Project: COVID-19 Assisted Diagnosis

Carlo Heemeryck[1] Willem Van Nieuwenhuyse[1] Seppe Vanrietvelde[2] Luca Visser[2]

[1] Bachelor of Science in de ingenieurswetenschappen - werktuigkunde-elektrotechniek
[2] Master of Science in Statistical Data Analysis - Computational Statistics

I. Introduction

Adds context to your project (referencing the material in your bibliography is important [**?**], [**?**]) and presents the content of the following sections (starting with Section **??**).

Includes tables with quantitative results (Table **??**) and images (Fig. **??**) from your project while carefully explaining their meaning and how you produced them.

TABLE I
Table Type Styles

Table Head	Table Column Head		
	Table column subhead	*Subhead*	*Subhead*
copy	More table copy[a]		

[a]Sample of a Table footnote.

Fig. 1. Example of a figure caption.

II. Task 1: Data Exploration, Pre-Processing and Augmentation

A. Loading the data

The grayscale chest X-ray images are loaded at their original resolution of 299 × 299 pixels. The labels, either COVID or NORMAL, are taken from the folder names containing the images and assumed to be accurate. From the outset, the dataset was pre-divided into training (1,600 images), validation (400 images), and test (200 images) sets. This original distribution is preserved for the exercise, with an additional combined training and validation set comprising 2,000 images.

B. Data exploration

Exploration consists of checking the shape of the data, evaluating the distribution of the two classes, plotting a few examples and checking pixel and global average and standard deviation. This is replicated for training, validation and test datasets and serves as an initial look at the data and to possibly detect problems at an early stage in the project. All the images in the training dataset are of the same size and there are an equal amount in both classes, which is good. A few labeled examples are given in figure **??**.

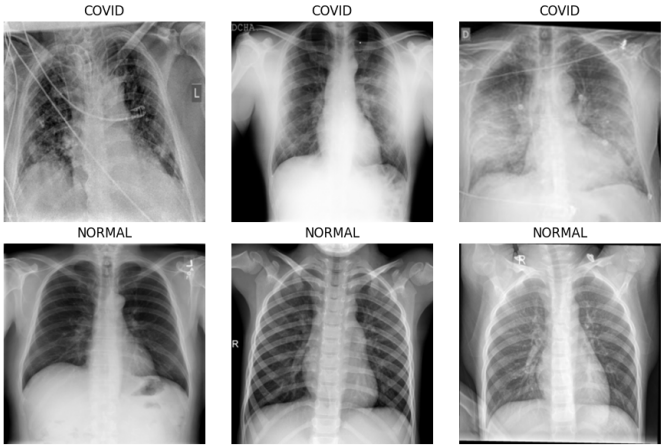

Fig. 2. A few images from the training dataset with accompanying labels.

There seem to be no clear differences between the NORMAL and COVID images. However, the limitation to grayscale images, low contrasts, different zoom levels and slight variations in angles could be problems for classification. Also if there are, to us unrecognisable, artifacts in mostly one category, generalization to new images could be problematic. Over all images in the training dataset the average value and the standard deviation at every pixel location can be visualized to show a sort of average image and a heatmap for variation. These visualizations, together with the average and standard deviation of all pixels, are given in figure **??**.

Replicating all this for the validation and test datasets give very comparable results, indicating that the images were uniformly divided over the different sets. The exploration of the validation and test sets can be consulted in the notebook.

C. Pre-processing

Next, some pre-processing steps are implemented and evaluated, namely down sampling to 128 x 128 pixels using bilinear interpolation and a few normalization strategies. The result of down sampling, given in figure **??**, is still recognisable to the human eye. As such, it is assumed that the model still gets enough information from this. Implementing this down sampling strategy should reduce training time while not losing too much performance.

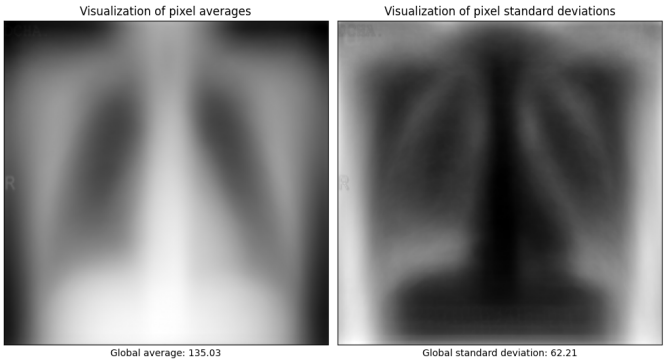

Fig. 3. A visualization of the average image (left) and a variation heatmap (right) with accompanying global statistics from the training dataset.

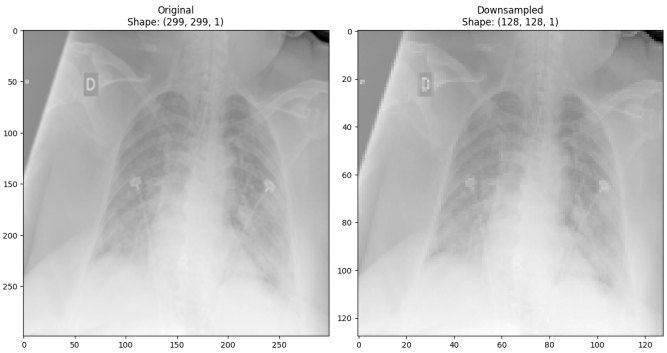

Fig. 4. An example of a downsampled image using bilinear interpolation (299x299 -¿ 128x128).

Normalization through the use of a fixed value (dividing by 255), sample statistics (subtracting the overall mean and dividing by the overall standard deviation of the current sample (training, validation or test)) and statistics of the training dataset (subtracting the overall mean and dividing by the overall standard deviation of the training dataset) were tried out. All of these strategies resulted in visually the same image but with pixel values with a far lower magnitude. However, normalizing using training set-wide image statistics (mean and std) and applying the same normalization across all images is the most correct method as it causes no data leakage. Normalization based on the training dataset will thus be used in the project.

D. Augmentation

Finally, based on the variety of the plotted images a few augmentation strategies are implemented, namely randomly altering the brightness and the contrast by a factor between 0.6 and 1.4 and randomly rotating the images (anti)clockwise up to 30°. Examples of these augmentation are given in figure **??**.

Augmentation should only result in images that could occur in test/unseen data. Based on the data we have, intense augmentation doesn't seem al that necessary. It will thus only be considered if necessary.

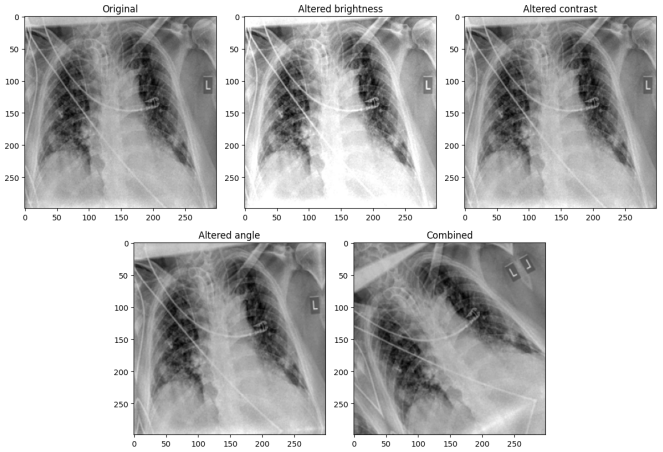

Fig. 5. Examples of different transformations as well as the combination of them together.

E. Pipeline

A pipeline consisting of loading the images correctly, with optional downsampling, followed by optional normalization using the training dataset is implemented for later use. Data augmentations can be used as layers in the models if necessary.

III. Task 2: Building the Baseline Model

This section describes the construction, training, and evaluation of our initial convolutional neural network (CNN) model for COVID-19 detection from chest X-ray images. We start by outlining the model architecture, proceed to training hyperparameters and procedures, and conclude with performance analysis on the validation and test sets.

A. Initial Model Architecture

We implemented a custom CNN with the following components:

- **Input layer**: $128 \times 128 \times 1$ grayscale image
- **Convolutional blocks**: three blocks with increasing filter sizes $\{16,\ 32,\ 64\}$, each block consists of a 3×3 convolution, ReLU activation, max-pooling, and dropout (rate 0.1)
- **Fully connected layer**: 512 units with ReLU activation and dropout (rate 0.1)
- **Output layer**: single unit with sigmoid activation for binary classification

The model uses the Adam optimizer with a learning rate of 10^{-3} and binary cross-entropy loss. Accuracy was chosen as the primary metric. The full architecture summary is shown in Table **??**.

B. Training Procedure

Training was conducted with the following settings:

- Batch size: 32
- Epochs: 30 (with early stopping patience of 2 on validation loss)

TABLE II
INITIAL BASELINE CNN ARCHITECTURE OVERVIEW

Layer (type)	Output Shape	# Params
Conv2D (3×3, 16 filters)	(None, 128, 128, 16)	160
MaxPooling2D (2×2)	(None, 64, 64, 16)	0
Dropout (rate = 0.1)	(None, 64, 64, 16)	0
Conv2D_1 (3×3, 32 filters)	(None, 64, 64, 32)	4,640
MaxPooling2D_1 (2×2)	(None, 32, 32, 32)	0
Conv2D_2 (3×3, 64 filters)	(None, 32, 32, 64)	18,496
MaxPooling2D_2 (2×2)	(None, 16, 16, 64)	0
Dropout_1 (rate = 0.1)	(None, 16, 16, 64)	0
Flatten	(None, 16·16·64 = 16384)	0
Dense (512 units)	(None, 512)	8,389,120
Dense_1 (1 unit, sigmoid)	(None, 1)	513
Total parameters		**8,412,929**
Trainable parameters		**8,412,929**
Non-trainable parameters		**0**

- Data generators: real-time data augmentation including random rotations (up to 30°), brightness and contrast adjustments in [0.6, 1.4]

The learning curves are depicted in Figure **??**.

Fig. 6. Training and validation accuracy and loss for the baseline CNN over 30 epochs.

C. Hyperparameter Tuning

A grid search was performed over the following hyperparameters:

- Dropout rate: {0.1, 0.2}
- Initial convolutional filters: {8, 16}
- Learning rate: {1e-3, 5e-4}
- Dense units: {256, 512}

Early stopping restored the best weights. The best configuration achieved a validation accuracy of **0.8958** with dropout rate **0.1**, **16** filters, learning rate $\mathbf{5 \times 10^{-4}}$, and **256** dense units. Full results are in Table **??**.

D. Final Baseline Model

Using the best hyperparameters, the model was retrained on the full training set (2,000 images) for 8 epochs. Evaluation on the held-out test set (200 images) yielded:

- Test accuracy: 0.8900
- Test loss: 0.2772

The confusion matrix is shown in Figure **??**.

TABLE III
HYPERPARAMETER TUNING RESULTS (SORTED BY VAL ACC)

Dropout	Filters	LR	Dense Units	Val Acc	Epoch	Val Loss
0.1	**16**	**5e-4**	**256**	**0.8958**	**11**	**0.3090**
0.1	8	1e-3	512	0.8958	5	0.2937
0.2	16	1e-3	512	0.8880	4	0.3031
0.1	16	1e-3	512	0.8828	6	0.2919
0.2	16	5e-4	256	0.8828	8	0.3199
0.1	16	5e-4	512	0.8724	5	0.3057
0.1	8	5e-4	256	0.8724	5	0.2813
0.1	8	5e-4	512	0.8724	7	0.3119
0.2	8	5e-4	512	0.8698	8	0.3170
0.1	8	1e-3	256	0.8672	5	0.3261
0.2	16	1e-3	256	0.8672	8	0.3626
0.2	16	5e-4	512	0.8672	10	0.3336
0.2	8	1e-3	256	0.8646	6	0.3521
0.2	8	5e-4	256	0.8594	11	0.3652
0.1	16	1e-3	256	0.8385	6	0.3685
0.2	8	1e-3	512	0.8385	6	0.3605

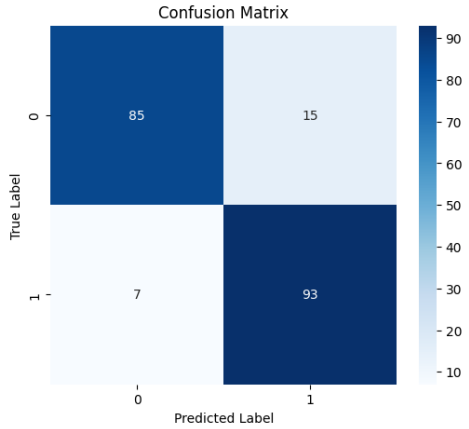

Fig. 7. Confusion matrix of the final baseline model on the test set.

E. Discussion

The baseline model achieved 89% accuracy, demonstrating strong discriminative power between COVID-19 and normal chest X-ray images. From the confusion matrix (Fig.**??**), we observe:

- True negatives (NORMAL correctly classified): 85
- False positives (NORMAL misclassified as COVID): 15
- False negatives (COVID misclassified as NORMAL): 7
- True positives (COVID correctly classified): 93

This yields a sensitivity (recall for COVID) of $93/(93+7) = 0.93$ and a specificity of $85/(85+15) = 0.85$. The slightly higher sensitivity indicates the model is more likely to correctly detect COVID cases, at the cost of some false alarms.

Training curves in Fig.**??** show rapid convergence of training accuracy to near 100% within the first 10 epochs, while validation accuracy plateaus around 88%, suggesting some overfitting that was controlled via dropout and early stopping.

Overall, the baseline CNN provides a solid starting point;

IV. Task 3: Transfer Learning

Here we will apply transfer learning by using a pre-trained ResNet50V2 model trained on Imagenet. This model has been trained on a large dataset images. We will use this model as a starting point for our own model. Transfer learning is a powerful machine learning technique for leveraging knowledge learned from one task and applying it to a new task. This is especially useful when you have limited training data.

The typical workflow for transfer learning is as follows:
1.Starting with a pretrained model
2.Freeze the layers of the pretrained model
3.Add new trainablelayers to the pretrained model
4.Training on the dataset
5.Fine-tuning (optional)

We freeze the layers of the pretrained model to prevent information learned from the initial task from being overwritten during the new training process.Then new layers are added to the pretrained model to adapt it to the new task. And then we train the model and fine-tune it.

A. Setup the base model

We instantiate the base model with the "imagenet" pre-trained weights. We do not include the top layers. The model is ResNet50V2. We freeze al the layers of the base model so they are not updated during the training process.

New layers are added to the model. explain layers.

Finally a dropout layer is added before the new layers. The Adam optimizer is used with a learning rate of .. and the loss function is binary cross-entropy. Now we compile the model.

print summary of model.

We also have a augmentation layer. We use RandomBrightness, RandomContrast and RandomRotation to augment the images.

B. Train the model

We train the model using the best hyperparameters found while optimizing the baseline. We use the training and validation data together as training data

best Hyperparameter

Here are the plots we get from training the model for trainingset and the validationset

plot for trainingset
plot for validationset

conclusion

C. Hyperparameter tuning

Now we implement the EarlyStopping callback. While tuning a few parameters. We tune batch$_size, learning_rateanddropout_rate.Wedothisbyconductingarough$

After the parameter search, we fine-tune the entire model using the the best hyperparameters we found. Which are

best Hyperparameter

D. Train pre-trained model

Now we train the model using the best hyperparameters found while fine-tuning.
By doing this we get the following plots which represent the training accuracy and loss for the pre-trained model over ... epochs.

E. Fine-tuning the entire model

In this step all of the base model layers are unfrozen. The whole model is retrained end to end. We also use the training and validation data together as training data

We display the confusion matrix.

confusion matrix

Finally a few images are plotted from the test dataset. First without pre-processing, second with their evaluations after pre-processing.

example images

V. Task 4: Explainability through Grad-CAM

VI. Conclusions

VII. Author contributions and collaboration

VIII. Use of Generative AI

A. Figures and Tables

a) Positioning Figures and Tables:

References

Example References:

References

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, "Title of paper if known," unpublished.
[5] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.