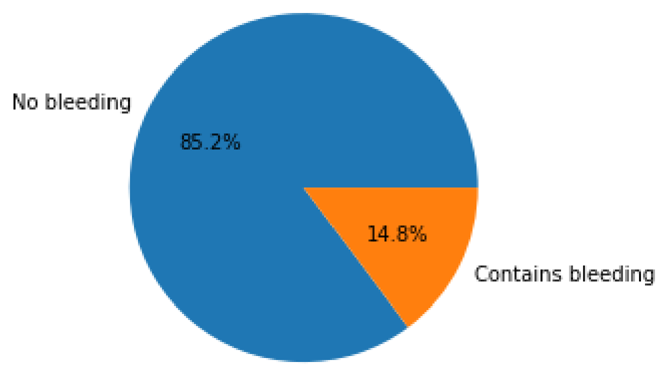
Brain Hemorrhage Identification

Melinte Tudor-Matei
507

Dataset

The dataset contains images of brain CT scans, each sample being a grayscale image of 224x224 pixels. The training set contains 17,000 labeled examples and the test set another 5149 examples. The training set is very unbalanced, because only 14.8% of the data represents cases of bleeding.

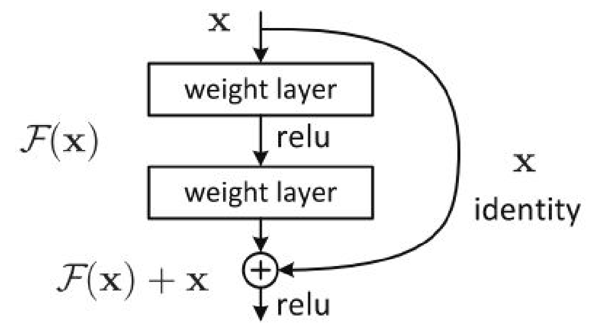


Dataset Distribution

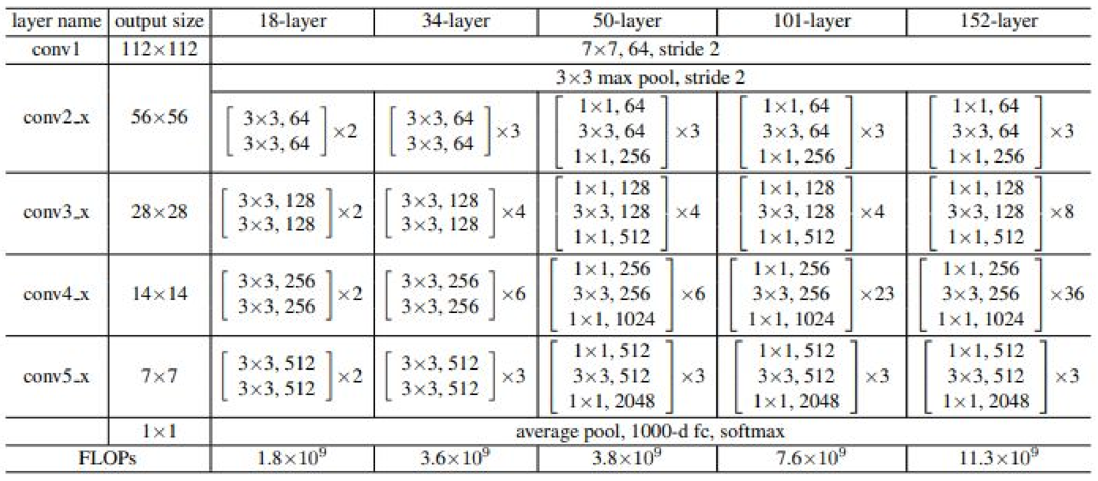
Proposed neural network architectures

ResNet

The core idea of ResNet is introducing a so-called “identity shortcut connection” that skips one or more layers, as shown in the figure below. Rather than fitting the latent weights to predict the final emotion at each layer, ResNet models fit a residual mapping to predict the delta needed to reach the final prediction from one layer to the next. The identity mapping enables the model to bypass a typical CNN weight layer if the current layer is not necessary. This further helps the model to avoid overfitting to the training set. I used the ResNet 101 network pretrained on the ImageNet. I replaced the original prediction layers (top) with a Dropout layer with a rate of 0.5, followed by a Sigmoid layer for prediction.



ResNet residual block diagram with identity mapping.

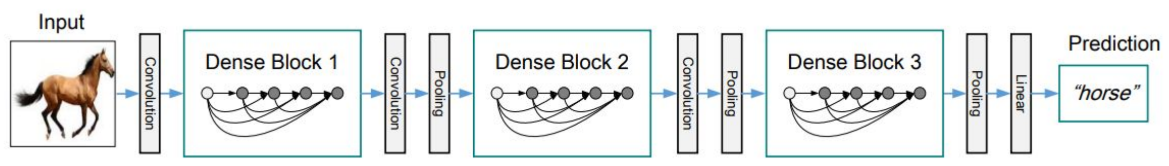


layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

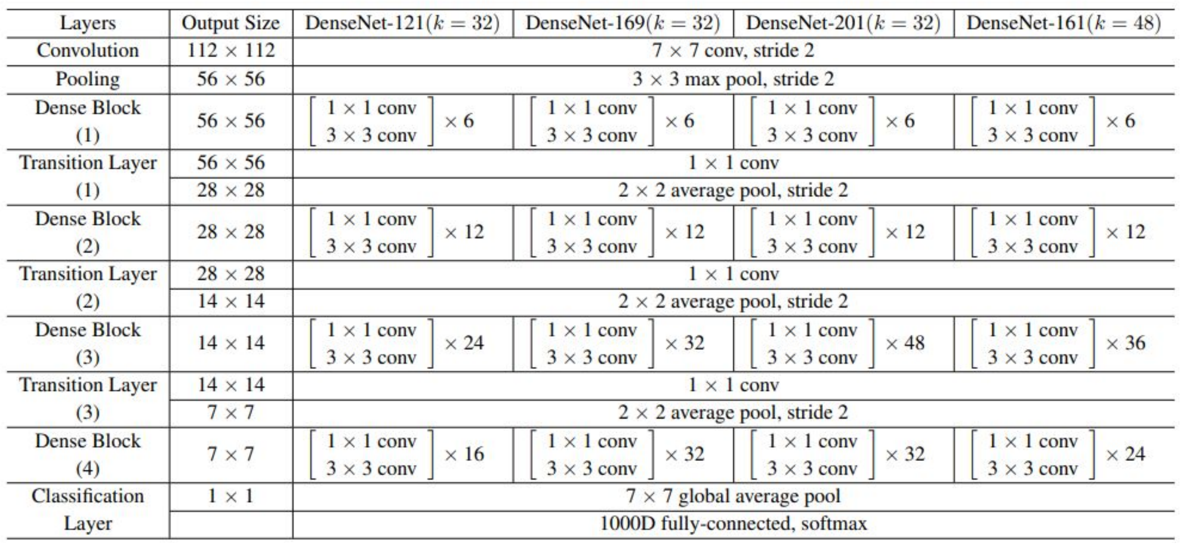
ResNet architectures for ImageNet

DenseNet

ResNet architecture has a fundamental building block (Identity) where you merge (additive) a previous layer into a future layer. Reasoning here is by adding additive merges we are forcing the network to learn residuals (errors i.e. diff between some previous layer and current one). In contrast, DenseNet paper proposes concatenating outputs from the previous layers instead of using the summation. I used the DenseNet 121 network pretrained on the ImageNet. I replaced the original prediction layers (top) with a Dropout layer with a rate of 0.5, followed by a Sigmoid layer for prediction.



A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature map sizes via convolution and pooling



Layers	Output Size	DenseNet-121($k = 32$)	DenseNet-169($k = 32$)	DenseNet-201($k = 32$)	DenseNet-161($k = 48$)
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 36$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

DenseNet architectures for ImageNet

Training details

On all experiments I tried to minimize the binary cross-entropy using Adam. All models were trained for 7 epochs at a learning rate of $5*10^{-4}$.

Improvements for an unbalanced dataset

Because the training set is unbalanced I tried different techniques for helping the model train better.

Set an initial bias for the final layer

By initializing the final layer's weights, the network will predict the class with lower number of samples with a higher probability. E.g. if you are regressing some values that have a mean of 50 then initialize the final bias to 50. If you have an imbalanced dataset of a ratio 1:10 of positives:negatives, set the bias on your logits such that your network predicts probability of 0.1 at initialization. Setting these correctly will speed up convergence and eliminate "hockey stick" loss curves where in the first few iterations your network is basically just learning the bias.

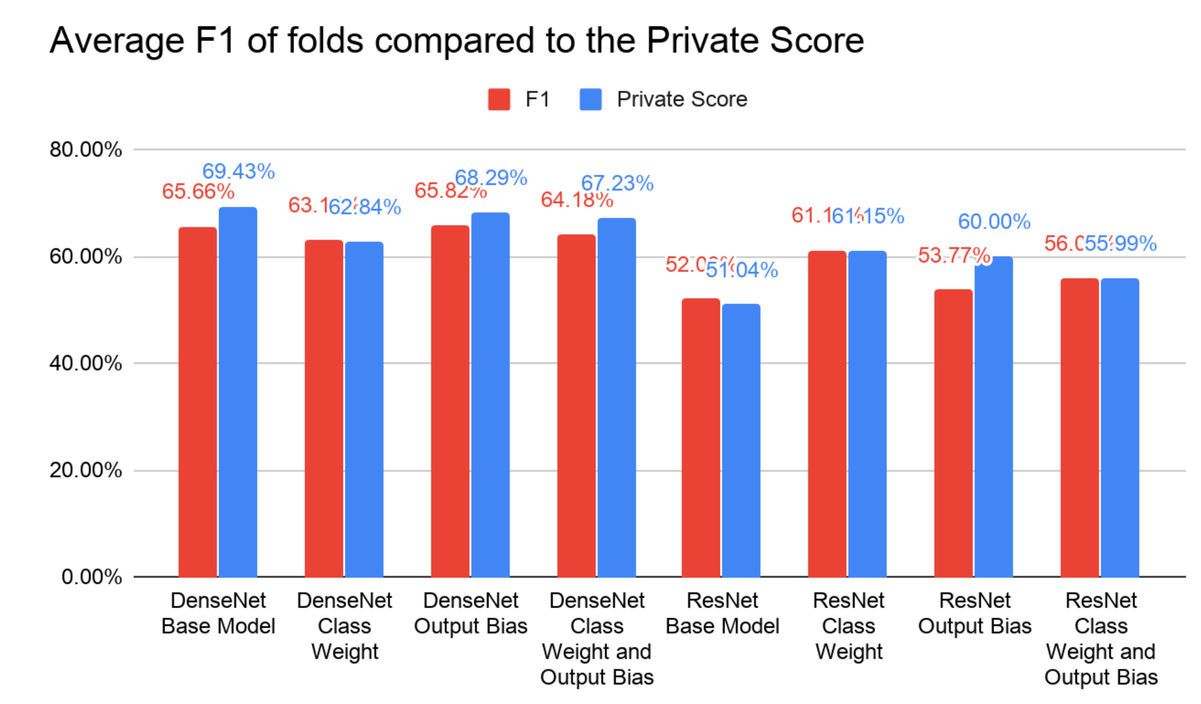
Class weights

Because we don't have many positive samples to work with, so you would want to have the classifier heavily weight the few examples that are available. You can do this by passing Keras weights for each class through a parameter. These will cause the model to "pay more attention" to examples from an under-represented class.

Experiments

For both network architectures I tested the base model, the model trained with an initial bias for the final layer, the model trained with class weights and a model with both the initial bias and the class weights.
For the DenseNet models, the changes for the unbalanced dataset did not improve the base model, but achieved similar performances. The only exception is for the class weight model, which had a big drop in accuracy.
For the ResNet models, the base model had the lowest F1 score, but the other models had a much better performance, class weight model and the output bias model achieved similar performances.
Overall the DenseNet models achieved much better scores then the ResNet models.

Model	Accuracy	Precision	Recall	F1	Private Score	Public Score
DenseNet Base Model	91.64%	85.47%	53.30%	65.66%	69.43%	68.09%
DenseNet Class Weight	87.09%	54.65%	74.78%	63.15%	62.84%	66.67%
DenseNet Output Bias	89.81%	71.07%	61.30%	65.82%	68.29%	72.95%
DenseNet Class Weight and Output Bias	86.69%	55.20%	76.65%	64.18%	67.23%	68.35%
ResNet Base Model	89.48%	87.26%	37.11%	52.08%	51.04%	55.38%
ResNet Class Weight	82.25%	59.78%	62.65%	61.18%	61.15%	67.17%
ResNet Output Bias	88.63%	75.84%	41.65%	53.77%	60.00%	65.49%
ResNet Class Weight and Output Bias	73.21%	53.91%	58.27%	56.01%	55.99%	58.48%

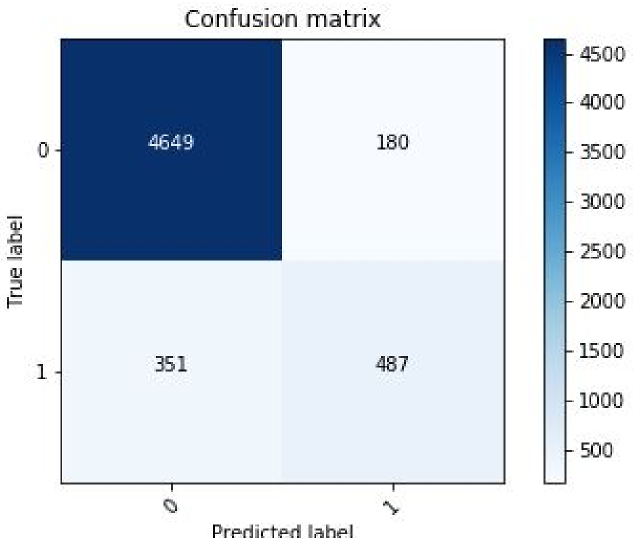


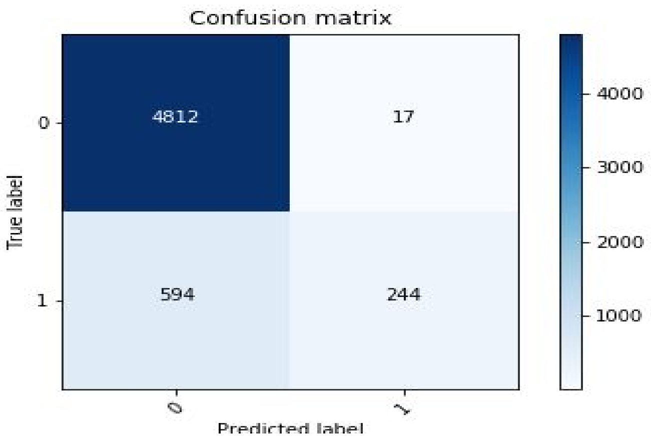
The F1 scores from the folds is similar with score obtained on the test dataset

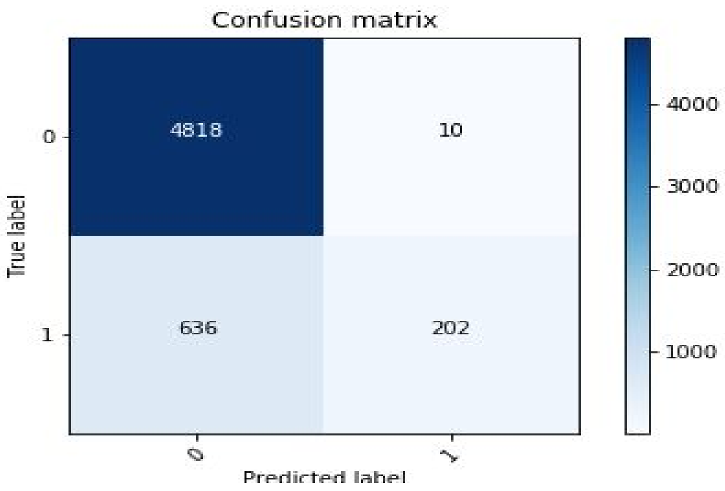
Confusion Matrices

ResNet

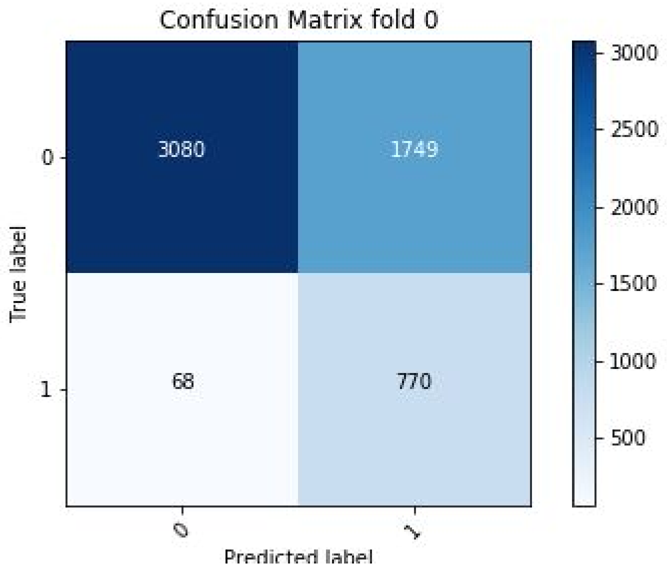
Base Model

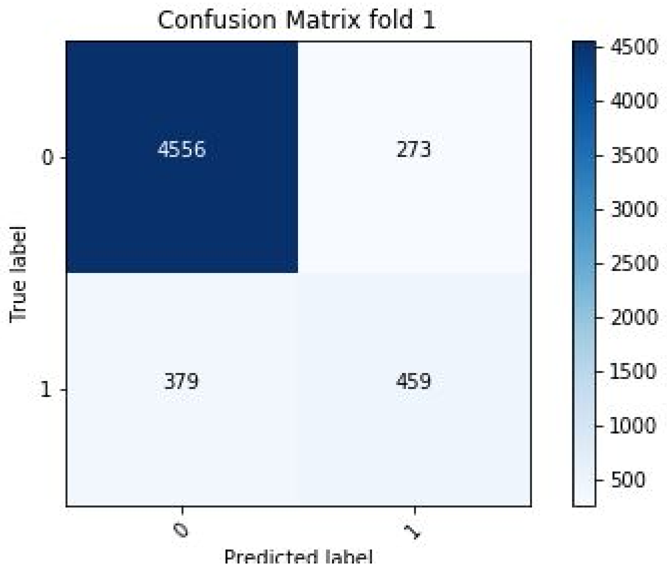


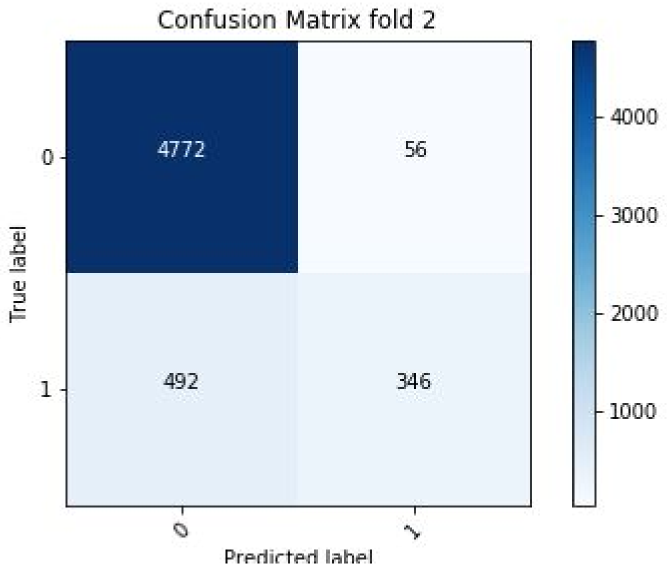




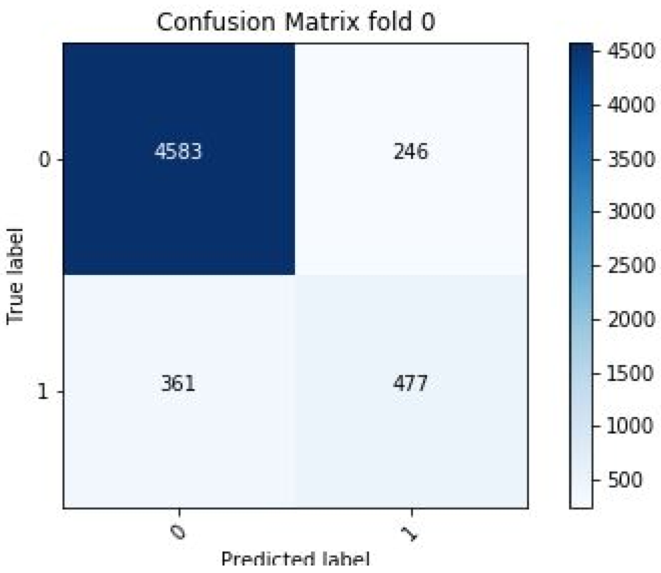
Class Weight



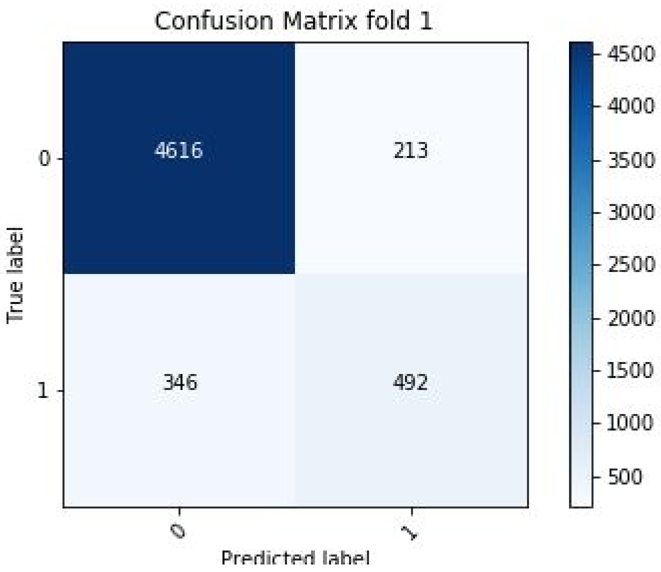




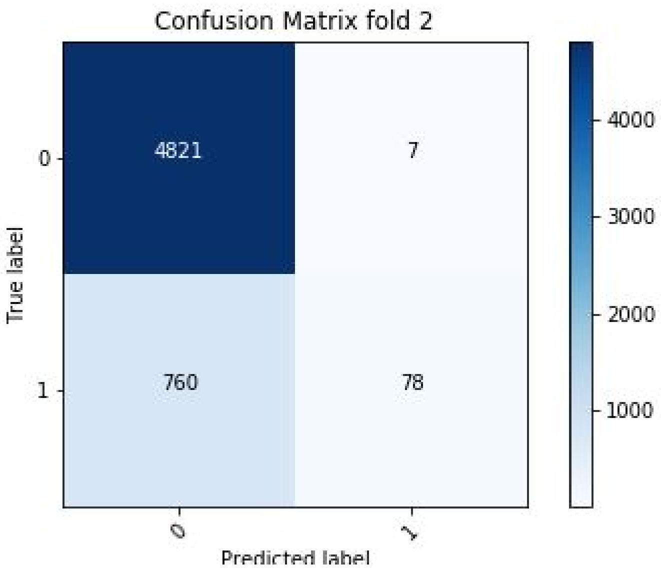
Output Bias



Confusion Matrix fold 0
True label
Predicted label
0
1
4583
246
361
477
4500
4000
3500
3000
2500
2000
1500
1000
500

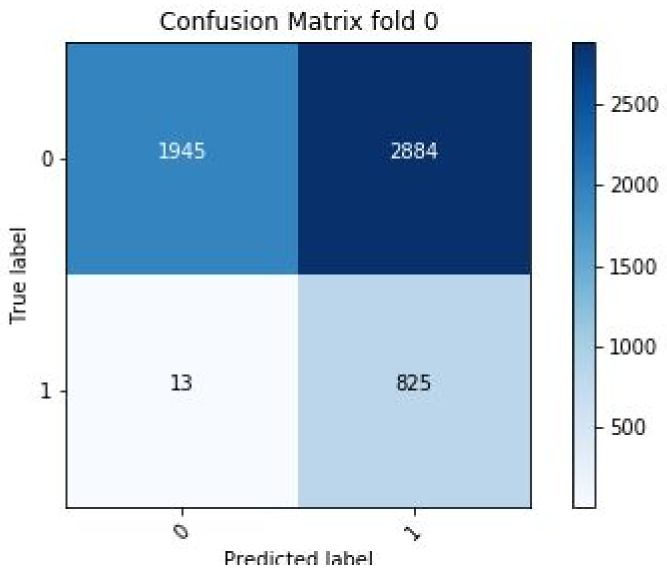


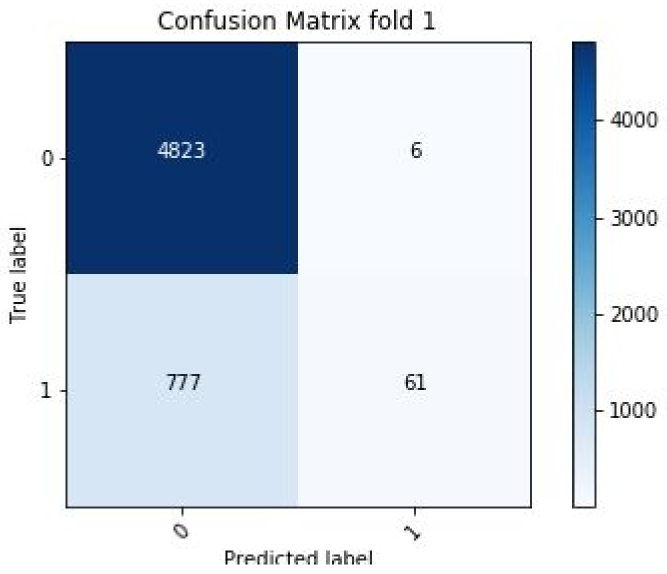
Confusion Matrix fold 1
True label
Predicted label
0
1
4616
213
346
492
4500
4000
3500
3000
2500
2000
1500
1000
500

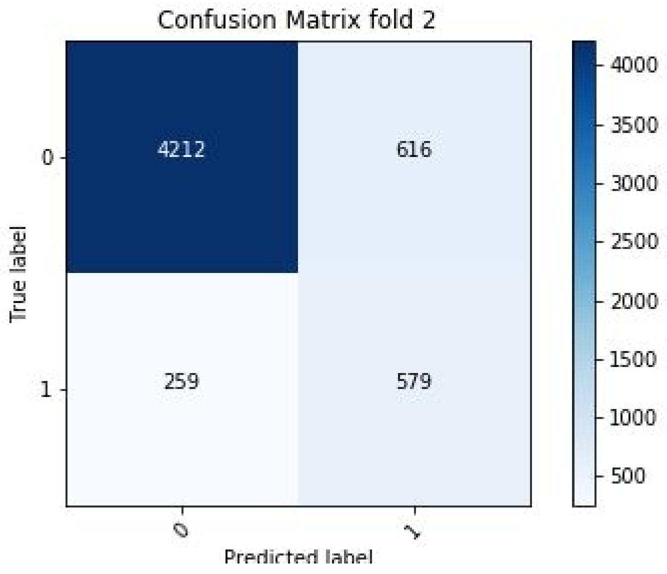


Confusion Matrix fold 2
True label
Predicted label
0
1
4821
7
760
78
4000
3000
2000
1000

Output Bias and Class Weight

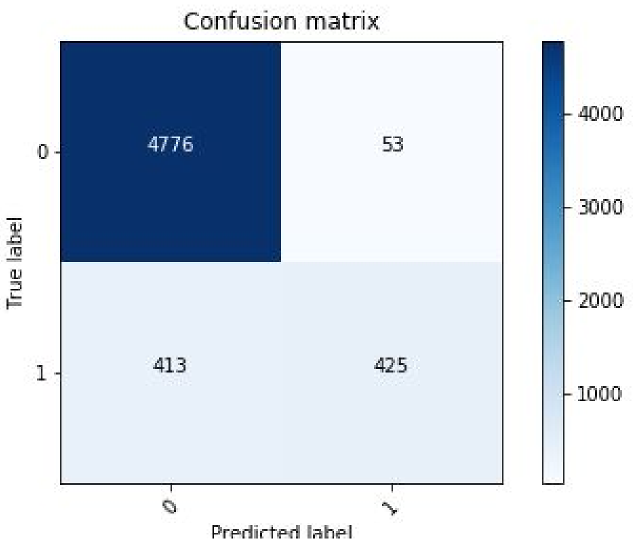


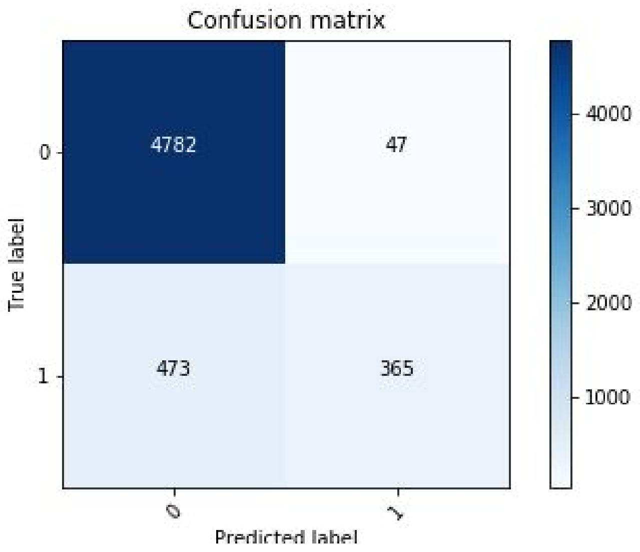


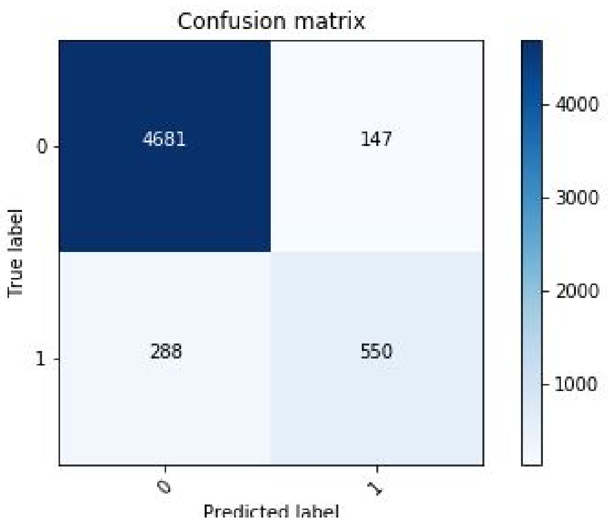


DenseNet

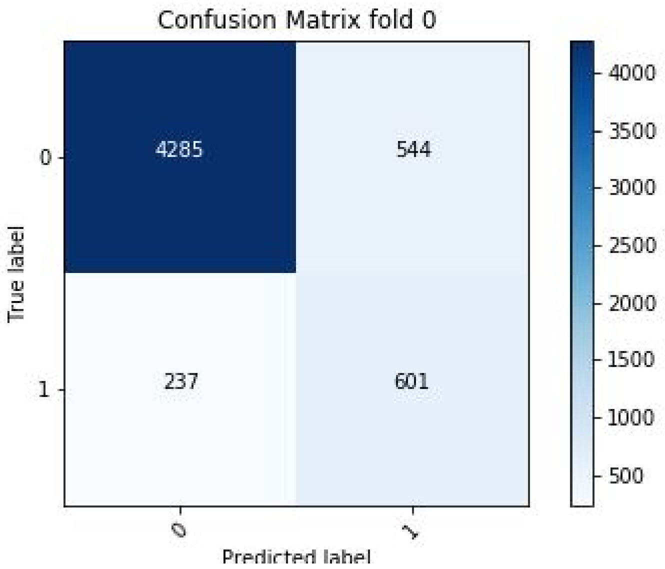
Base Model

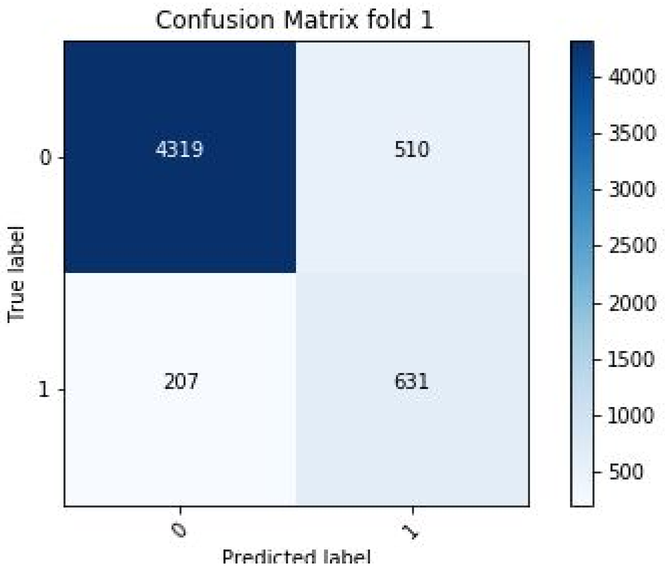


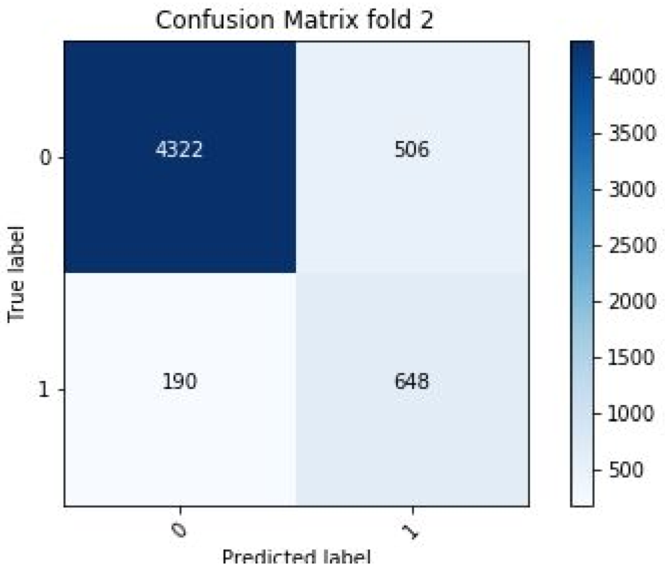




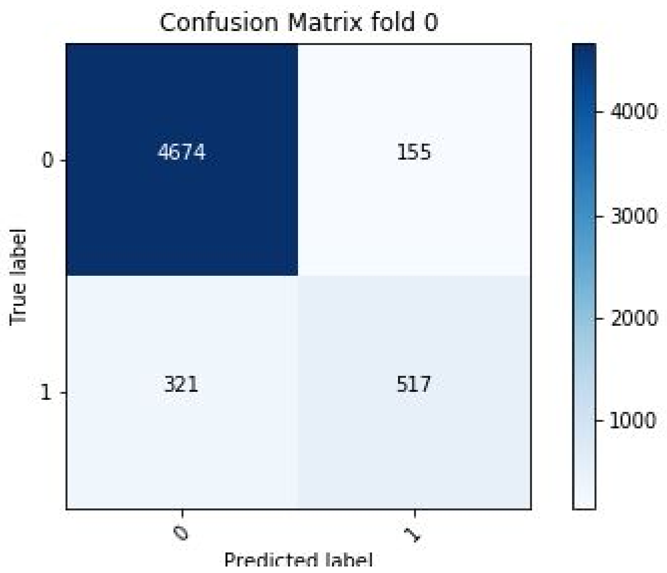
Class Weight



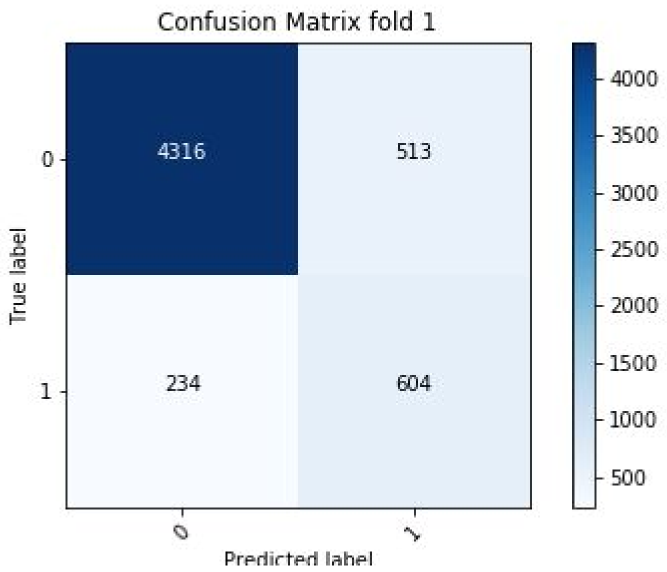




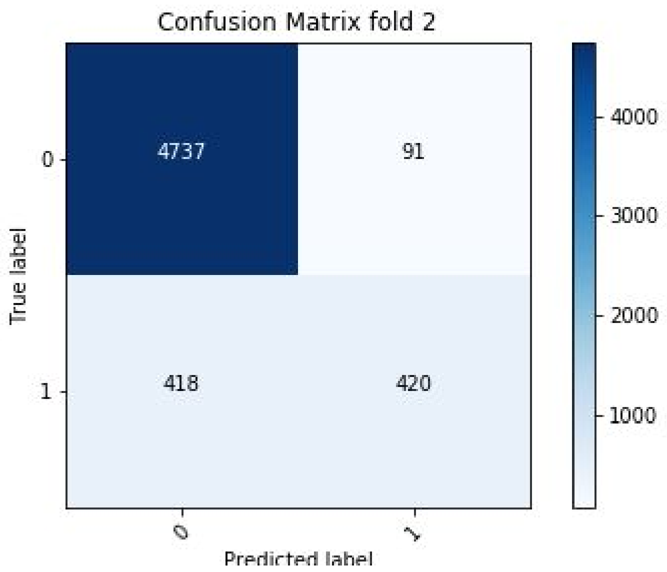
Output Bias



Confusion Matrix fold 0
True label
Predicted label
0
1
0
1
4674
155
321
517
4000
3000
2000
1000

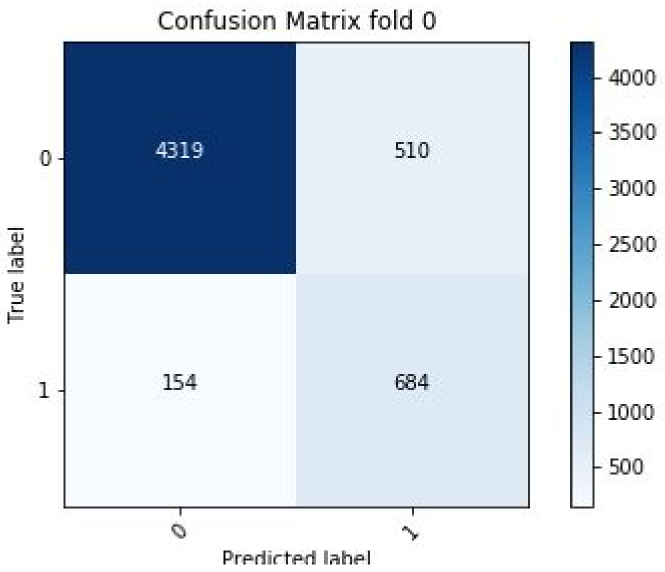


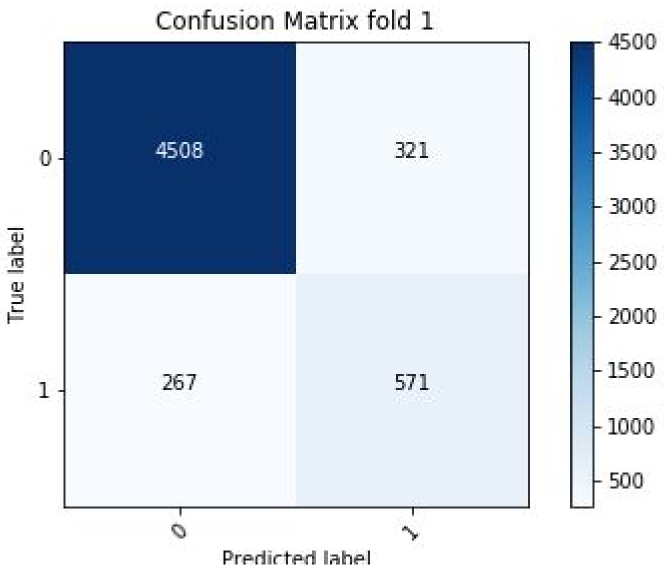
Confusion Matrix fold 1
True label
Predicted label
0
1
0
1
4316
513
234
604
4000
3500
3000
2500
2000
1500
1000
500

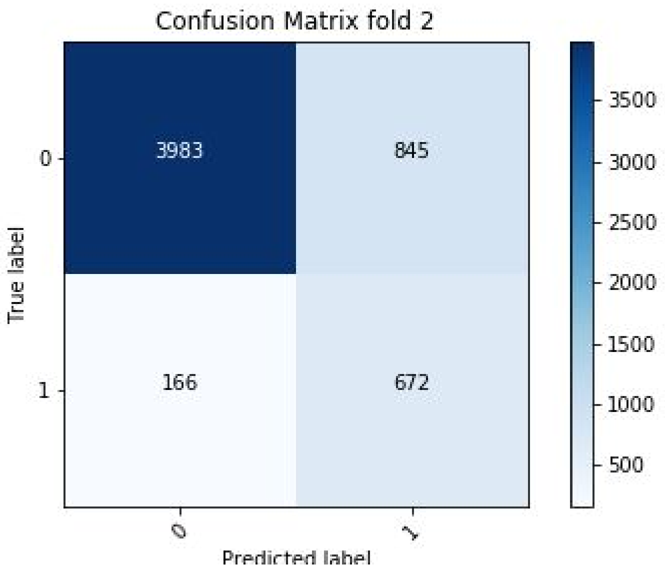


Confusion Matrix fold 2
True label
Predicted label
0
1
0
1
4737
91
418
420
4000
3000
2000
1000

Output Bias and Class Weight


Confusion Matrix fold 0
True label
Predicted label


Confusion Matrix fold 1
True label
Predicted label


Confusion Matrix fold 2
True label
Predicted label