# CNN for Pneumonia detection

Edoardo Monnetti
Università degli Studi di Torino
Dipartimento di Fisica
edoardo.monnetti@edu.unito.it

July 15, 2022

## Abstract

Machine learning has a phenomenal range of applications, including in health and diagnostics. The task of the project is to realize a Convolutional Neural Network aimed to recognize the presence of pneumonia in a patient, starting from the analysis of an X-ray scan. This is especially useful during these current times as COVID-19 is known to cause pneumonia.

## 1 Task

In this work the procedure to build an X-ray image classification model with a simple Convolutional Neural Network (CNN) is presented. The task of the network is to predict whether an X-ray scan shows presence of pneumonia. The network was trained with a dataset of X-ray images showing both pneumonia and normal lungs. Since the are only two possible outcomes, a binary classification network is the best choice.

## 2 Method

In order to implement the network, *TensorFlow* [1] and *Keras* [2] libraries were used in the Jupyter Notebook environment. The code is written in Python as it is the main programming language for this kind of applications.

### 2.1 Dataset & preprocessing

The dataset was taken from *Kaggle.com* [3]. It contains 5856 clinical X-ray scan images divided in three folders, namely training, test and validation. Since there are only 16 files in the validation folder, some of the images were manually moved from the training folder to the test and validation folders in order to balance the proportions. This because a less extreme division between the training and the validation set is preferred. The final percentages chosen for validation and test sets are 17% and 7% respectively. In addition the number of samples for each class in the training set is unbalanced:

- 3476 for pneumonia
- 942 for normal.

For this reason it is important not only to evaluate the accuracy of the model but also other metrics like precision and recall.

The dimensions (in pixels) of the images are all different but a rescaling process is applied before the input of the CNN in order to reduce and uniform the dimensions of the images. The final dimensions in pixel are 150 x 150. Since the dataset has only 4000 images for the training and unbalanced labels, an artificially expansion of the dataset is needed. In this regard to avoid overfitting problems a data augmentation process is performed. It makes random horizontal flips, rotations and zoom of the images of the dataset to increase the diversity of the training set. The images have only a grayscale channel for each pixel with values between 0 and 255. The pixel values are rescaled to fit the interval [0, 1] in order for the network to converge faster. The dataset is finally divided into batches of 32 elements.

### 2.2 Model

A sequential type model is used. The code is taken from a notebook related to the dataset with a modified architecture and some different parameters such as the optimizer and regularizer [4]. After some training attempts, changing the number of filters and convolutional layers, the most performing architecture of the network is displayed in Table 1.

The activation function is the same for all the layers (*ReLU* [5]) except for the output layer which has a sigmoid activation function. The first layer is the input layer which takes each 150 x 150 pixel values. Then three convolutional blocks are stacked. A convolutional block is made by a proper convolutional 2D layer with its number of filters, a batch normalization layer which normalizes its inputs and a max pooling layer.

| Layer type | Output shape | Filter size | Stride |
|---|---|---|---|
| Input layer | 148x148x16 | 3x3 | 1 |
| MaxPool2D | 74x74x16 | - | 1 |
| Conv2D | 72x72x32 | 3x3 | 1 |
| BatchNormalization | 72x72x32 | - | - |
| MaxPool2D | 36x36x32 | - | 1 |
| Conv2D | 34x34x16 | 3x3 | 1 |
| BatchNormalization | 34x34x16 | - | - |
| MaxPool2D | 17x17x16 | - | 1 |
| Dropout (0.5) | 7x7x16 | - | - |
| Flatten | 784 | - | - |
| Dense | 64 | - | - |
| Dropout (0.3) | 64 | - | - |
| Dense | 1 | - | - |

**Table 1.** Sequential model architecture

The batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

The max pooling layer reduce the number of features of the previous layer as it take only the maximum value in a matrix of 2 by 2 pixels. After the last max pooling layer there is a dropout layer, which randomly sets input units to 0 at each step during training time with a certain frequency: this will help prevent overfitting.

After that there is a flatten layer which collapses the spatial dimensions of the input into a vector and the last layers of the model are fully connected dense layers with respectively 64 and 1 units, with the last one corresponding to the binary classification. Before the output layer there is another dropout layer with frequency 0.3.

In this implementation the model was trained using the Adam [6] optimizer and the Binary Crossentropy loss function. Adam is an optimization algorithm that can be used in place of the classical stochastic gradient descent to iterative update network weights based on the training data. It is a composition of two optimization methods called *Root Mean Square propagation* (RMSprop) [7] and *Exponentially Weighted Moving Average*. Adam is different to classical stochastic gradient descent as it does not maintain a single learning rate (termed $\alpha$) for all weight updates and the learning rate does change during training. The magnitude of this changing in the learning rate is leaded by other two parameters, namely $\beta_1$ and $\beta_2$. Another optimizer used is a keras function that reduces the learning rate when the chosen metric has stopped improving called *"ReduceLROnPlateau"* [8].

Since the dropout layers couldn't fix the overfitting problem, a regularizer is then applied to reduce the accuracy on the training data promoting a greater generalization capacity on the test set. In particular an L2 regularizer was chosen [9].

During the training an early stopping algorithm stops the learning process as soon as the model starts becoming stagnant, or even worse, when the model starts overfitting.

## 3 Results

Before settling on the model's architecture described in Table 1, several configuration were tested. Concerning the number of layers and their filters the net performs very well under 500'000 parameters in terms of accuracy[1]. The first term that substantially affects the number of parameters computed by the neural network is the number of neurons in the dense layer before the output: 64 elements is a good trade-off between training time and generalization capacity without sacrificing accuracy.

In Table 2 a comparison between the actual model and another one with 4 layers is showed. Among the 4 layered architectures tried out, this one is the most performing one.

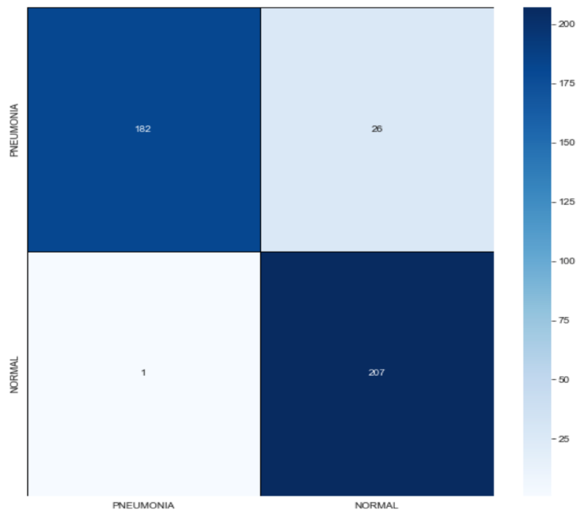
Looking at the confusion matrices of these two models (Fig. 1) it is clear that in both cases the model gets the normal guesses wrong more often. This translates in high precision and low recall for pneumonia case and vice versa for normal case (see Appendix, Table 5). This is probably due to the imbalance of samples for each class in the training set.

The model with 3 layers performs better so it was trained again adding the early stopping feature to the `model.fit` function. Since `restore_best_weights` is set to "True", the returned model at the

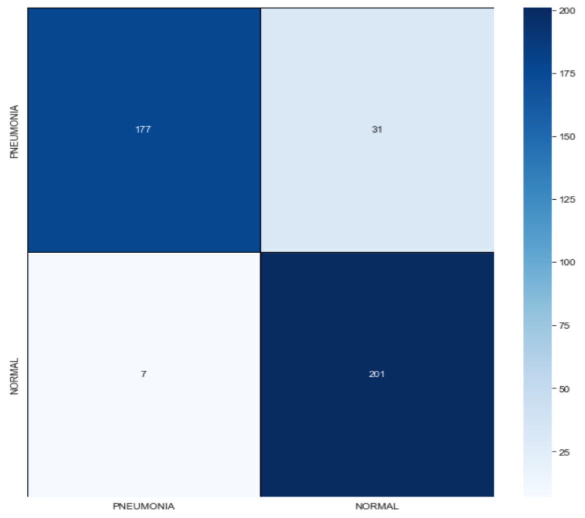
[1]The accuracy here mentioned is calculated on the test set, so on images never seen by the network

| Architecture | Accuracy | Precision | Recall | F1 score |
|---|---|---|---|---|
| 16x32x16 | 0.94 | 0.89 | 0.99 | 0.94 |
| 32x64x128x32 | 0.91 | 0.87 | 0.97 | 0.91 |

**Table 2.** Different performance metrics for the two models calculated on the test set after a training on 25 epochs



**(a)** Confusion matrix for the 16x32x16 model



**(b)** Confusion matrix for the 32x64x128x32 model

**Figure 1**

end of the training process will be the model with the best weights (i.e. low loss and high accuracy). The final performances of the model are listed in Table 3.

| Dataset | Loss | Accuracy |
|---|---|---|
| Training | 0.18 | 94.84 % |
| Validation | 0.23 | 92.79 % |
| Test | 0.27 | 93.50 % |

**Table 3.** Loss and accuracy for the three different datasets referred to the model with three convolutional layers 16x32x16.

Then the model was trained again for a period of 100 epochs to check if overfitting would happen. Data are presented in Fig. (2)

The model is quite stable over a longer period of training time and the overall performances does not change very much. Even though the validation accuracy stays under the 90 % during the entire training process, the network accuracy on the test set exceed the 93 %. This is a common trend with the other architecture shown in Table 1 (see Appendix).

### 3.1 Finetune the model

Being both precision and recall metrics a measure of the confidence of the classification for each class, in a real life application it could be preferred to promote a high precision for the normal class and high recall for the pneumonia class, leading to less wrong (and potentially dangerous) diagnosis. This translates in a lower value of the upper right corner in the confusion matrix. Looking at the previous results the situation is right the opposite, but it is worth trying to reduce the number of false negative (normal cases classified in the wrong way).

Taking into account the unbalance between the two classes a weight for each class is calculated as follows:

$$\text{weight}_i = \frac{1}{2}\frac{n_i}{N_{tot}}$$

where $n_i$ is the number of samples of that class, $N_{tot}$ the total number of training samples and $i = 0, 1$ counts for the classes. The resulting weights are:

- $\mathrm{w}_0 = 2.35$
- $\mathrm{w}_1 = 0.64$

The weight for class 0 (normal) is a lot higher than the weight for class 1 (pneumonia). Because there are less normal images, each normal image will be weighted more to balance the data as the CNN works best when the training data is balanced.

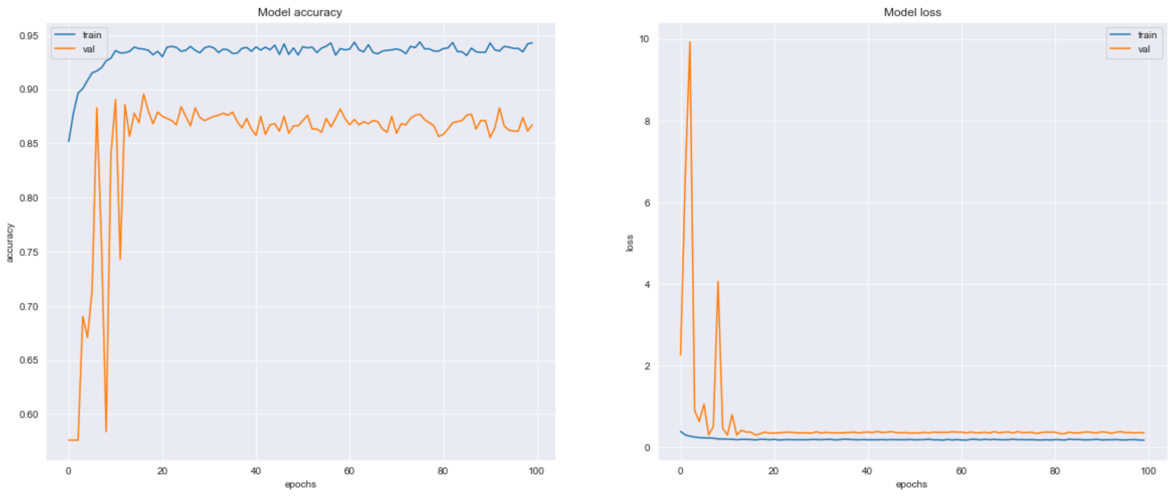


**Figure 2:** Training and validation accuracy and loss over 100 epochs.

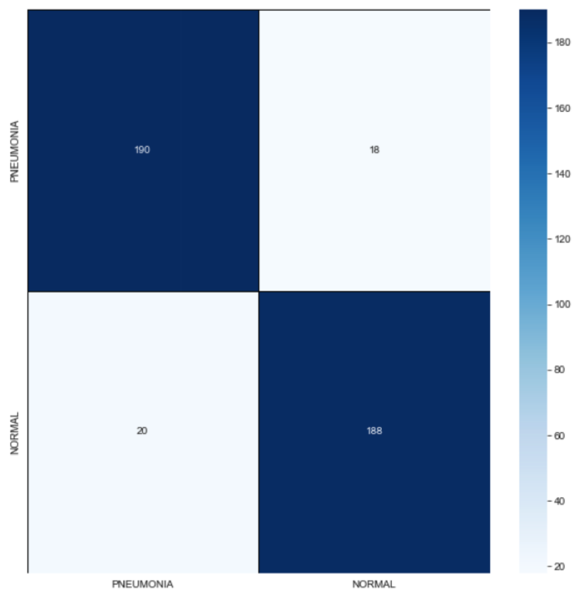


**Figure 3:** Confusion matrix for the model trained with class weights.

Considering this weights in the `model.fit` function the training is launched another time. From Table 4 it can be seen that including class weights the accuracy and precision are lower because there are more false positives, but conversely the recall is higher because the model also found more true positives. Despite having lower accuracy, this model has higher recall (and identifies more pneumonia cases) reducing the number of false normal cases identified, as wished before. This is further made clear by looking at the confusion matrix showed in Fig. (3): comparing the values of the confusion matrix in Fig. (1a) there are less false negative and more true positive.

| | Classes | |
|---|---|---|
| | Pneumonia | Normal |
| **Accuracy** | 0.91 | |
| **Precision** | 0.96 | 0.87 |
| **Recall** | 0.85 | 0.97 |
| **F1 score** | 0.90 | 0.91 |

**Table 4.** Network performances with class weights added.

## 4 Conclusions

Despite having lower performances on the validation set during the training, the model seems to predicts very well the labels of the test set, maintaining the accuracy even over longer period of training like the 100 epoch run. Even with only two hidden layers the net performs better compared to the one with an additional hidden layer. The unbalance in the dataset does not seem to heavily affect the overall accuracy of the network, while changes a little bit precision and recall metrics. In this regard a data augmentation process could be carried out targeting only the normal classes, instead of weight classes. The difference between the model with class weights and the one without is not striking but with a larger test set this could be hopefully more evident. Nonetheless it leaves some space to further improve the capability of the network in terms of global accuracy, since none of the other parameters were modified.

In conclusion artificial neural networks provide a powerful tool to help doctors analyze, model, and make sense of complex clinical data across a broad range of medical applications. Since the most applications of artificial neural networks in medicine are classification problems, a network like the one presented here could help in a faster recognition of potentially dangerous diagnosis. This is true not only for X ray images but for almost any clinical image, provided the right dataset, and a lot of effort is spent trying to apply this kind of tool to cancer diagnosis.

## References

[1] **Tensorflow®: Large-Scale Machine Learning on Heterogeneous Systems**, `https://www.tensorflow.org`

[2] **Keras**, Chollet, Francois et al. 2005, `https://github.com/fchollet/keras`

[3] **Chest X-Ray Images (Pneumonia)** `https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia`

[4] **Pneumonia Detection using CNN** `https://www.kaggle.com/code/madz2000/pneumonia-detection-using-cnn-92-6-accuracy`

[5] **Deep learning using rectified linear units (relu)**, Agarap, A. F., 2018, ArXiv Preprint ArXiv:1803.08375

[6] **Adam: A Method for Stochastic Optimization**, Diederik P. Kingma, Jimmy Ba, 2015, ICLR (Poster) 2015

[7] **Coursera Neural Networks for Machine Learning lecture 6**, Geoffrey Hinton, 2018.

[8] **ReduceLROnPlateau** `https://keras.io/api/callbacks/reduce_lr_on_plateau/`

[9] **Ridge Regression: Biased Estimation for Nonorthogonal Problems**, Arthur E. Hoerl & Robert W. Kennard, 1970, Technometrics, DOI: 10.1080/00401706.1970.10488634

## 5 Appendix

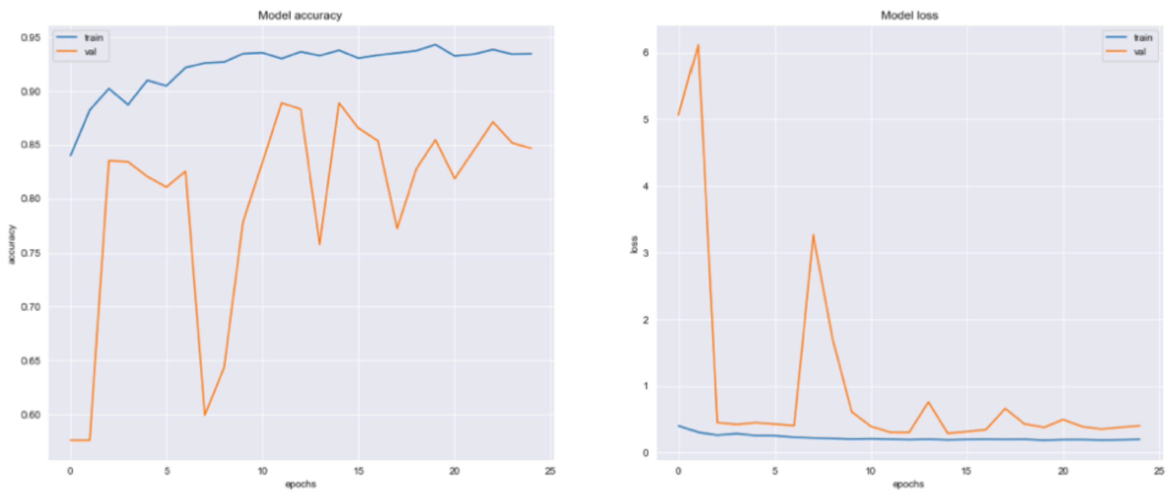


**Figure 4:** Training and validation accuracy and loss.
Early stopping, 16x32x16

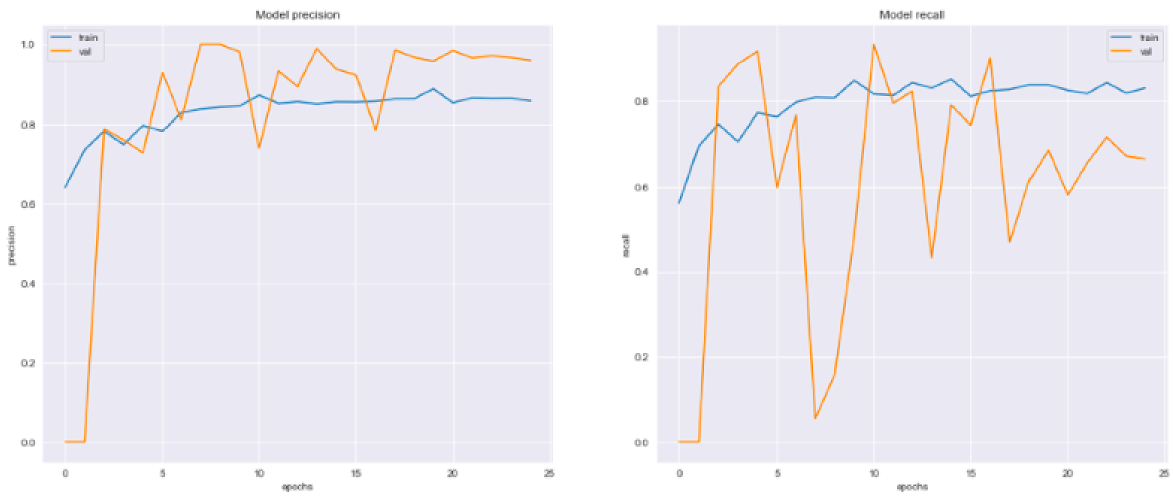


**Figure 5:** Training and validation precision and recall.
Early stopping, 16x32x16.

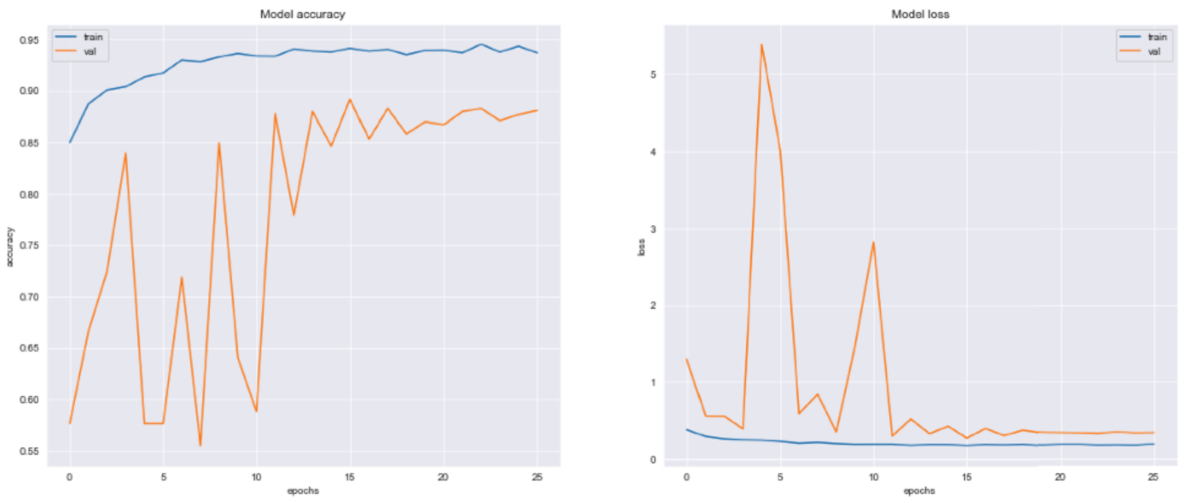


**Figure 6:** Training and validation accuracy and loss.
Early stopping, 32x64x128x32.

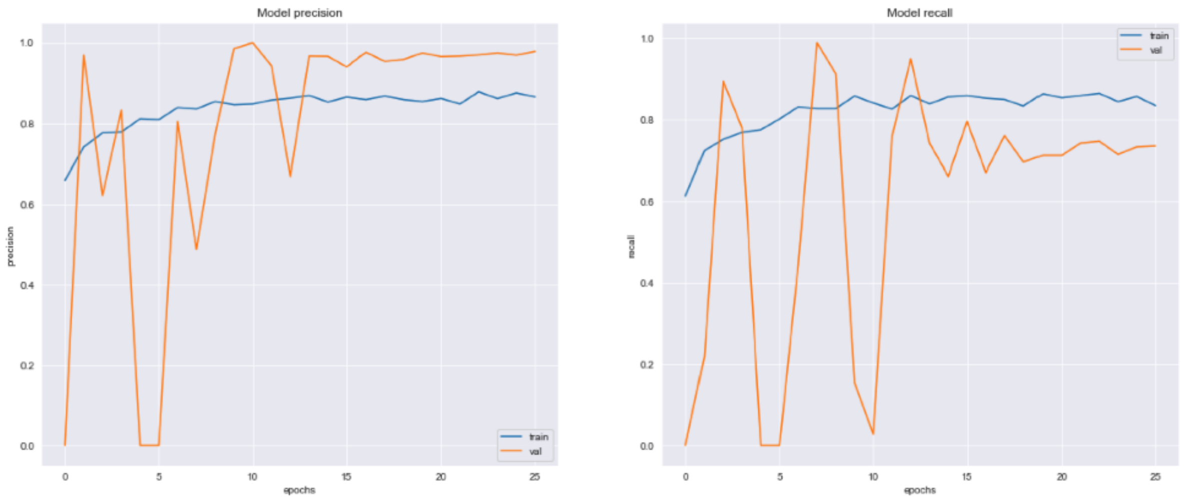


**Figure 7:** Training and validation precision and recall.
Early stopping, 32x64x128x32.

| | 16x32x16 | | 32x64x128x32 | |
|---|---|---|---|---|
| | Precision | Recall | Precision | Recall |
| Pneumonia | 0.99 | 0.88 | 0.96 | 0.85 |
| Normal | 0.89 | 0.99 | 0.87 | 0.97 |

**Table 5.** Precision and recall metrics for each class of the two model compared in Table 2.
Early stopping.

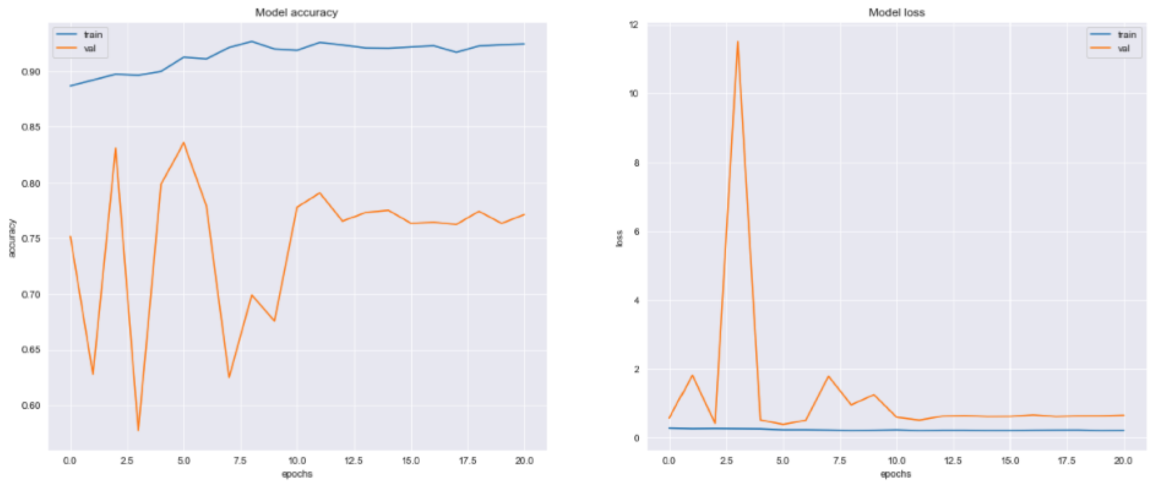


**Figure 8:** Training and validation accuracy and loss.
Early stopping, 32x64x128x32 with class weights.

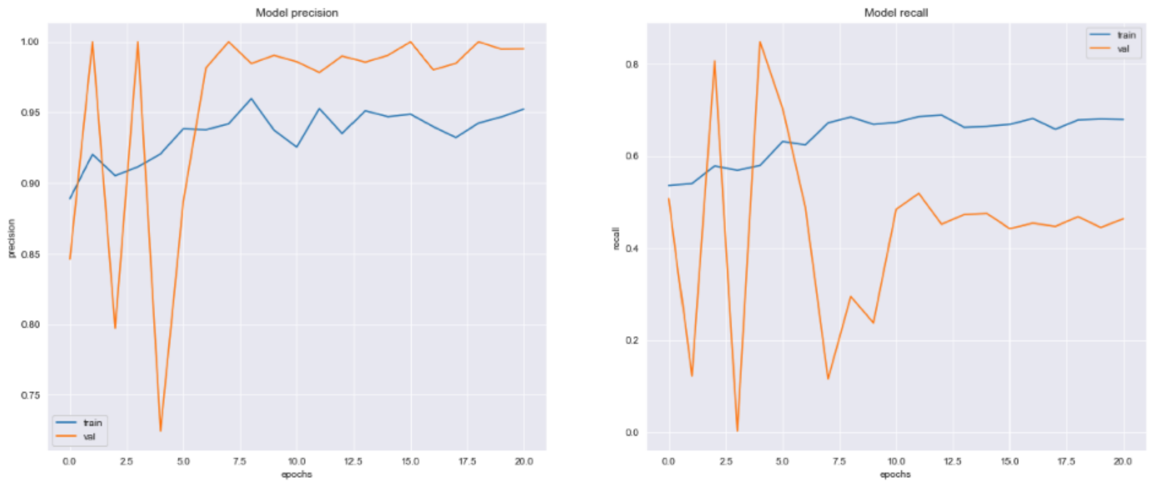


**Figure 9:** Training and validation precision and recall.
Early stopping, 32x64x128x32 with class weights.