score of 1. This can make classification challenging and we see that our DNN misclassified the image, attributing the normal class, but with only 60.3% of probability. The patient disease progression was fast, the second row has an image of the day 4 post-hospitalization and row 3 of day 5 (BORGHESI; MAROLDI, 2020). They have much higher Brixia scores, and our

DNN correctly classified both. Although our DNN was not created to classify disease severity, we see in figure 20 that images with higher overall Brixia scores show higher COVID-19 predicted probabilities. This indicates a correlation between the COVID-19 symptoms that the radiologists analyze and the ones our DNN analyses.

We now look at the partial Brixia scores and compare them to the heatmaps in figure 20. In the two correctly classified X-rays, we observe more relevance in the right lung in the heatmaps, and the right lungs also show higher Brixia scores. In the middle row of figure 20, in the right lung, we see more relevance in regions B and C, which also have the highest brixia scores in the lung; in the left lung the highest brixia score is in zone E, which also shows more relevance than zones D and F. In the third row's X-ray, the partial Brixia scores are again higher in regions B and C, which also show more relevance. We see a potential problem in region F of the last X-ray, as it has a high partial Brixia score (3) but is blue in the heatmap. Propagating relevance from the other output neurons, we discovered the problem is that the region was associated with pneumonia by the DNN, which may indicate similarity of symptoms.

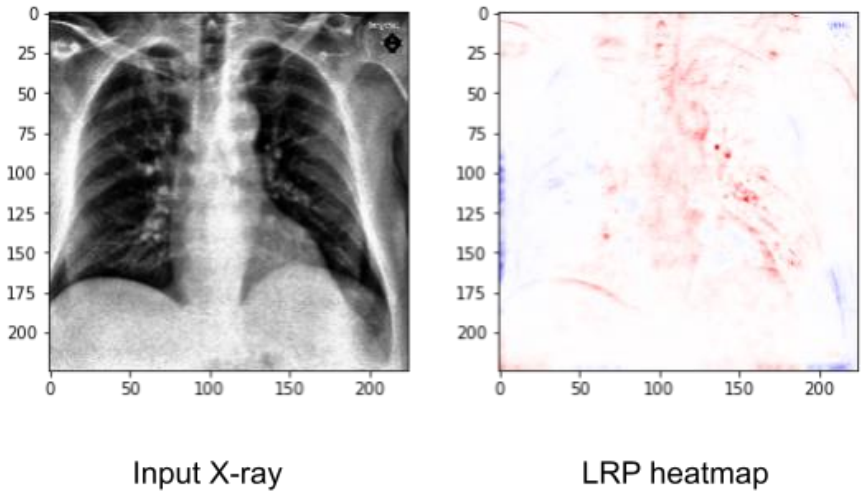
We also utilized LRP to check if our segmentation and intermediate modules were working as intended and removing relevance from unimportant regions of the X-rays. We see in figure 20 that this was the case. We also verified this aspect using test COVID-19 images, an example is shown in figure 21, along with its heatmap and the U-Net generated segmentation mask. In this case we observe that the mask is not perfect, but the regions outside of the lungs were not very bright and there is almost no relevance over them. The X-ray in figure 21 was correctly classified as COVID-19 by the stacked DNN, with 89.6% estimated probability of the disease.

Figure 21. COVID-19 test X-ray analyzed with the stacked DNN and LRP.



We can also use LRP to compare the stacked DNN and the DNN without segmentation. Therefore, we can compare the heatmaps in figures 20 and 21 to the one in figure 22, which was created with the DenseNet201, using the same X-ray shown in figure 21. The DNN without segmentation also correctly classified the image, but it estimated only a 46.2% probability of COVID-19. Again, red areas in the image indicate COVID-19 and other areas were associated with the other two classes.

Figure 22. COVID-19 test X-ray analyzed with the DNN without segmentation and LRP.



We can see relevance outside of the lungs in figure 22, which may create bias and hinder generalization, explaining why the stacked DNN has better performances in the external datasets. However, some COVID-19 signs shown in the heatmap in figure 21 are also present in figure 22 (mostly on the left lung).

5.4 COVID-19 DETECTION WITH LUNG SEGMENTATION: CONCLUSION

In this part of our study, we analyzed the effects of lung segmentation on generalization. Firstly, we observed 78.7% mean test accuracy with segmentation and 74% without it. In section 4 test accuracies reached 100%, in the preliminary test of the stacked DNN with internal validation test accuracy was above 90%, and many other studies using DNNs with internal validation and small and mixed dataset (like ours) show accuracies in the 90% range (SHOEBI et al., 2020). The much lower performance with external validation may indicate that small and mixed dataset, like

most of the open COVID-19 X-rays dataset today, create bias, which artificially improves performances in internal validation, as Maguolo and Nanni (2021) suggest.

With segmentation, in our stacked DNN, we could improve generalization and increase the mean accuracy score in the external test dataset by 4.7% (increasing the Bayesian estimation mean by 4.4%). Some other techniques we think helped mitigating dataset bias in this study were histogram equalization (in the input X-rays), batch normalization (in our intermediate module), removing pediatric patients from the datasets (as the youngest patient in the COVID-19 class is 20 years old), utilizing an external validation dataset, regularization (dropout and weight decay), twice transfer learning and data augmentation. We observed that preprocessing the data before classification (e.g., with lung segmentation and histogram equalization) can be important to improve generalization, even with deep learning, as also explored in other works (MARQUES, 2018).

Knowing that our small test dataset size would reduce the statistical significance of our performance metrics, we resorted to Bayesian estimation to quantify this effect and create interval estimates of them. Our calculated 95% high density intervals were relatively large (e.g., 0.131 length in the stacked DNN miF1), as expected with only 150 test images. This result shows the importance of making interval estimates and also shows how large, high quality and open COVID-19 X-ray datasets, with all classes collected from the same sources, would be beneficial.

We suggested comparing LRP heatmaps to X-rays analyzed by radiologists with the Brixia score, and this indicated that the radiologists and the stacked DNN tend to focus on the same symptoms of COVID-19 in X-rays. Regions with higher partial brixia scores normally showed higher relevance on heatmaps and images with higher overall score had higher predicted COVID-19 probability. Still, our DNN could not correctly classify an X-ray with a very low overall score (only 1).

LRP also showed that the stacked DNN correctly segmented the lungs and ignored areas outside them (they had low relevance). It also showed that the DNN without segmentation paid attention to areas outside of the lungs, suggesting that lung segmentation can reduce dataset bias.

Therefore, we can affirm that the bias from small and mixed datasets is significant, but our DNNs' performance on the external datasets and our analysis with LRP and the Brixia score indicate that it can be partially avoided and DNNs can be

successfully trained on this type of database. On the external test dataset, with our stacked DNN, we achieved 0.916 AUC, and, with a Bayesian model, we estimated a macro-averaged F1-Score with mean of 0.754 and 95% high density interval of [0.687,0.82].

Finally, even though we explored external validation, clinical tests are needed to further ensure that the performances we observed in this study are replicable in a real-world scenario. This last part of our work can also be seen in the preprint (BASSI; ATTUX, 2021b).

6 GENERAL CONCLUSION

In this work we studied deep convolutional neural networks for image classification in two biomedical contexts: SSVEP based brain-computer interfaces and COVID-19 detection with chest X-rays.

In both contexts we had original contributions, which generated positive results. In the SSVEP context, we modified the SpecAugment technique to utilize it with BCIs (using smaller and fewer spectrograms in comparison to speech recognition, the area which SpecAugment was developed for). Using window slicing with 0.5 s windows and 0.5 s displacement, the technique improved the DNN mean accuracy by 0.6% and mean F1-Score by 0.008. With 0.1 s displacement, our modified SpecAugment improved the DNN mean accuracy by 0.4% and the mean F1-Score by 0.006.

In the context of COVID-19 detection with chest X-rays, we proposed the output neuron keeping (ONK) technique, a stacked DNN to perform segmentation and classification, and an evaluation technique, comparing radiologist's analysis with the Brixia score with LRP heatmaps. In the first COVID-19 study, analyzing the DenseNet201 DNNs, ONK had a strong effect improving accuracy in the beginning of the training process, and it also improved the final accuracy (by 1.3%). Our stacked DNN successfully ignored areas outside of the lungs (as shown by LRP), and improved generalization (increasing accuracy by 4.7%). Finally, our analysis with the Brixia score and LRP heatmaps indicated that the stacked DNN and radiologists look for the same COVID-19 signs.

In this study we observed that deep neural networks have great potential in the field of image recognition with biomedical data. They outperformed other classifiers, like SVMs, in single channel BCIs, and showed potential to become an auxiliary diagnosis method for COVID-19.

We also saw some of the DNNs' limitations, analyzing the effects of small and mixed COVID-19 X-ray datasets in generalization. Furthermore, we searched for methods to avoid these limitations, like lung segmentation, and we studied techniques that make DNNs more interpretable, like LRP and our analysis with heatmaps and the Brixia score.

Perspectives of future work are expanding the good results that we achieved with DNNs in a single-channel BCI to the multi-channel case and training deep neural networks for COVID-19 detection in large datasets, when they become available.

BIBLIOGRAPHY

ALBER, M. et al. iNNvestigate neural networks!. **J. Mach. Learn. Res.**, [*S. l.*], v. 20, n. 93, p. 1-8, 2019. Disponível em: https://jmlr.org/papers/volume20/18-540/18-540.pdf. Acesso em: 19 ago. 2021.

ALU, E. CECNL_RealTimeBCI. **Github**, [*S. l.*], 2019. Disponível em: https://github.com/eugeneALU/CECNL_RealTimeBCI. Acesso em: 20 ago. 2021.

BACH, S. et al. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. **PloS one**, [*S. l.*], v. 10, n. 7, p. e0130140, 10 jul. 2015. DOI: 10.1371/journal.pone.0130140. Disponível em: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0130140. Acesso em: 20 ago. 2021.

BASSI, P. R. A. S.; ATTUX, R. A deep convolutional neural network for COVID-19 detection using chest X-rays. **Research on Biomedical Engineering**, [*S. l.*], p. 1-10, 02 abr. 2021a. ISSN 2446-4740. DOI: 10.1007/s42600-021-00132-9. Disponível em: https://link.springer.com/article/10.1007/s42600-021-00132-9. Acesso em: 19 ago. 2021.

BASSI, P. R. A. S.; ATTUX, R. COVID-19 detection using chest X-rays: is lung segmentation important for generalization? **ArXiv**, [*S. l.*], 12 abr. 2021b). Disponível em: https://arxiv.org/abs/2104.06176. Acesso em: 19 ago. 2021.

BASSI, P. R. A. S.; ATTUX, R. Covid-19 twice transfer dnns. **Github**, [*S. l.*], 2020. Disponível em: https://github.com/PedroRASB/COVID-19-Twice-Transfer-DNNs. Acesso em: 19 ago. 2021.

BASSI, P. R. A. S.; RAMPAZZO, W.; ATTUX, R. Redes neurais profundas triplet aplicadas a classificação de sinais em interfaces cérebro-computador. *In*: Simpósio Brasileiro de Telecomunicações e Processamento de Sinais, 37., Petrópolis, 2019. **Anais [...]**, Petrópolis: SBrT, 29 set. - 02 out. 2019. Disponível em: https://biblioteca.sbrt.org.br/articlefile/2076.pdf. Acesso em: 19 ago. 2021.

BASSI, P. R. A. S.; RAMPAZZO, W.; ATTUX, R. Transfer learning and SpecAugment applied to SSVEP based BCI classification. **Biomedical Signal Processing and Control**, [*S. l.*], v. 67, p. 102542, maio 2021. DOI: 10.1016/j.bspc.2021.102542. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1746809421001397?via%3Dihub. Acesso em: 19 ago. 2021.

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. **IEEE transactions on pattern analysis and machine intelligence**, [*S. l.*], v. 35, n. 8, p. 1798-1828, 2013. DOI: 10.1109/TPAMI.2013.50. Disponível em: https://ieeexplore.ieee.org/abstract/document/6472238?casa_token=c2i5SCpYMtgAAAAA:Q2AoK_Q7wnKYQfgEEdFzb966JkfW8JDmiNMIxEUNL7XxaiEtq8xnJfQb7FZCG1RNTmbXyRwSaeY. Acesso em: 19 ago. 2021.

BEVERINA, F. et al. User adaptive BCIs: SSVEP and P300 based interfaces. **PsychNology Journal**, [*S. l.*], v. 1, n. 4, p. 331-354, 2003. Disponível em: https://psycnet.apa.org/record/2004-19755-001. Acesso em: 19 ago. 2021.

BORGHESI, A.; MAROLDI, R. COVID-19 outbreak in Italy: experimental chest X-ray scoring system for quantifying and monitoring disease progression. **La radiologia medica**, [*S. l.*], v. 125, n. 5, p. 509-513, 01 maio 2020. DOI: 10.1007/s11547-020-01200-3. Disponível em: https://link.springer.com/article/10.1007/s11547-020-01200-3. Acesso em: 19 ago. 2021.

CAI, Q.; LIU, X.; GUO, Z. Identifying Architectural Distortion in Mammogram Images Via a SE-DenseNet Model and Twice Transfer Learning. *In*: International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, 11., China, 2018. **Anais [...]**, China, 13-15 out. 2018. p. 1-6. DOI: 10.1109/CISP-BMEI.2018.8633197.

CHEN, X. et al. Filter bank canonical correlation analysis for implementing a high-speed SSVEP-based brain–computer interface. **Journal of neural engineering**, [*S. l.*], v. 12, n. 4, p. 046008, 2015. Disponível em: https://iopscience.iop.org/article/10.1088/1741-2560/12/4/046008/pdf. Acesso em: 19 ago. 2021.

COHEN, J. P. et al. Covid-19 image data collection: Prospective predictions are the future. **ArXiv**, [*S. l.*], 14 dez. 2020. Disponível em: https://arxiv.org/abs/2006.11988. Acesso em: 20 ago. 2021.

CUI, Z.; CHEN, W.; CHEN, Y. Multi-scale convolutional neural networks for time series classification. **ArXiv**, [*S. l.*], 11 maio 2016. Disponível em: https://arxiv.org/abs/1603.06995. Acesso em: 20 ago. 2021.

DENG, J. et al. Imagenet: A large-scale hierarchical image database. *In*: IEEE conference on computer vision and pattern recognition. Miami: Ieee, 20-25 jun. 2009. p. 248-255. DOI: 10.1109/CVPR.2009.5206848.

GALLOWAY, N. R. Human brain electrophysiology: Evoked potentials and evoked magnetic fields in science and medicine. **The British journal of ophthalmology**, [*S. l.*], v. 74, n. 4, p. 255, abr. 1990. DOI: 10.1136/bjo.74.4.255-a. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1042082/. Acesso em: 20 ago. 2021.

GEMMEKE, J. F. et al. Audio set: An ontology and human-labeled dataset for audio events. *In*: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017. **Anais [...]**, Nova Orleans: Ieee, 5-9 mar. 2017. p. 776-780. DOI: 10.1109/ICASSP31846.2017. Disponível em: https://ieeexplore.ieee.org/abstract/document/7952261?casa_token=gvi40t8_dqYAAAAA:PfivpOgoVE92e4we728OfTIuTA8Bvn6GLbjReRUqr7A-nO3yVarz_O_GJko2aX1SZLuve64HNVY. Acesso em: 20 ago. 2021.

GÉRON, A. **Hands-on Machine Learning with Scikit-Learn, Keras & Tensorflow**. 2. ed. Canadá: O'Reilly Media, 2019.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep learning**. Cambridge: MIT Press, nov. 2016 ISBN: 9780262035613. 800 pp.

HEO, S.-J. et al. Deep learning algorithms with demographic information help to detect tuberculosis in chest radiographs in annual workers' health examination data. **International journal of environmental research and public health**, [*S. l.*], v. 16, n. 2, p. 250, jan. 2019. DOI: 10.3390/ijerph16020250. Disponível em: https://pubmed.ncbi.nlm.nih.gov/30654560/. Acesso em: 20 ago. 2021.

HERSHEY, S. et al. CNN architectures for large-scale audio classification. *In*: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017. **Anais [...]**, Nova Orleans: Ieee, 5-9 mar. 2017. p. 131-135. DOI: 10.1109/ICASSP.2017.7952132. Disponível em: https://ieeexplore.ieee.org/abstract/document/7952132. Acesso em: 20 ago. 2021. p. 131-135.

HOFFMAN, M. D. et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. **Journal of Machine Learning Research**, [*S. l.*], v. 15, n. 1, p. 1593-1623, 2014. Disponível em: https://www.jmlr.org/papers/volume15/hoffman14a/hoffman14a.pdf. Acesso em: 20 ago. 2021.

HOWARD, J.; RUDER, S. Universal language model fine-tuning for text classification. **ArXiv**, [*S. l.*], S. 328-339, maio 2018. Disponível em: https://arxiv.org/abs/1801.06146. Acesso em: 20 ago. 2021.

HUANG, G. et al. Densely connected convolutional networks. *In*: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2017. **Anais [...]**, Nova Orleans: Ieee, 5-9 mar. 2017. p. 4700-4708. Disponível em: https://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html. Acesso em: 20 ago. 2021. p. 131-135.

IRVIN, J. et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *In*: Proceedings of the AAAI conference on artificial intelligence, 33., Havaí, n. 1, 2019. **Anais [...]**, Honolulu: AAAI-19, 27 jan. – 01 fev. 2019. p. 590-597. DOI: 10.1609/aaai.v33i01.3301590. Disponível em: https://ojs.aaai.org/index.php/AAAI/article/view/3834. Acesso em: 20 ago. 2021.

JAEGER, S. et al. Two public chest X-ray datasets for computer-aided screening of pulmonary diseases. **Quantitative imaging in medicine and surgery**, [*S. l.*], v. 4, n. 6, p. 475, dez. 2014. DOI: 10.3978/j.issn.2223-4292.2014.11.20. Disponível em: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4256233/. Acesso em: 20 ago. 2021.

KWAK, N.-S.; MÜLLER, K.-R.; LEE, S.-W. A convolutional neural network for steady state visual evoked potential classification under ambulatory environment. **PloS one**, [*S. l.*], v. 12, n. 2, p. e0172578, 22 fev. 2017. DOI: 10.1371/journal.pone.0172578. Disponível em: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0172578. Acesso em: 20 ago. 2021.

MAGUOLO, G.; NANNI, L. A critic evaluation of methods for covid-19 automatic detection from x-ray images. **Information Fusion**, [*S. l.*], v. 76, p. 1-7, dez. 2021. DOI: 10.1016/j.inffus.2021.04.008. Disponível em: https://www.sciencedirect.com/science/article/abs/pii/S1566253521000816?via%3Dihub. Acesso em: 20 ago. 2021.

MARQUES, A. **Contribuição à abordagem de problemas de classificação por redes convolucionais profundas**. 2018. 1 recurso online (121 p.). Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2018. Disponível em: http://www.repositorio.unicamp.br/handle/REPOSIP/331033. Acesso em: 3 set. 2018.

MONTAVON, G. et al. Layer-wise relevance propagation: an overview. Explainable AI: interpreting, explaining and visualizing deep learning. *In*: DEMIRKOL, A.; DEMIR, Z.; EMRE, E. **Lecture notes in computer science**. [*S. l.*], 10 set. 2019. p. 193-209. Disponível em: https://link.springer.com/chapter/10.1007/978-3-030-28954-6_10. Acesso em: 20 ago. 2021.

NGUYEN, T.-H.; CHUNG, W.-Y. A single-channel SSVEP-based BCI speller using deep learning. **IEEE Access**, [*S. l.*], v. 7, p. 1752-1763, 14 dez. 2018. DOI: 10.1109/ACCESS.2018.2886759. Disponível em: https://ieeexplore.ieee.org/abstract/document/8576511. Acesso em: 20 ago. 2021.

PARK, D. S. et al. Specaugment: A simple data augmentation method for automatic speech recognition. **ArXiv**, [*S. l.*], 03 dez. 2019. DOI: 10.21437/Interspeech.2019-2680. Disponível em: https://arxiv.org/abs/1904.08779. Acesso em: 20 ago. 2021.

RAJPURKAR, P. et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. **ArXiv**, [*S. l.*], 25 dez. 2017. Disponível em: https://arxiv.org/abs/1711.05225. Acesso em: 20 ago. 2021.

RONNEBERGER, O.; FISCHER, P.; BROX, T. U-net: Convolutional networks for biomedical image segmentation. *In*: International Conference on Medical image computing and computer-assisted intervention, Alemanha, 2015. **Anais [...]**, Munique: Springer, 05-09 out. 2015. p. 234-241. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28. Acesso em: 20 ago. 2021.

SAKAI, T. Evaluating evaluation metrics based on the bootstrap. *In*: Annual international ACM SIGIR conference on Research and development in information retrieval, 29., EUA, 2006. **Anais [...]**, Nova Iorque, 06 ago. 2006, p. 525-532. DOI: 10.1145/1148170.1148261. Disponível em: https://dl.acm.org/doi/proceedings/10.1145/1148170. Acesso em: 20 ago. 2021.

SALVATIER, J.; WIECKI, T. V.; FONNESBECK, C. Probabilistic programming in Python using PyMC3. **PeerJ Computer Science**, [*S. l.*], v. 2, p. e55, 2016. DOI: 10.7717/peerj-cs.55. Disponível em: https://peerj.com/articles/cs-55.pdf. Acesso em: 20 ago. 2021.

SETIONO, T. et al. Brain Computer Interface for controlling RC-Car using Emotiv Epoc+. **Journal of Telecommunication, Electronic and Computer Engineering**

(JTEC), [*S. l.*], v. 10, n. 2-3, p. 169-172, 31 maio 2018. Disponível em: https://jtec.utem.edu.my/jtec/article/view/4212. Acesso em: 20 ago. 2021.

SHOEIBI, A. et al. Automated detection and forecasting of covid-19 using deep learning techniques: A review. **ArXiv**, [*S. l.*], 27 jul. 2020. Disponível em: https://arxiv.org/abs/2007.10785. Acesso em: 20 ago. 2021.

SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. **ArXiv**, [*S. l.*], 04 set. 2014. Disponível em: https://arxiv.org/abs/1409.1556. Acesso em: 20 ago. 2021.

STIRENKO, S. et al. Chest X-ray analysis of tuberculosis by deep learning with segmentation and augmentation. *In*: 2018 International Conference on Electronics and Nanotechnology (ELNANO), 38., Ucrânia, 2018. **Anais [...]**, Kyiv: IEEE, 24-26 abr. 2018. p. 422-428. DOI: 10.1109/ELNANO.2018.8477564. Disponível em: https://ieeexplore.ieee.org/abstract/document/8477564?casa_token=ZIw2ksHIRBIAAAAA:8a8YzOe490JIiaonwRYCnP3tDtZjx4HScd8KwloDIJJfx5FbO2Gt0lpvgQvKpln1x6XW9C59F_o. Acesso em: 20 ago. 2021.

SUTJIADI, R.; PATTIASINA, T. J.; LIM, R. SSVEP-based Brain-computer Interface for Computer Control Application Using SVM Classifier. **International Journal of Engineering & Technology**, [*S. l.*], v. 7, n. 4, p. 2722-2728, 27 set. 2018. ISSN 2227-524X. Disponível em: http://repository.ikado.ac.id/62/. Acesso em: 20 ago. 2021.

WANG, L.; LIN, Z. Q.; WONG, A. Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. **Scientific Reports**, [*S. l.*], v. 10, n. 1, p. 1-12, 11 nov. 2020. DOI: 10.1038/s41598-020-76550-z. Disponível em: https://www.nature.com/articles/s41598-020-76550-z. Acesso em: 20 ago. 2021.

WANG, X. et al. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *In*: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Havaí, 2017. **Anais [...]**, Honolulu: Ieee, 21-26 jul. 2017, p. 2097-2106. DOI: 10.1109/TNSRE.2016.2627556. Disponível em: https://openaccess.thecvf.com/content_cvpr_2017/html/Wang_ChestX-ray8_Hospital-Scale_Chest_CVPR_2017_paper.html. Acesso em: 20 ago. 2021.

WANG, Y. et al. A benchmark dataset for SSVEP-based brain–computer interfaces. **IEEE Transactions on Neural Systems and Rehabilitation Engineering**, [*S. l.*], v. 25, n. 10, p. 1746-1752, 2016. Disponível em: https://ieeexplore.ieee.org/abstract/document/7740878?casa_token=X0K6Cla0gSUAAAAA:mtJDdjj_zK9Kd27lWCGQVKenu-BkxxWxjsp5WG8OTjw5Pifz3VwNGny-OkSONCAz77qQVo1WN7M. Acesso em: 20 ago. 2021.

ZECH, J. Reproduce-chexnet. **Github**, [*S. l.*], 2018. Disponível em: https://github.com/jrzech/reproduce-chexnet. Disponível em:. Acesso em: 20 ago.

ZHANG, D.; WANG, J.; ZHAO, X. Estimating the uncertainty of average F1 scores. In: International Conference on The Theory of Information Retrieval, 15., EUA, 2015.

Anais [...], Massachusetts, 27-30 set. 2015, p. 317-320. ISBN: 978-1-4503-3833-2. Disponível em: https://dl.acm.org/doi/abs/10.1145/2808194.2809488. Acesso em: 20 ago.

ZHANG, Y. et al. Pushing the limits of semi-supervised learning for automatic speech recognition. **ArXiv**, [*S. l.*], 20 out. 2020. Disponível em: https://arxiv.org/abs/2010.10504. Acesso em: 20 ago.