Lung cancer detection

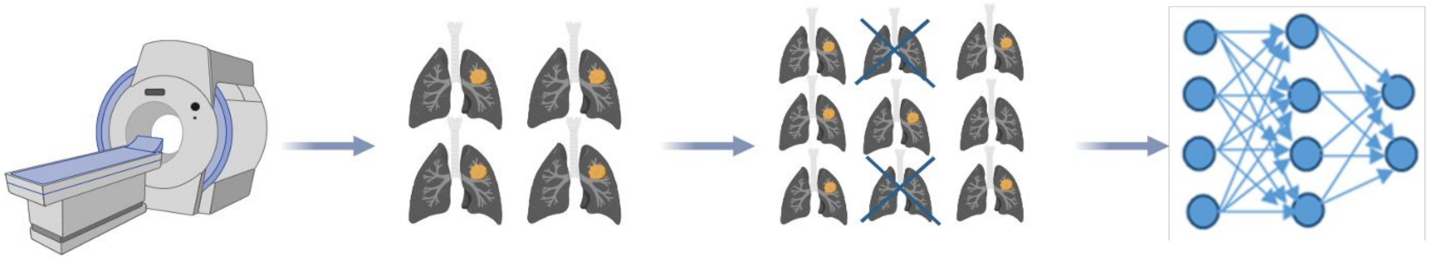
False Positive Reduction - Task 1

Neuroengineering
AY 2023-2024

Deside Guillaume, Khiabani Hasan, Milcent Romane, Sarmiento Clarysse Allyssa, Javadi Namin Mohammadreza

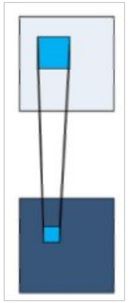
Lung Cancer and Artificial Intelligence

1. Early screening with low-dose computerized tomography LDCT
2. Large scale screening context
3. False positive reduction
4. Distinction malignant from benign nodules: 3D CNNs



Aim: **implement a 3D CNN for false positive reduction**

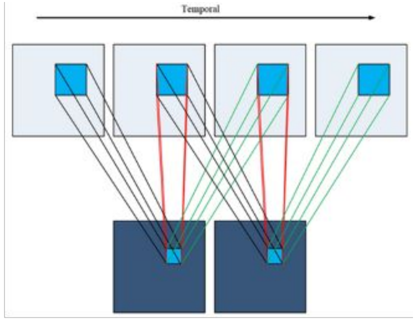
Multi-layer classification networks



2-D CNNs

VS

3-D CNNs



Pro of 3-D CNNs:

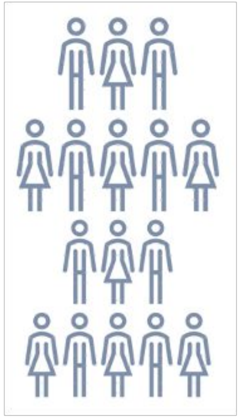
- higher discrimination capability

+

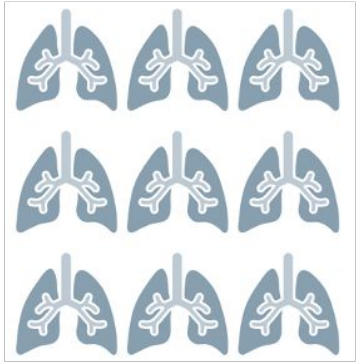
Large variations and hard mimics of pulmonary nodules → Integration of a set of 3-D CNNs with different sizes of receptive field. → Improved detection accuracy

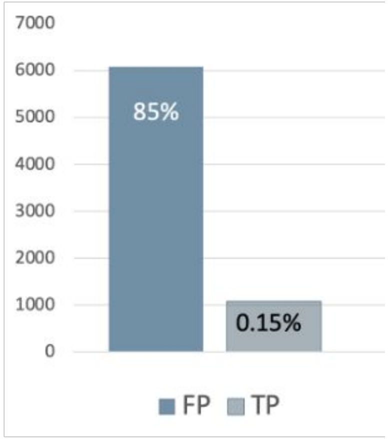
Aim of the work

81 patients



7161 region preposals





Plan of the work

1 Pre-processing/Data preparation*

*Some parts were already implemented by the educational team

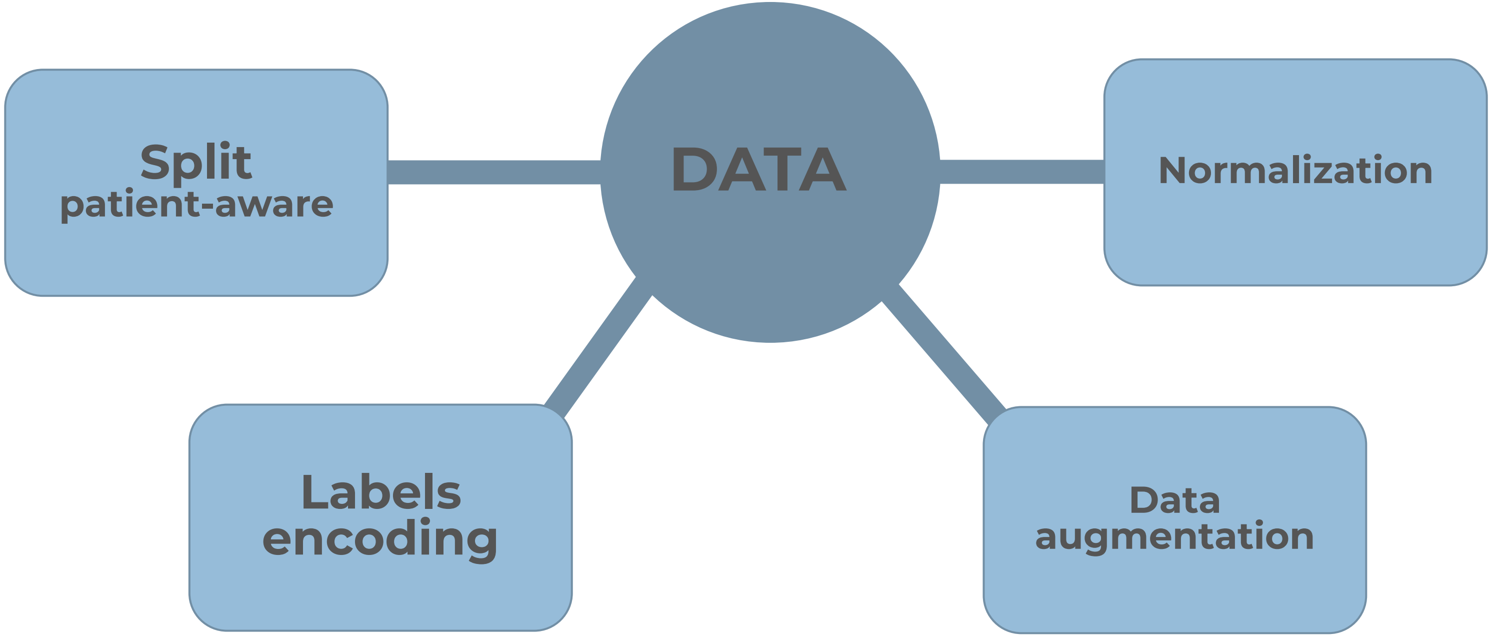
2 Implementation of original architecture

3 Test on singles architectures

4 Implementation of full architecture

5 Test on full architecture

Data preparation



DATA

Split
patient-aware

Normalization

Labels encoding

Data augmentation

The single architectures

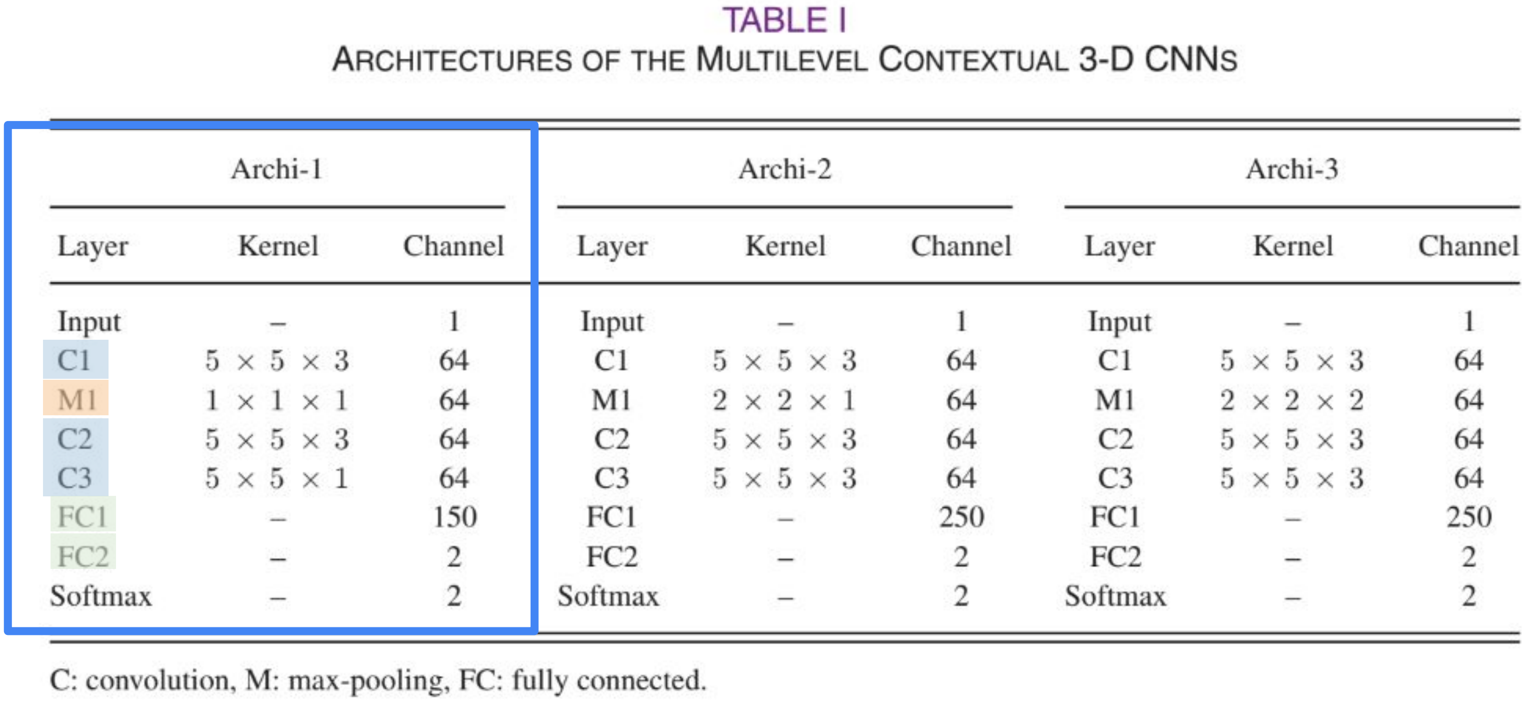


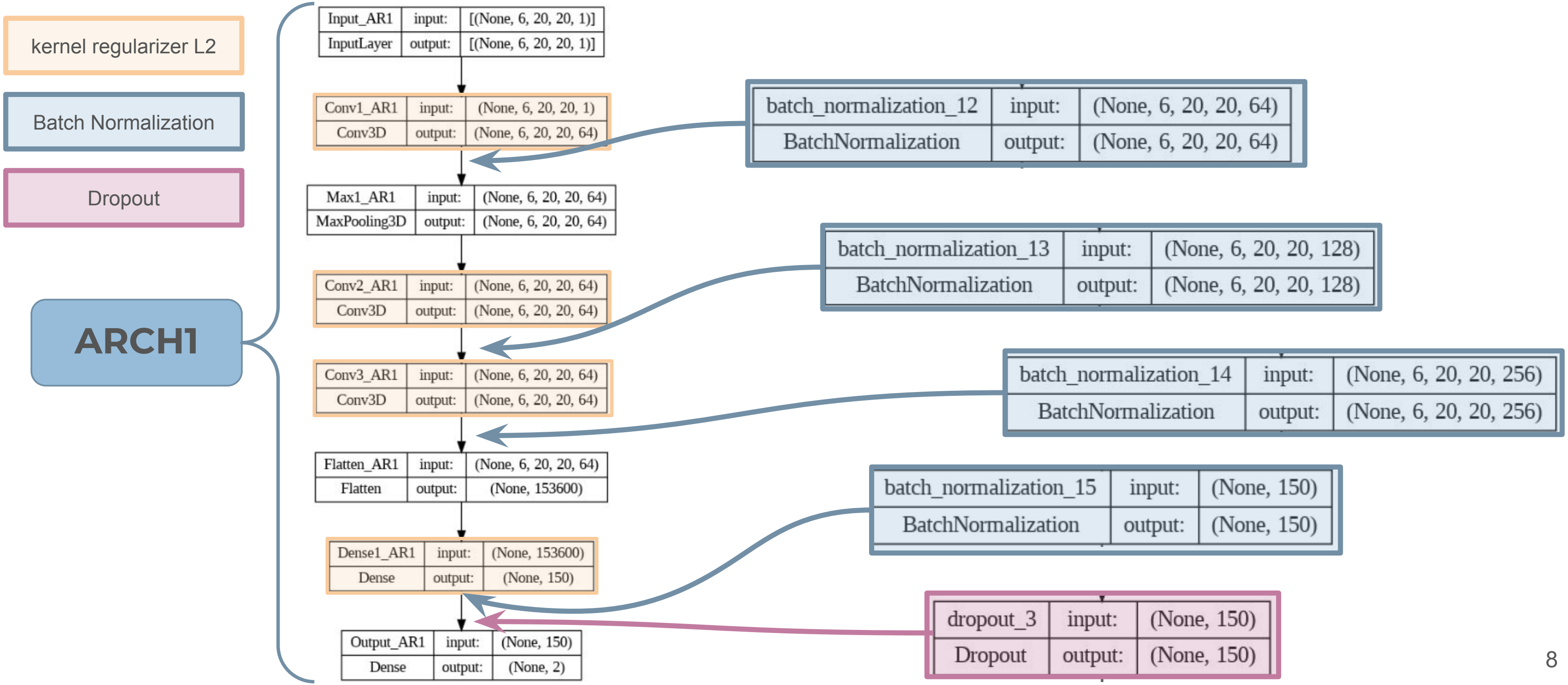
TABLE I
ARCHITECTURES OF THE MULTILEVEL CONTEXTUAL 3-D CNNS

Archi-1			Archi-2			Archi-3		
Layer	Kernel	Channel	Layer	Kernel	Channel	Layer	Kernel	Channel
Input	–	1	Input	–	1	Input	–	1
C1	5 × 5 × 3	64	C1	5 × 5 × 3	64	C1	5 × 5 × 3	64
M1	1 × 1 × 1	64	M1	2 × 2 × 1	64	M1	2 × 2 × 2	64
C2	5 × 5 × 3	64	C2	5 × 5 × 3	64	C2	5 × 5 × 3	64
C3	5 × 5 × 1	64	C3	5 × 5 × 3	64	C3	5 × 5 × 3	64
FC1	–	150	FC1	–	250	FC1	–	250
FC2	–	2	FC2	–	2	FC2	–	2
Softmax	–	2	Softmax	–	2	Softmax	–	2

C: convolution, M: max-pooling, FC: fully connected.

from Dou et al (2017)

The single architectures



kernel regularizer L2

Batch Normalization

Dropout

ARCH1

Input_AR1	input:	[(None, 6, 20, 20, 1)]
InputLayer	output:	[(None, 6, 20, 20, 1)]

Conv1_AR1	input:	(None, 6, 20, 20, 1)
Conv3D	output:	(None, 6, 20, 20, 64)

Max1_AR1	input:	(None, 6, 20, 20, 64)
MaxPooling3D	output:	(None, 6, 20, 20, 64)

Conv2_AR1	input:	(None, 6, 20, 20, 64)
Conv3D	output:	(None, 6, 20, 20, 64)

Conv3_AR1	input:	(None, 6, 20, 20, 64)
Conv3D	output:	(None, 6, 20, 20, 64)

Flatten_AR1	input:	(None, 6, 20, 20, 64)
Flatten	output:	(None, 153600)

Dense1_AR1	input:	(None, 153600)
Dense	output:	(None, 150)

Output_AR1	input:	(None, 150)
Dense	output:	(None, 2)

batch_normalization_12	input:	(None, 6, 20, 20, 64)
BatchNormalization	output:	(None, 6, 20, 20, 64)

batch_normalization_13	input:	(None, 6, 20, 20, 128)
BatchNormalization	output:	(None, 6, 20, 20, 128)

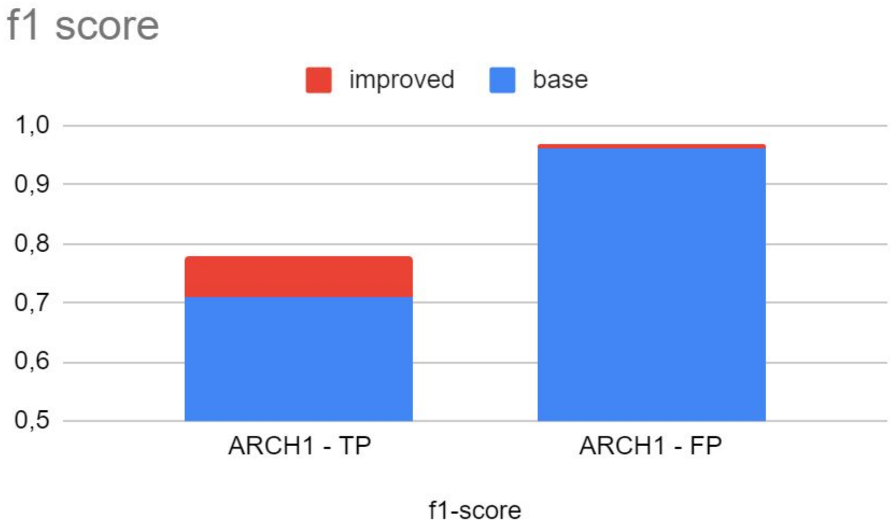
batch_normalization_14	input:	(None, 6, 20, 20, 256)
BatchNormalization	output:	(None, 6, 20, 20, 256)

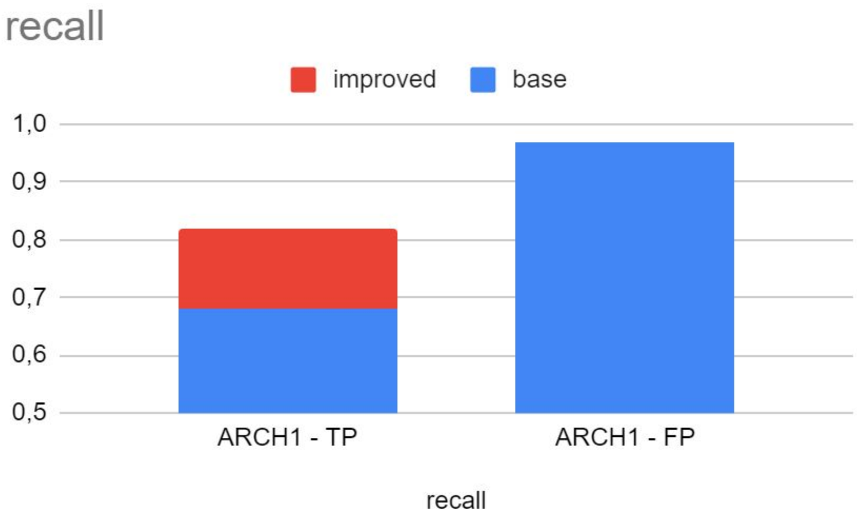
batch_normalization_15	input:	(None, 150)
BatchNormalization	output:	(None, 150)

dropout_3	input:	(None, 150)
Dropout	output:	(None, 150)

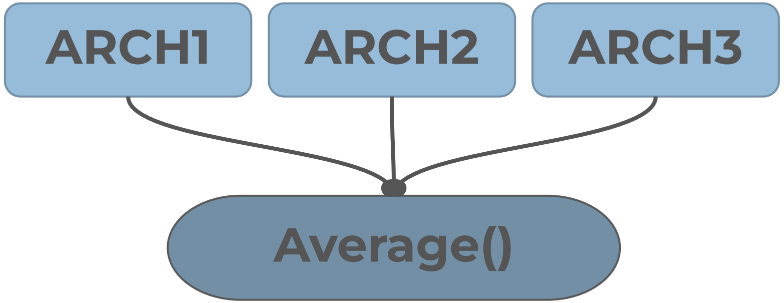
Results single architectures

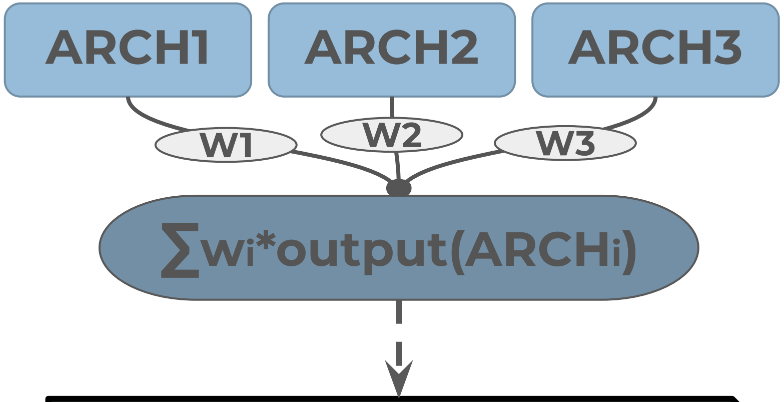
ARCH1





Merging strategies

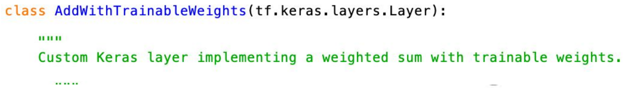




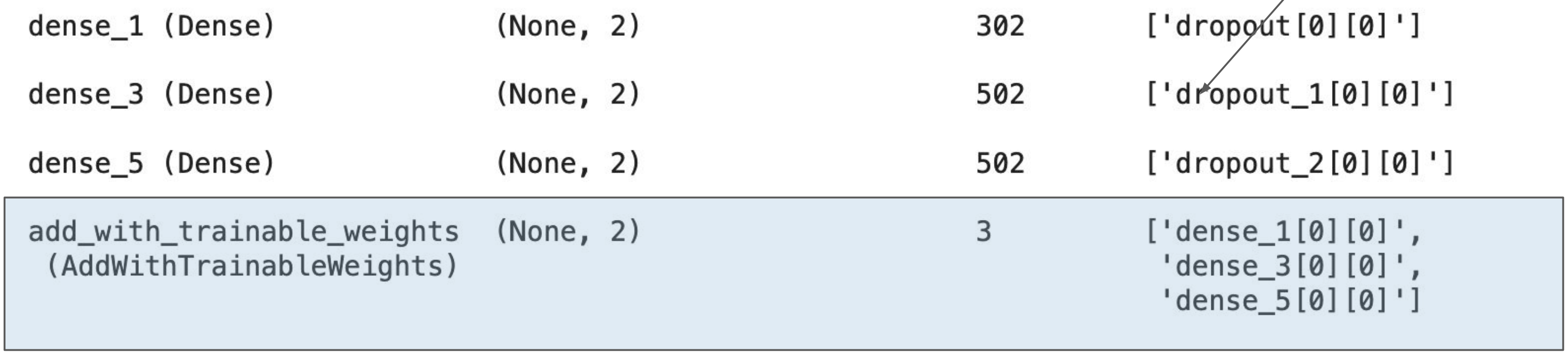
- ➔ Compute weights on individual models
- ➔ Define weights through trial and error
- ➔ Trainable weights on full model

Trainable weights

constraint



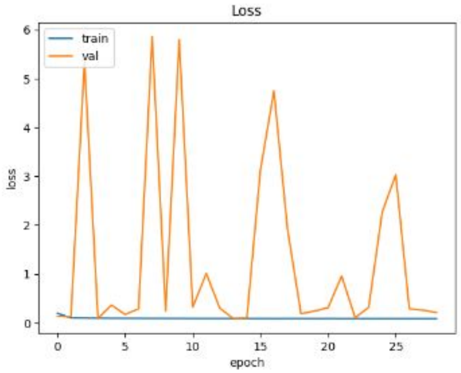
```
class AddWithTrainableWeights(tf.keras.layers.Layer):

    """
    Custom Keras layer implementing a weighted sum with trainable weights.
      """
```



```
dense_1 (Dense)                (None, 2)                       302        ['dropout[0][0]']

dense_3 (Dense)                (None, 2)                       502        ['dropout_1[0][0]']

dense_5 (Dense)                (None, 2)                       502        ['dropout_2[0][0]']

add_with_trainable_weights     (None, 2)                       3          ['dense_1[0][0]',
 (AddWithTrainableWeights)                                                 'dense_3[0][0]',
                                                                           'dense_5[0][0]']
```

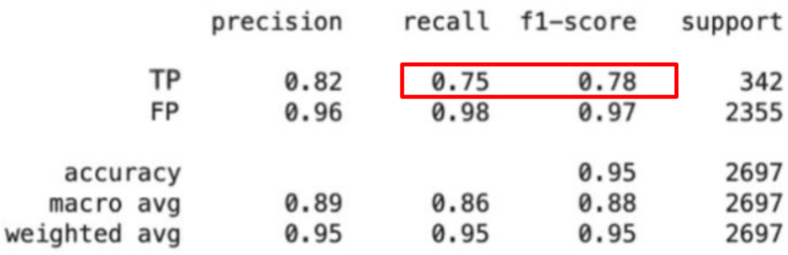
Problem : 5 hours by epoch

Problems encountered

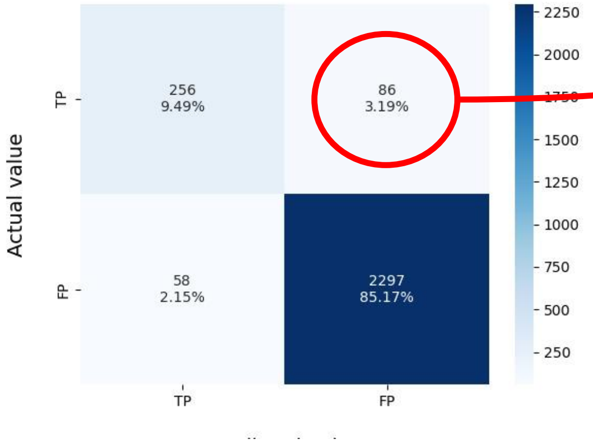
Problems	Solutions
Instability  	→ Adjust learning rate → Batch Normalization → Change the optimizer Adam AdamW Adadelta
Overfitting	→ Adjust data augmentation to compensate for the unbalanced dataset → Adjust L2 regularizer and dropout → Early Stopping

Results for the full architecture

Base

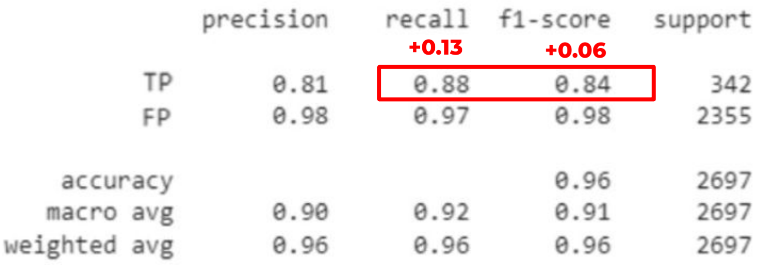


```
              precision    recall  f1-score   support

          TP       0.82      0.75      0.78       342
          FP       0.96      0.98      0.97      2355

    accuracy                           0.95      2697
   macro avg       0.89      0.86      0.88      2697
weighted avg       0.95      0.95      0.95      2697
```



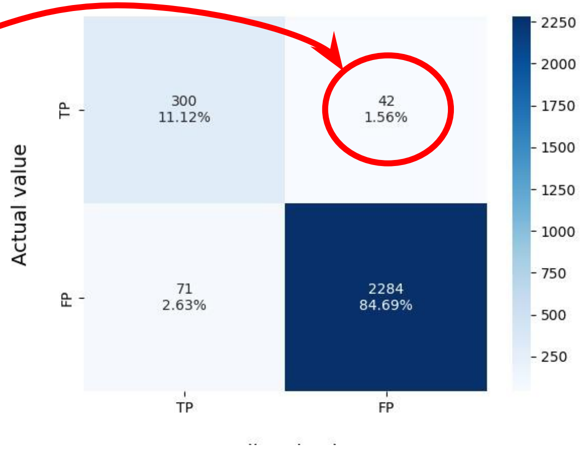
÷2

Improved

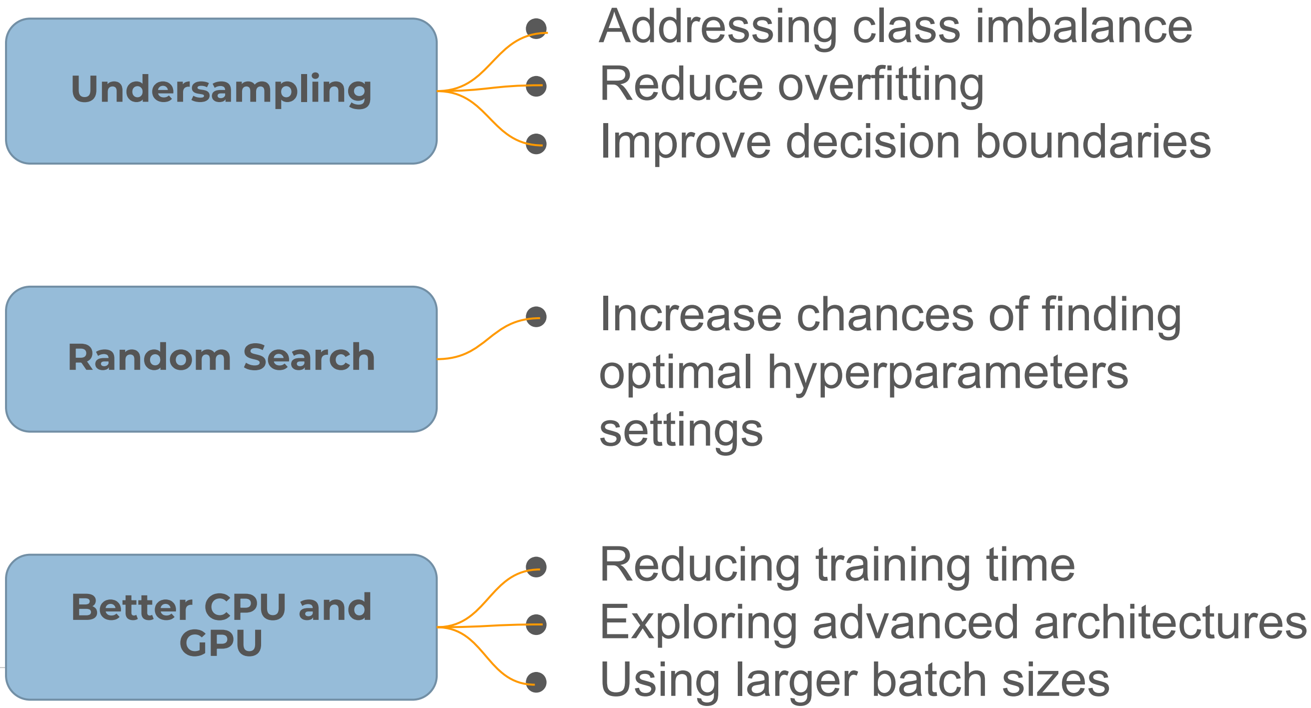


+0.13 +0.06

```
              precision    recall  f1-score   support

          TP       0.81      0.88      0.84       342
          FP       0.98      0.97      0.98      2355

    accuracy                           0.96      2697
   macro avg       0.90      0.92      0.91      2697
weighted avg       0.96      0.96      0.96      2697
```



Project improvement



Undersampling

- Addressing class imbalance
- Reduce overfitting
- Improve decision boundaries

Random Search

- Increase chances of finding optimal hyperparameters settings

Better CPU and GPU

- Reducing training time
- Exploring advanced architectures
- Using larger batch sizes

That's all Folks!

Conclusion

Reducing FP rate ✓