Medical Image Multi-Label Classification

Ariton Cosmin

TensorFlow library with Keras

I was stunned by the ease of use of TensorFlow and keras libraries, setting up a basic model taking a very short time.

TensorFlow is great at being a low-level library for numerical computations. It proved quite helpful in computing the mean F1 score and the precisions of the model. Although, at first, the tensor operations seemed slow, but that was due to bad programming by not taking into account the GPU architecture. It should be more clearly stated that instruction forks like “if” really harm the performance. The reason behind the loss of performance is that GPUs, or at least mine, has a single instruction decoder, but many cores that can perform computations in parallel. The introduction of forks means that some cores will run while others will wait for the instruction decoder to become available. All such forks were removed with some clever use of matrix operations to compute the necessary metrics. All this code can be found in custom_metrics.py file.

Keras is a way higher-level library, built on top of TensorFlow, which is very easy to use, but at the same time incredibly powerful. A lot of models were tried with ease, but manually. Being such a high-level library, restricts some of the fine control you have over the code. I have spent a lot of hours trying to train more models in a row, with randomized hyper parameters so I don’t have to tune them manually, but ended up failing due to resources being unallocated properly. Some fixes were found, like using the multiprocessing module from python, to force a deallocation of memory, but time was short and the idea was scrapped. Grid search was considered and random search were considered, but due to poor equipment and long training times, a more hands down approach was used by manually picking reasonable architectures.

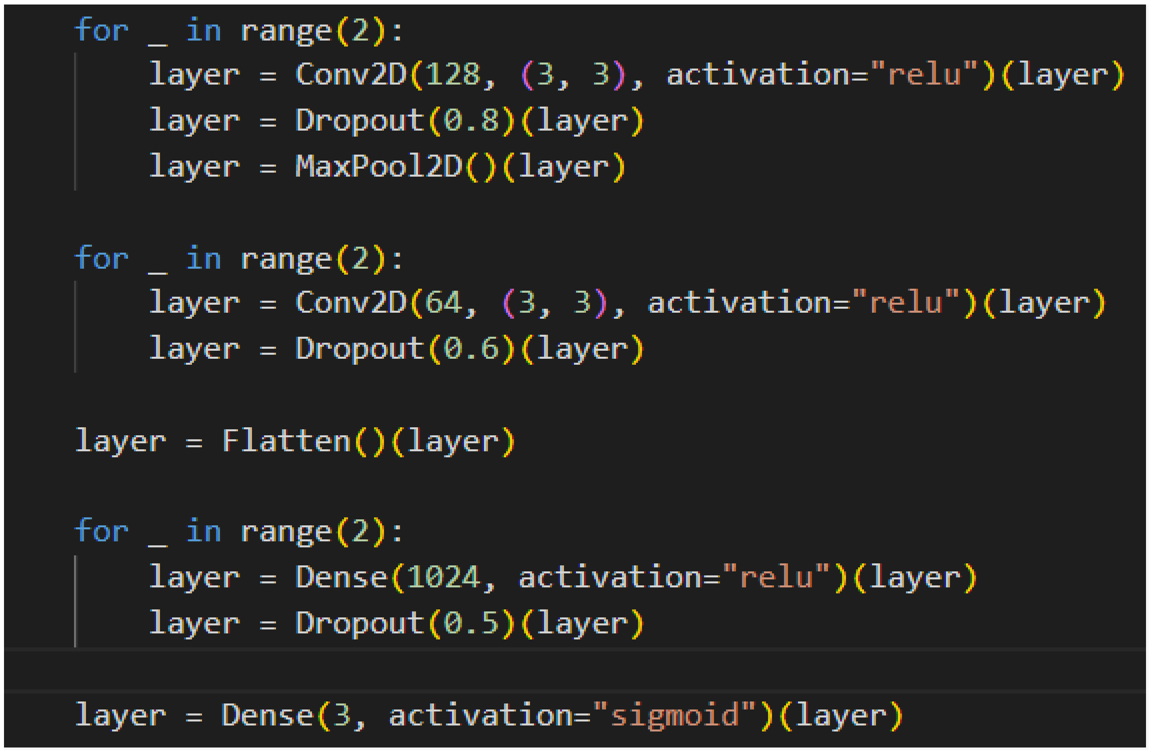
Used models

Classic CNNs

This model showed a lot of success even from start, without any tuning. It is well known that CNNs are particularly well-suited for image classification tasks. After a little research online, I have found that CNNs have been used for a variety of medical image classification tasks, including detecting abnormalities in X-rays and identifying cancerous cells in microscopy images, or pockets of air in CT scans. Also CNNs architectures like U-Net have been very successful in segmentation tasks for medical images which can help in predicting volumes of tumors or other objects of interests. Having that in mind, this was the first model I tried.

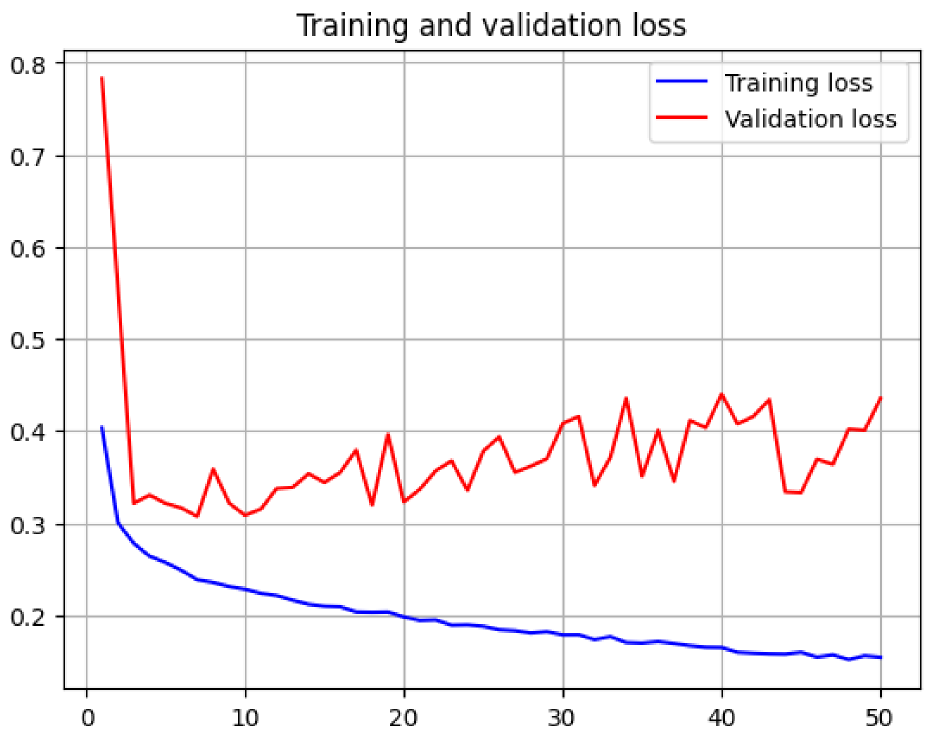
Although I searched for many architectures, most of them performed similarly with some exceptions where too few convolutional filters and layers were used and the model was underfitting. It is to be noted that the model also overfitted while using too many layers or filters, but that was a much rarer case due to hardware limitations.

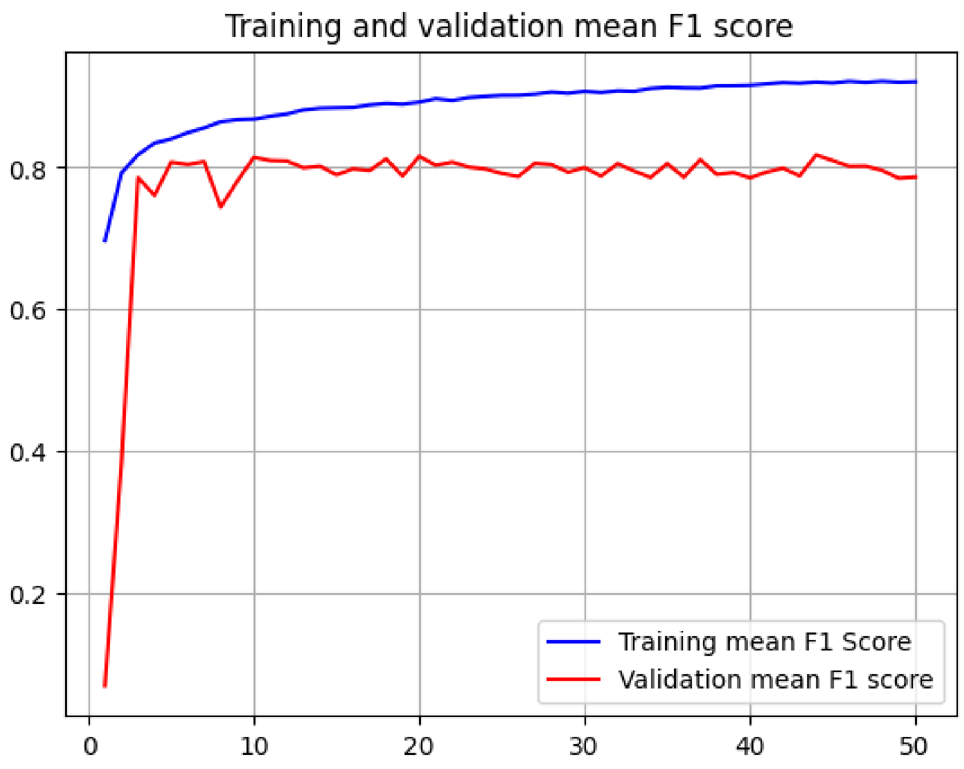
One of the best performing architecture found is the following:



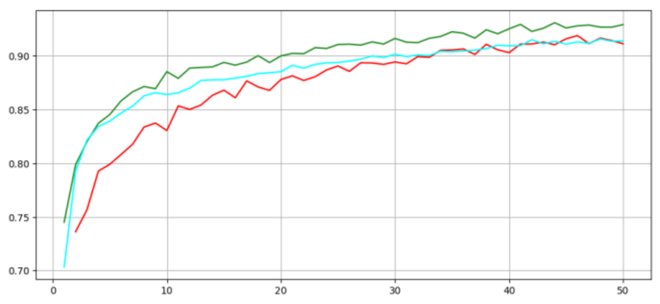
```
for _ in range(2):
    layer = Conv2D(128, (3, 3), activation="relu")(layer)
    layer = Dropout(0.8)(layer)
    layer = MaxPool2D()(layer)

for _ in range(2):
    layer = Conv2D(64, (3, 3), activation="relu")(layer)
    layer = Dropout(0.6)(layer)

layer = Flatten()(layer)

for _ in range(2):
    layer = Dense(1024, activation="relu")(layer)
    layer = Dropout(0.5)(layer)

layer = Dense(3, activation="sigmoid")(layer)
```

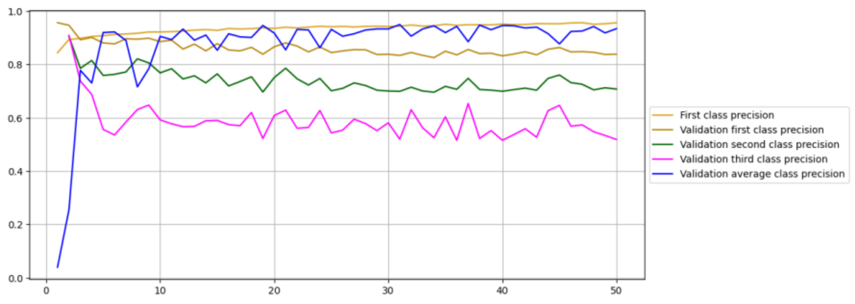
Below are presented training and validation losses and accuracies of the above model over 50 epochs:





The graph below represents the precision over the 50 epochs trained:

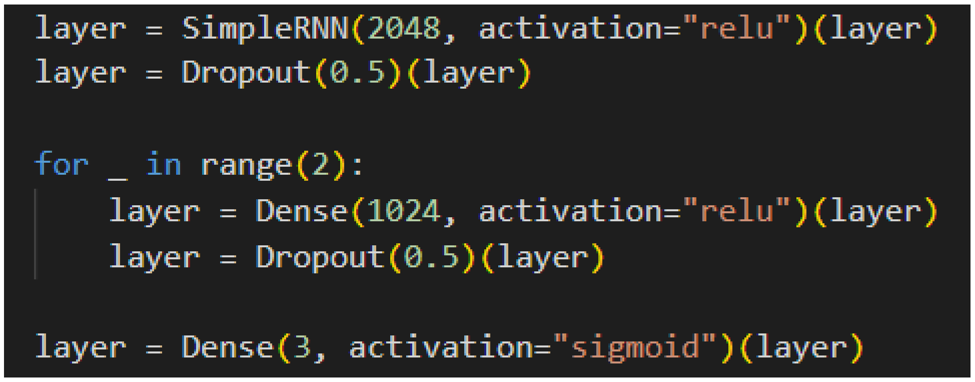




RNNs

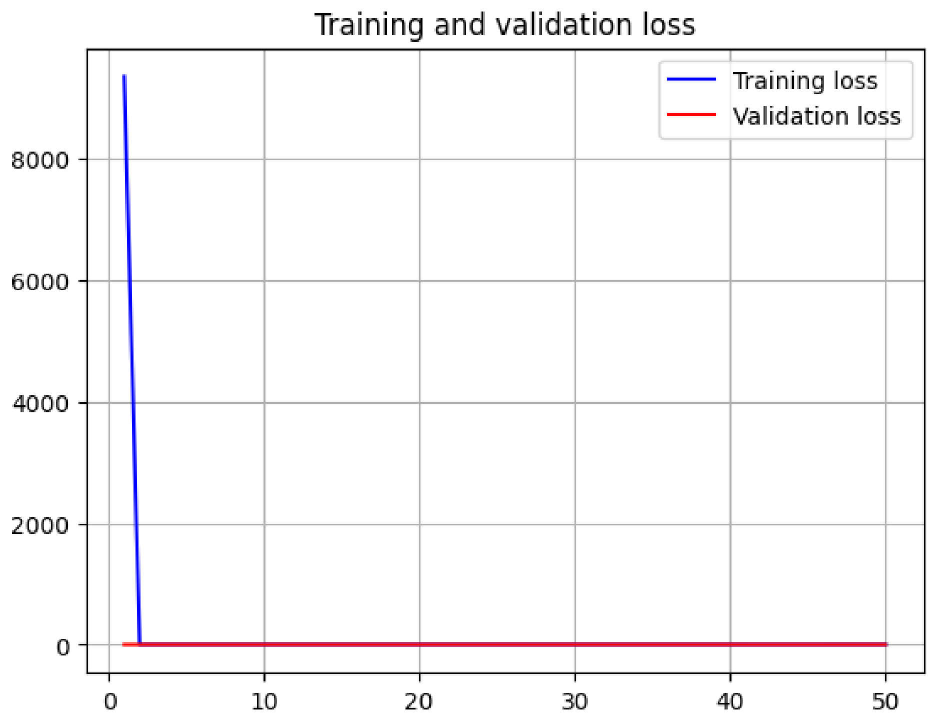
While both RNNs and CNNs have been successful in multilabel image classification, there are some considerations to keep in mind when using these models. RNNs can be more difficult to train and require more data to achieve good performance, while CNNs may be less able to capture temporal context. With all this in mind, I shallowly tried some RNNs architectures, but found poorer results on average compared to CNNs. It should be noted that the RNNs still had a good overall performance. RNNs are a type of neural network that are well-suited for processing sequential data, such as time series or natural language. With all this in mind, I stopped trying this types of models, the data not being sequential.

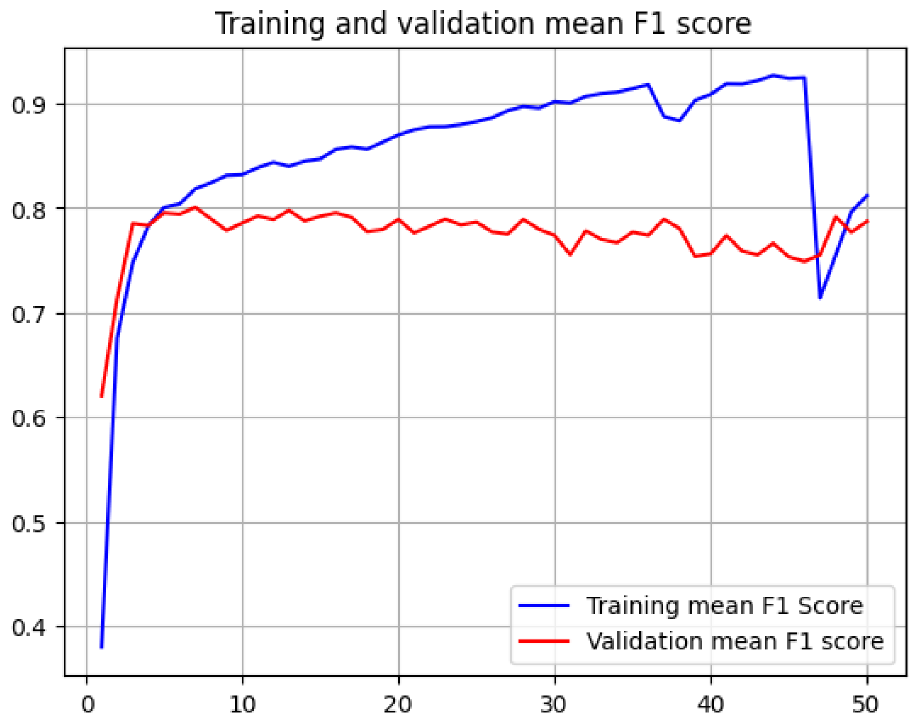
One of the best performing architecture found is the following:



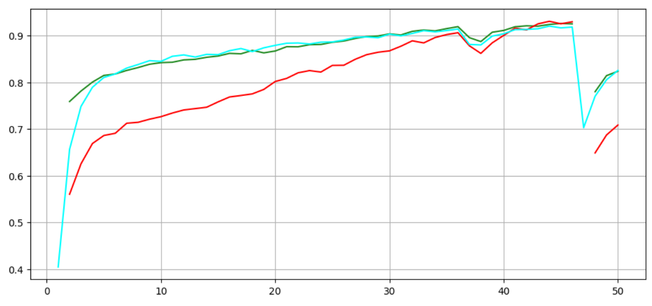
```
layer = SimpleRNN(2048, activation="relu")(layer)
layer = Dropout(0.5)(layer)

for _ in range(2):
    layer = Dense(1024, activation="relu")(layer)
    layer = Dropout(0.5)(layer)

layer = Dense(3, activation="sigmoid")(layer)
```

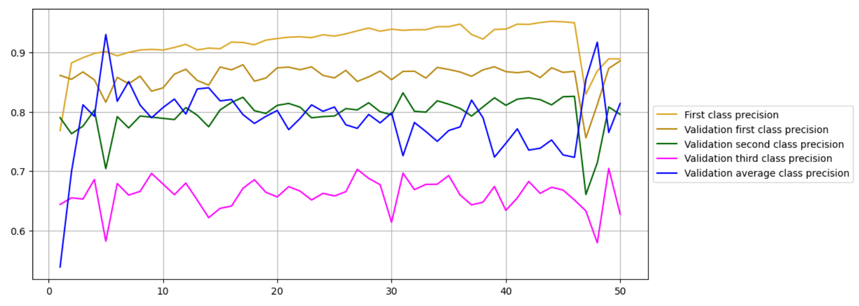
Below are presented training and validation losses and accuracies of the above model over 50 epochs:





The graph below represents the precision over the 50 epochs trained:





Preprocessing

A simple normalization technique was applied for each image: dividing each pixel by 255, the maxim value one can have.

Data augmentation

Data augmentation was not performed due to now knowing what exactly is present in the images and how it can affect them. A GAN approach was also not tried due to the same reasons.

Conclusion

Overall, RNNs and CNNs are powerful tools for multilabel image classification in the medical field. Although way more computational power is needed and time for better results.