Zackary Ducreux

P2023

Predictive modelling and unsupervised clustering

Supervised Learning Project

Introduction

Describe the dataset and task

- What is the data about, who collected it and how?

The data is about the classification of textures in colorectal cancer histology. Each class is composed of 625 images which are 150x150x3 RGB images.

Below we can see the authors who published the scientific report related to this study.

@Article {kather2016multi, title= {Multi-class texture analysis in colorectal cancer histology},

authors= {Kather, Jakob Nikolas and Weis, Cleo-Aron and Bianconi, Francesco and Melchers, Susanne M and Schad, Lothar R and Gaiser, Timo and Marx, Alexander and Zollner, Frank Gerrit},

journal= {Scientific reports}, year= {2016}, publisher= {Nature Publishing Group}}

- What is the task targeted by your experiments?

The goal of my experiments is to identify the type of disease or sanity of a person with a Machine Learning algorithm. That will allow us to certify with a degree of confidence a result that will save time to many doctors and by consequence save human lives for being treated faster. As we can see below we can import some images to see how does it work, what the diseases look like.

Summarize your objectives and what you aim to achieve with your experiments

This will be done by training a model with supervised learning that mean give the expected result like seen above. After that we will test it on a sample of images with their associated labels that the algorithm hasn't seen before. In that way we can see the accuracy of the model and improve it to the maximum we can while avoiding overfitting.

To reduce overfitting, we would use dropouts layers, data augmentation and an optimization in the selection of the layers to reach the more precise accuracy we can. This means that we need to experiment with different parameters, layers and activation functions.

Test will be made with different activations functions like Softmax, Sigmoid, ReLu on the layers of the model to see the impact of it on the accuracy.

Proposed pipeline

Briefly describe and justify the models you used and how you applied it/them to your task

- Describe what your baseline model is and the more complex models you intend to try.

My baseline model will be composed of a rescaling layer that will normalize data and specify the input shape at the very first layer. Then Conv2D layer with a ReLU activation function followed by a MaxPooling2D will be done 3 times with different values of neurons.

Describe and justify any data pre-processing you had to do

We can see by experiment that this will overfit so we could add a dropout layer that will prevent that. A layer will be flattening the input, another applies the ReLU again with 128 neurons and finally a layer will have the same number of neurons as the number of possible labelled values which will be the output of the model.

```
class_names = df_images_labels['label'].unique()
class_names
```

```
array(['TUMOR', 'STROMA', 'COMPLEX', 'LYMPHO', 'DEBRIS', 'MUCOSA',
       'ADIPOSE', 'EMPTY'], dtype=object)
```

```
num_classes = len(class_names)
```

```
model = Sequential([
    layers.Rescaling(1./255, input_shape=(150, 150, 3)),
    layers.Conv2D(16, 3, activation='relu', padding="same"),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(num_classes)])
```

Then, we must split train and test with the TensorFlow function. Then we need to apply a label encode on the labels of the 'y_train' and 'y_test' variable so that the string they represented will be transformed into a number associated with their label.

```
lb = LabelEncoder()
y_lb = lb.fit_transform(y_train)
y_lb_test = lb.fit_transform(y_test)
y__lb_validation = lb.fit_transform(y_validation)
```

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

Experimental methodology

Describe and justify the methodology used to test your pipeline

- Metrics you've evaluated

To evaluate the model, we need some criteria's that will not change through time. For that we have 2 axis which are the accuracy and the loss. The accuracy and the loss are evaluated on both train and test sets. Thus, we can see if the model is overfitting if the test loss is not following the curve of the train set and getting bigger through the epochs.

- Hyper-parameter choice criteria (which were set to the framework default, which were tuned using grid/random/other search method)

For the Hyper-parameter choice we need to run a GridSearchCV that will test every combination of parameters to give the one that gives the best accuracy on the model. For that we need to define a dictionary of parameters with values that they can take.

```
parameter_space = {
    'hidden_layer_sizes': [(10,30,10),(20,)],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
    'batch_size' : [10, 20, 40, 60, 80, 100]}
```

After that we need to define the classifier that will be used as the GridSearchCV estimator. It's a Multi-layer Perceptron Classifier with some parameters including the solver which is a Limited-memory BFGS. It approximates the Broyden–Fletcher–Goldfarb–Shanno algorithm using a limited amount of computer memory. The algorithm's target problem is to minimize the function over unconstrained values of a real vector. Alpha which is the weight of the L2 regularization term is also one of the parameters as well as the tolerance for the optimization.

```
clf = MLPClassifier(solver='lbfgs', alpha=1, tol=5e-3)
```

After that the GridSearchCV which will the best parameters for the model.

```
grid = GridSearchCV(estimator=clf, param_grid=parameter_space, n_jobs=-1)
```

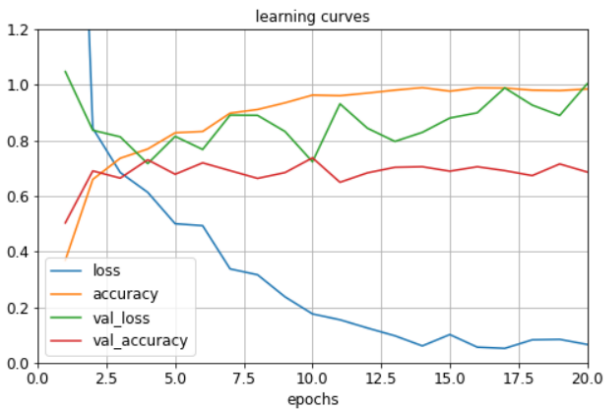
Results and discussion

Present the main relevant results of your experiments

- Use tables and/or plots to summarize the obtained performances on the train/valid/test sets

Sequential baseline model:

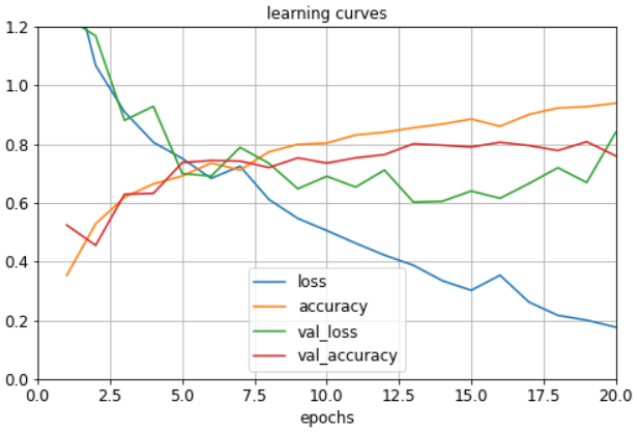
- One rescaling layer
- One Convolution 2D layers and one max pooling 2D layer (ReLu activation)
- One flattens layer
- Two dense layers (ReLu activation)
- Final dense layer




```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 150, 150, 3)       0

 conv2d_3 (Conv2D)           (None, 150, 150, 16)      448

 max_pooling2d_3 (MaxPooling  (None, 75, 75, 16)       0
 2D)

 flatten_1 (Flatten)         (None, 90000)             0

 dense_2 (Dense)             (None, 128)               11520128

 dense_3 (Dense)             (None, 8)                 1032

=================================================================
Total params: 11,521,608
Trainable params: 11,521,608
Non-trainable params: 0
_________________________________________________________________
```

Sequential developed improved model:

- One rescaling layer
- Three Convolution 2D layers and three max pooling 2D layers (ReLu activation)
- One flattens layer
- Two dense layers (ReLu activation)
- Final dense layer



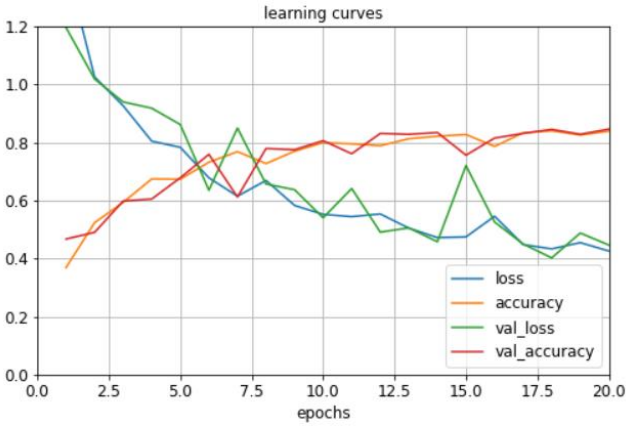
Model: "sequential_5"

Layer (type)	Output Shape	Param #
rescaling_2 (Rescaling)	(None, 150, 150, 3)	0
conv2d_4 (Conv2D)	(None, 150, 150, 16)	448
max_pooling2d_4 (MaxPooling 2D)	(None, 75, 75, 16)	0
conv2d_5 (Conv2D)	(None, 75, 75, 32)	4640
max_pooling2d_5 (MaxPooling 2D)	(None, 37, 37, 32)	0
conv2d_6 (Conv2D)	(None, 37, 37, 64)	18496
max_pooling2d_6 (MaxPooling 2D)	(None, 18, 18, 64)	0
flatten_2 (Flatten)	(None, 20736)	0
dense_4 (Dense)	(None, 128)	2654336
dense_5 (Dense)	(None, 8)	1032

Total params: 2,678,952
Trainable params: 2,678,952
Non-trainable params: 0

Sequential developed optimized model:

- One rescaling layer
- Data augmentation (Flip, rotation, zoom)
- Three Convolution 2D layers and three max pooling 2D layers (ReLu activation)
- One dropout layer (parameter = 0.3)
- One flattens layer
- Two dense layers (ReLu activation)
- Final dense layer



Model: "sequential_7"

Layer (type)	Output Shape	Param #
rescaling_3 (Rescaling)	(None, 150, 150, 3)	0
sequential_6 (Sequential)	(None, 150, 150, 3)	0
conv2d_7 (Conv2D)	(None, 150, 150, 16)	448
max_pooling2d_7 (MaxPooling 2D)	(None, 75, 75, 16)	0
dropout_1 (Dropout)	(None, 75, 75, 16)	0
flatten_3 (Flatten)	(None, 90000)	0
dense_6 (Dense)	(None, 128)	11520128
dense_7 (Dense)	(None, 8)	1032

Total params: 11,521,608
Trainable params: 11,521,608
Non-trainable params: 0

Based on your results, explain your reasoning behind changes you made to improve your performance

The Dropout layer randomly sets input units to 0 with a frequency of **rate** (which is the principal dropout parameter) at each step during training time, which helps prevent overfitting. Inputs not set to 0 are scaled up by 1/ (1 - rate) such that the sum over all inputs is unchanged. In simple terms the layer is temporarily disabling certain neurons in the network, as well as all its incoming and outgoing connections.

```
layers.Dropout(0.3)
```

This layer is used to avoid overfitting in the model by applying it at strategic times in the model.

Data augmentation was also used in the optimized developed model to increase the amount of data on which the model is training. For that we have multiple solutions.

- Geometric transformations – you can randomly flip, crop, rotate or translate images etc…
- Colour space transformations – change RGB colour channels, intensify any colour
- Kernel filters – sharpen or blur an image
- Random Erasing – delete a part of the initial image
- Mixing images – basically, mix images with one another

As we can see below, I used only the geometric transformations which are Flip, Rotation and Zoom.

```
Entrée [122]: data_augmentation = keras.Sequential(
                [
                  layers.RandomFlip(mode="horizontal_and_vertical"),
                  layers.RandomRotation(0.2),
                  layers.RandomZoom(0.1, 0.1),
                ]
              )
```

Once I determined what model was the most optimized for my problem, and what could be applied (Data augmentation, dropouts, etc.) I can now choose the activation function for the final layer. Considering the dataset, we can deduct that this is a classification problem. Thus, we have the choice between ‘SoftMax’ and ‘Sigmoid’ for the final year that have as an output one of the eight label. As we can see below the sigmoid is clearly the best activation function, it does not overfit, the loss is relatively small, and the accuracy is close to 85%.

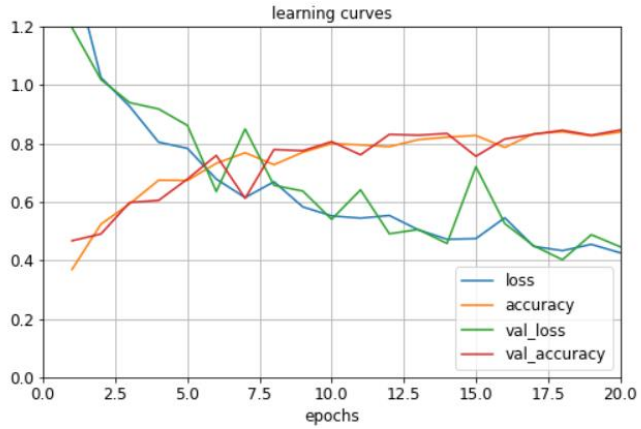


*Figure 2 : Optimized layer with **Softmax** activation function*

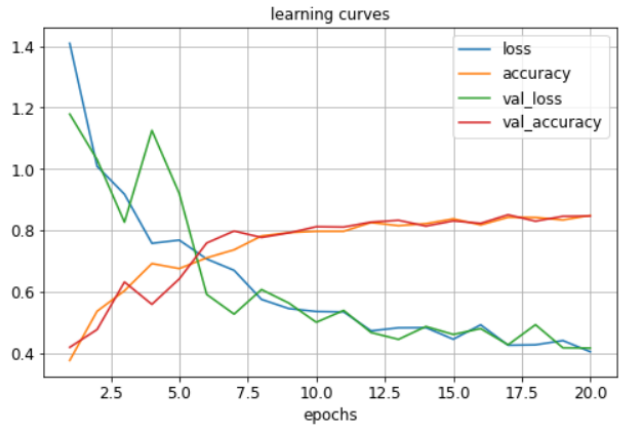


*Figure 1 : Optimized layer with **Sigmoid** activation function*

We can see that the Sigmoid activation function clearly reduces the noise of the loss and by consequence of the accuracy.

Analyse and compare the results between baseline and proposed models, discuss which is the best model for the task

The best model is the developed model with the three convolution layers, dropout and data augmentation. For the final layer the best activation function is the 'Sigmoid'.

Comment on the main difficulties encountered in running your experiments and/or to improve your models

The main difficulty as always is the problem of overfitting in the model. This was resolved by adding layers to the model like the dropout or the data augmentation.

sklearn.model_selection.RandomizedSearchCV wasn't adapted to the model as I couldn't fit the grid and have as an output the best parameters. That is why I chose GridSearchCV to find the optimal parameters for the current model.

Finally, it would be better for the GitHub repository to add the logs of the different models. Unfortunately, the files were too large, and it blocked the push of the repository. That is why the logs are in the Jupyter notebook but does not appear on GitHub.

Conclusion and next steps

Decide which model you would use for the task given your experimental results

Comment on eventual extra experiments or procedures you could perform to:

- refine your decision

After testing multiple models in my notebook, the best predictions I could find were on the models presented in this report and so we will choose the Sequential developed optimized model with a Sigmoid applied on the final layer to resolve the colorectal cancer histology problem.

- to improve your model's performance

To improve my model performance, I should use the GridSearchCV output parameters that I didn't have the time to run.

Another thing would be to improve the data augmentation with saturations, colour space transformations, Kernel filters, Random Erasing or Mixing images, I'm sure that will improve the model accuracy.

References

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

https://www.kaggle.com/code/hectorlopezhernandez/colorectalhistologymodel-cnn?scriptVersionId=111780293

https://medium.com/@ashraf.dasa/tensorflow-image-classification-of-colorectal-cancer-histology-92-5-accuracy-8b8b40ac775a