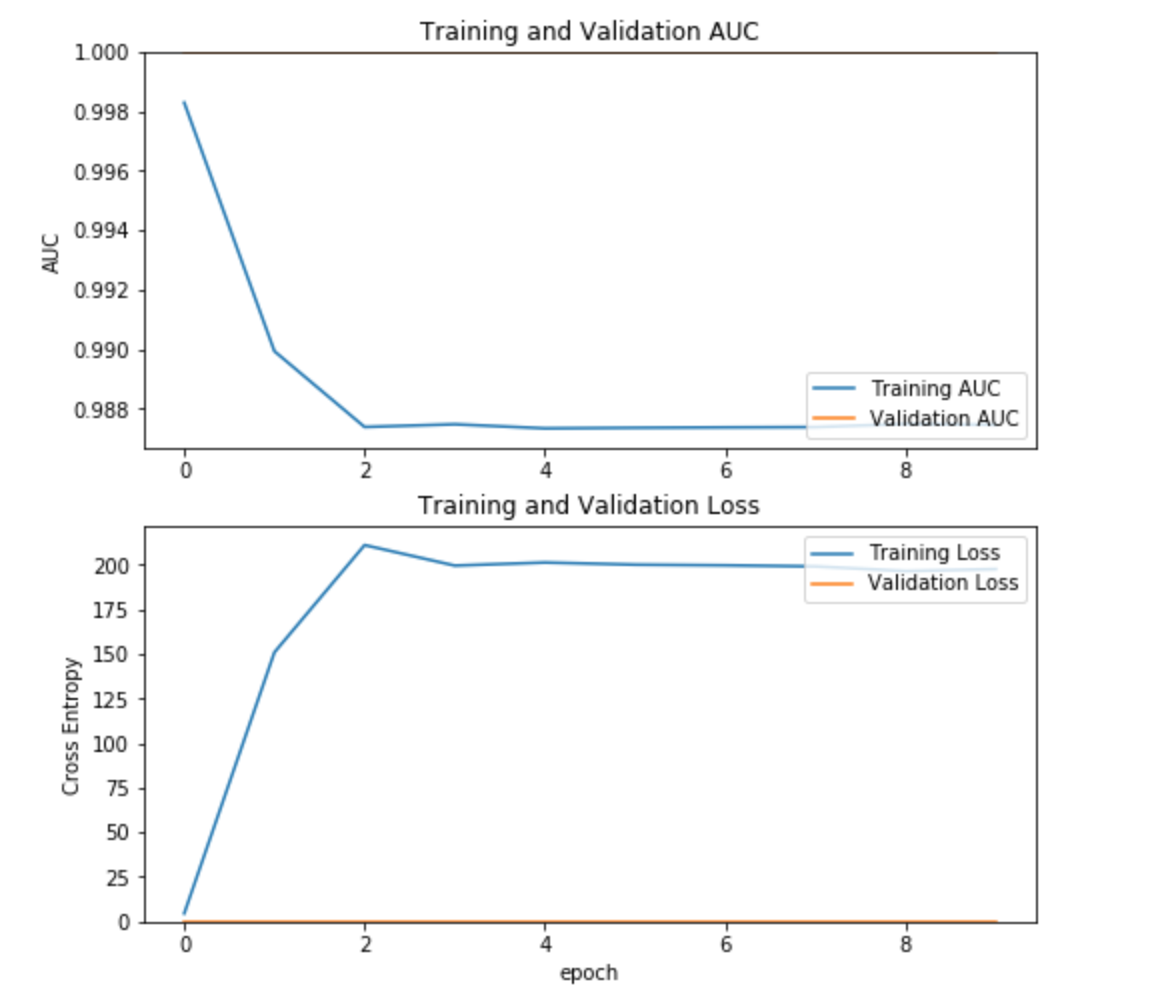
10 epochs, batch\_size =200, loading dataset without shuffle. 0.3 split.

No data augmentation.

Imbalance class.



Using oda-compute-0-11

Have to run this:

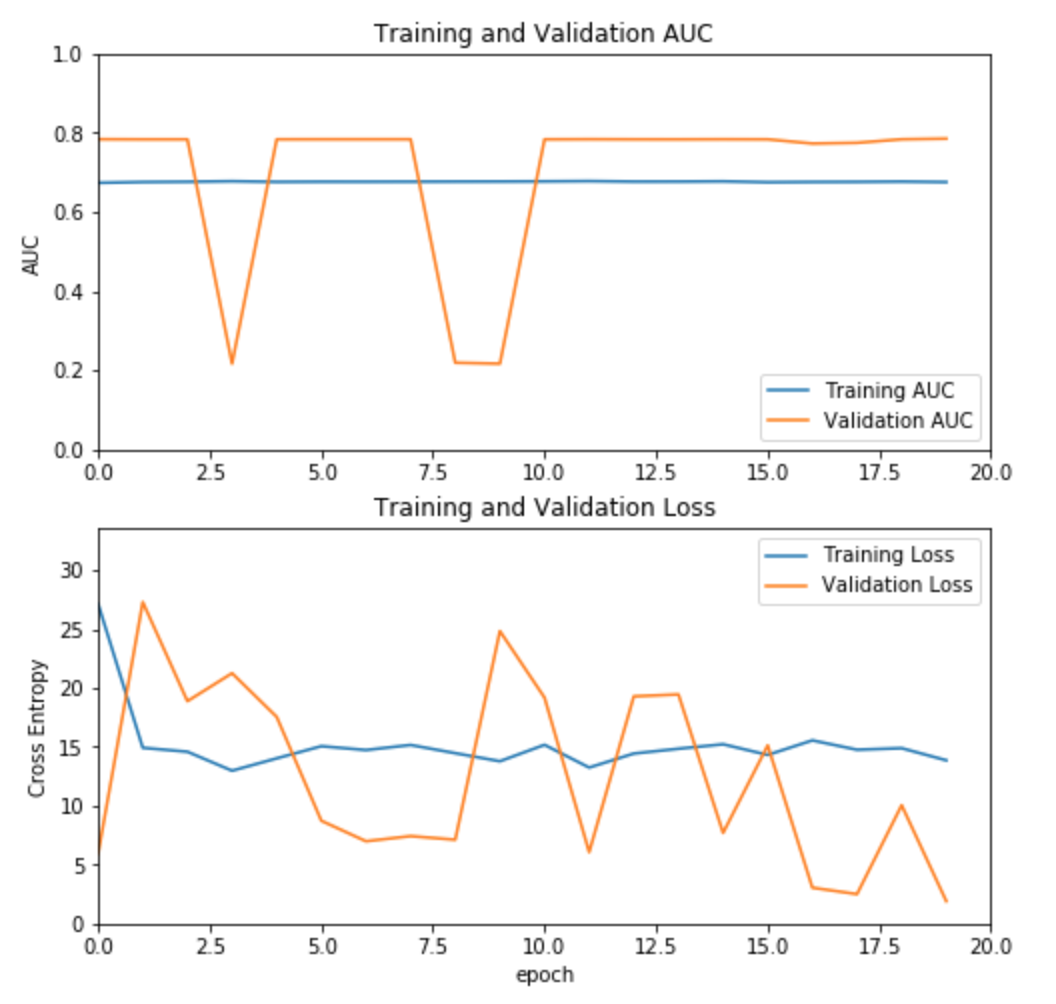
export LD\_LIBRARY\_PATH=/lib:/usr/lib:/usr/local/lib

before activate python 3.7

20epochs, batch\_size =200, loading dataset with shuffle. 0.3 split.

No data augmentation.

Imbalance class.



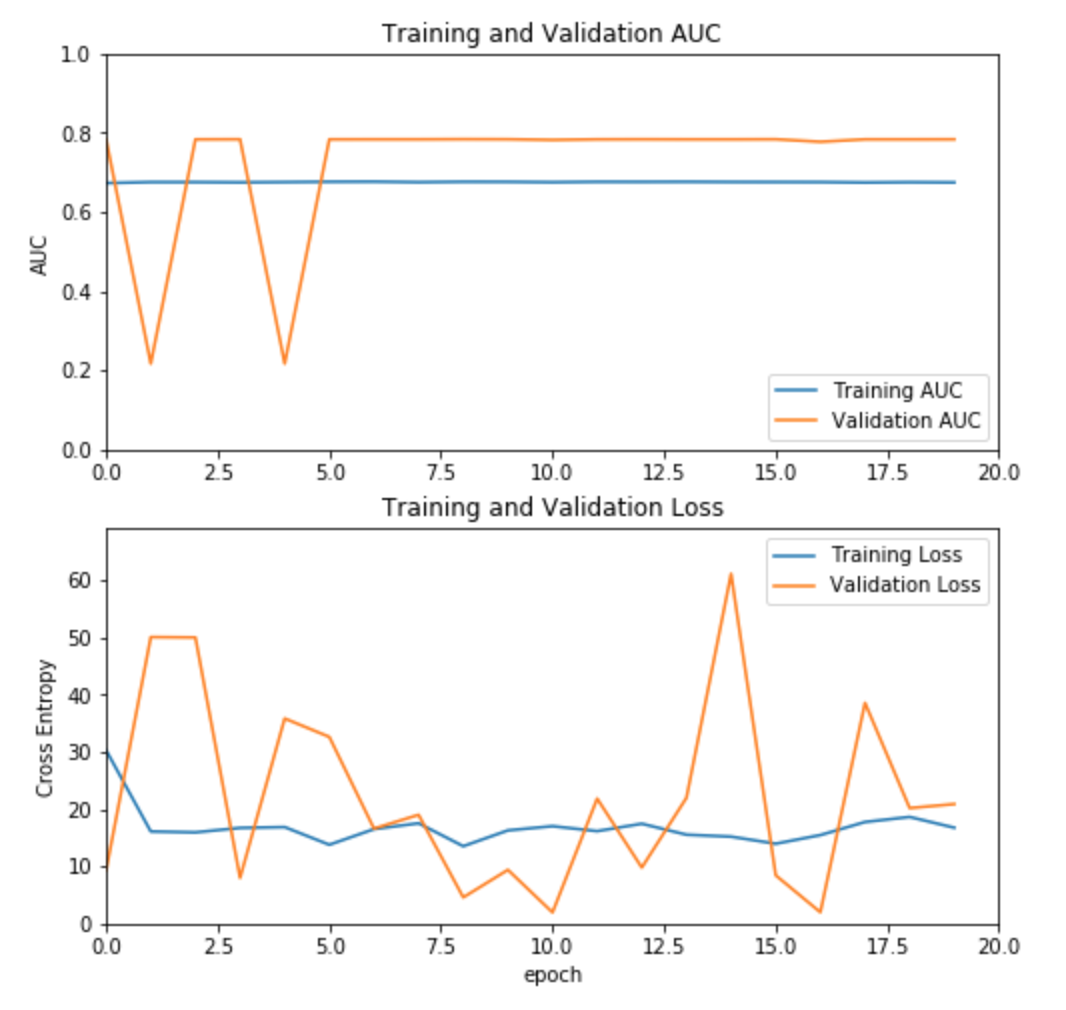
20 epochs, batch\_size =300, loading dataset with shuffle. 0.3 split.

No data augmentation.

Imbalance class.

* Uncomment some code(mode.evalutate) to try to avoid OOM error.

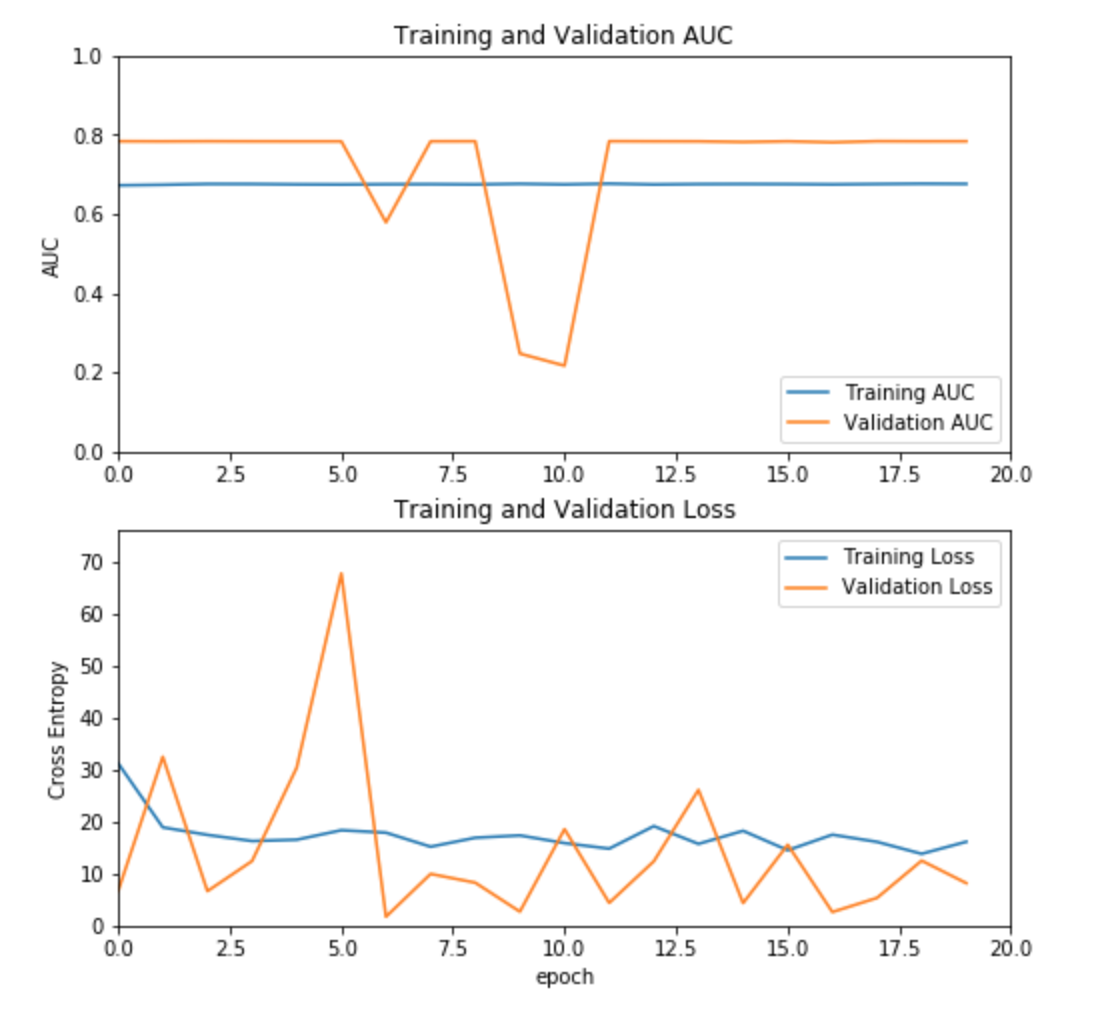
Seems like running fine



20 epochs, batch\_size =400, loading dataset with shuffle. 0.3 split.

No data augmentation.

Imbalance class.



50 epochs, batch\_size =500, l(OOM)oading dataset with shuffle. 0.3 split.

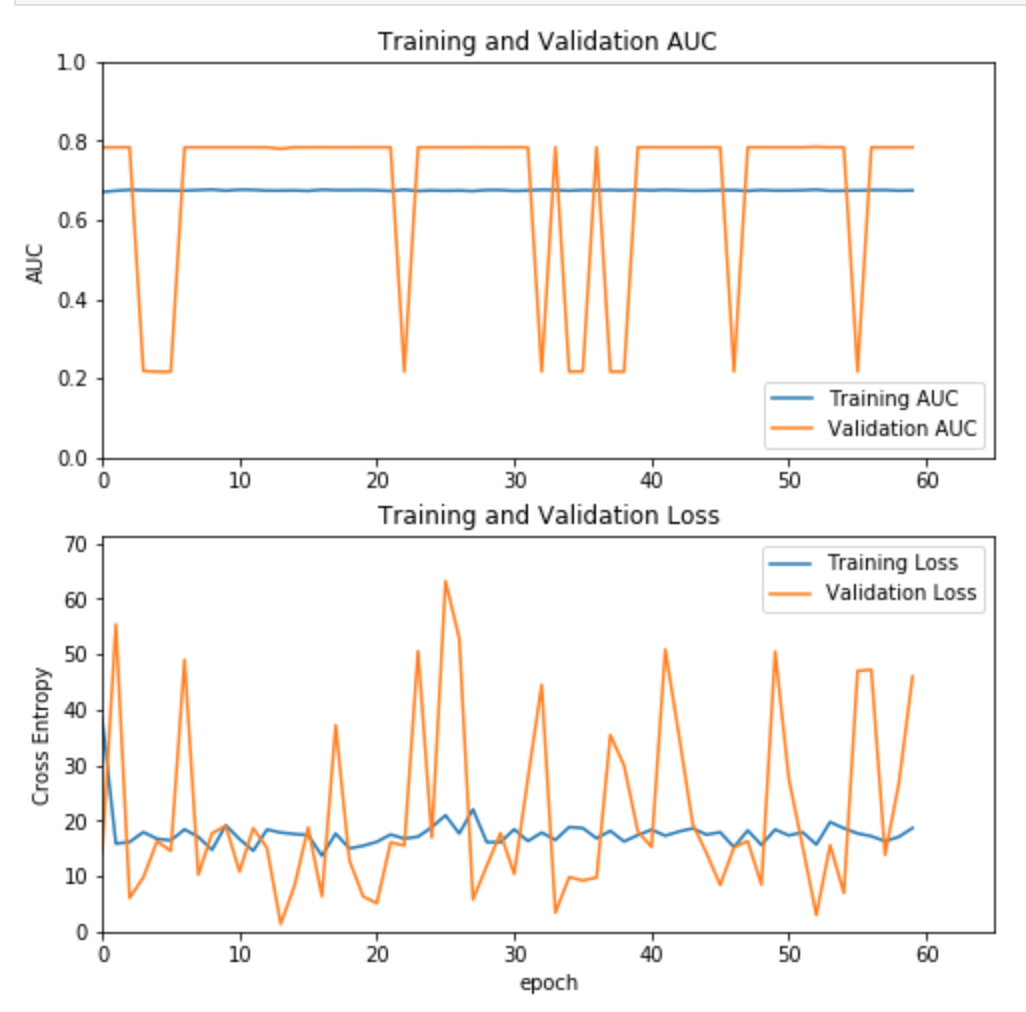
No data augmentation.

Imbalance class.

🡺 OOM

#-------------------------------

Batch size stay at 400. Epoch = 60.



Task: data augmentation for imbalance class.

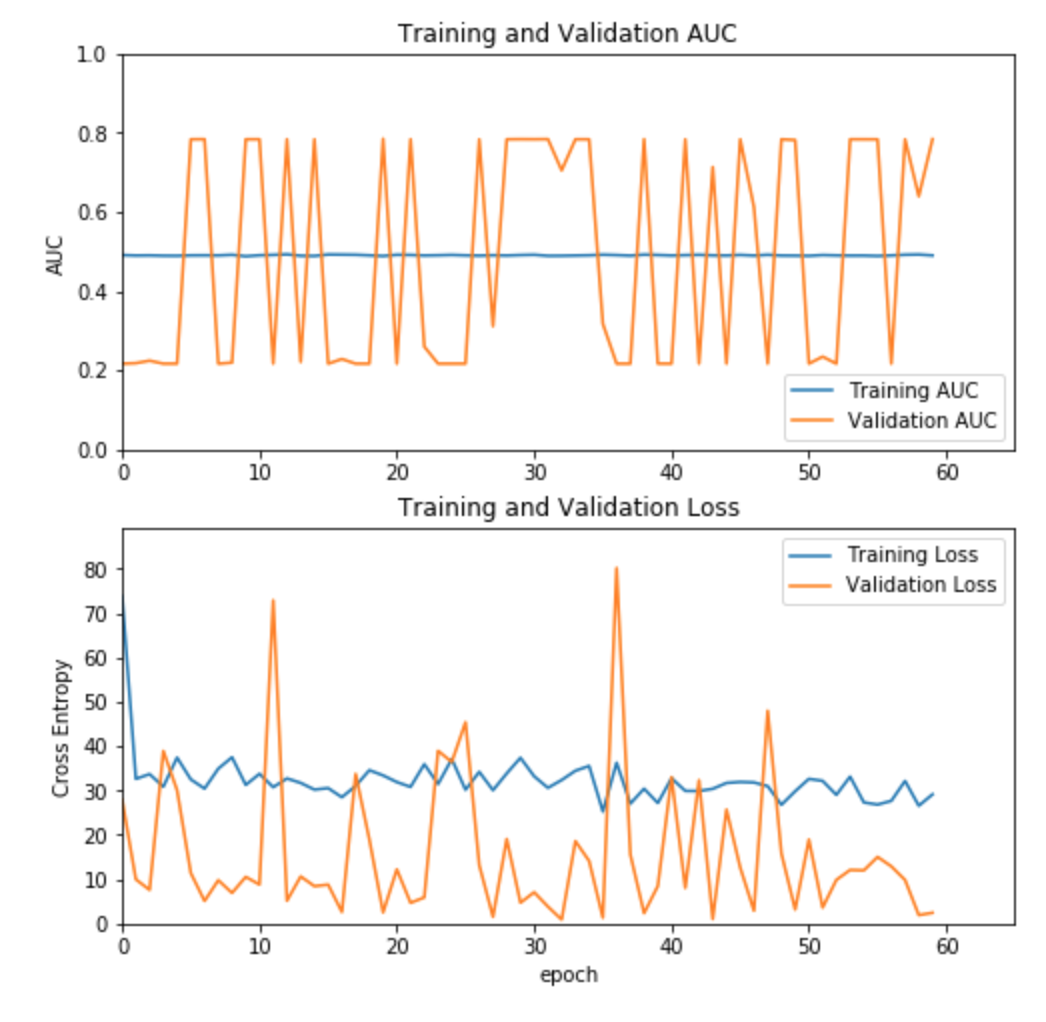
**class\_weight**: Optional dictionary mapping class indices (integers) to a weight (float) value, used for weighting the loss function (during training only). This can be useful to tell the model to "pay more attention" to samples from an under-represented class.

Batch size stay at 400. Epoch = 60.

Change the scripts to

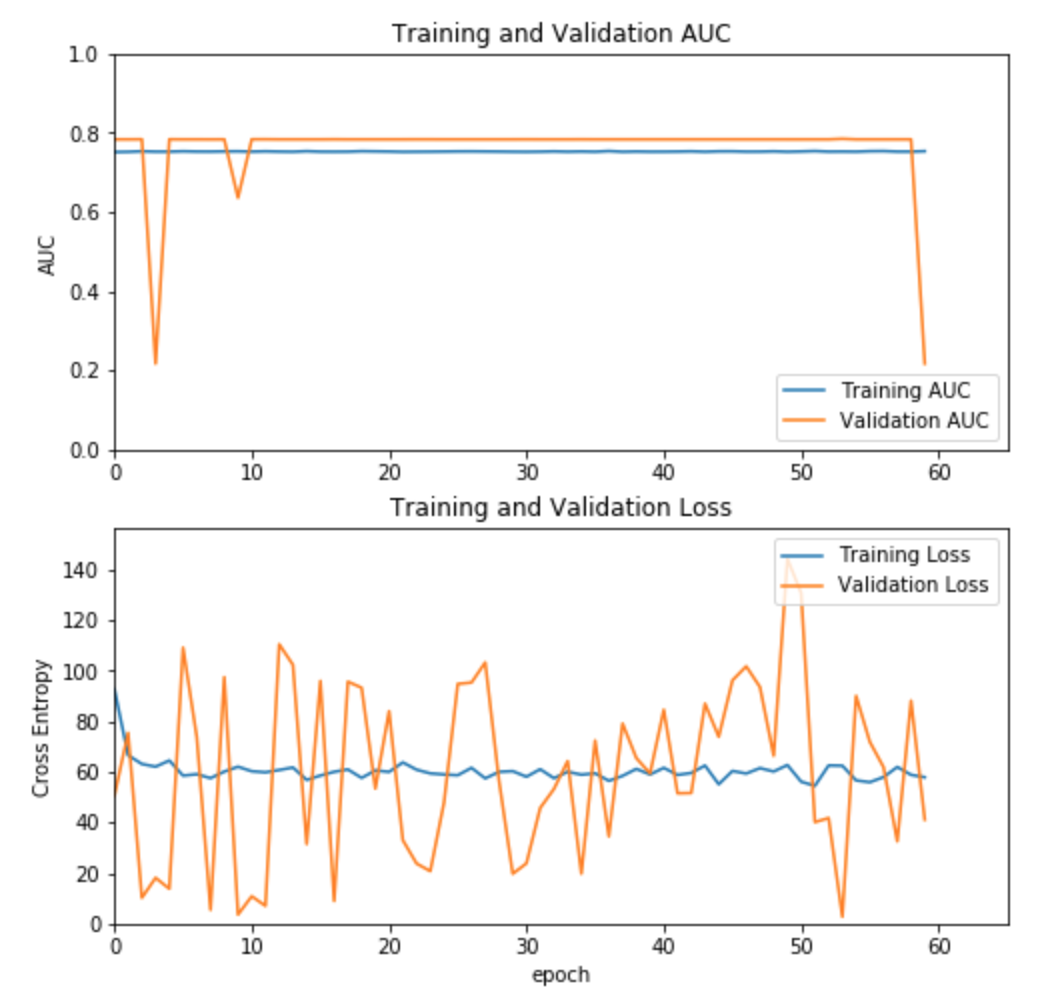
class\_weight = {0:4 , 1:1}

for solving the imbalance class problem.



Change the scripts to

class\_weight = {0:1 , 1:4} # may reversed the class proportion?



Also try:

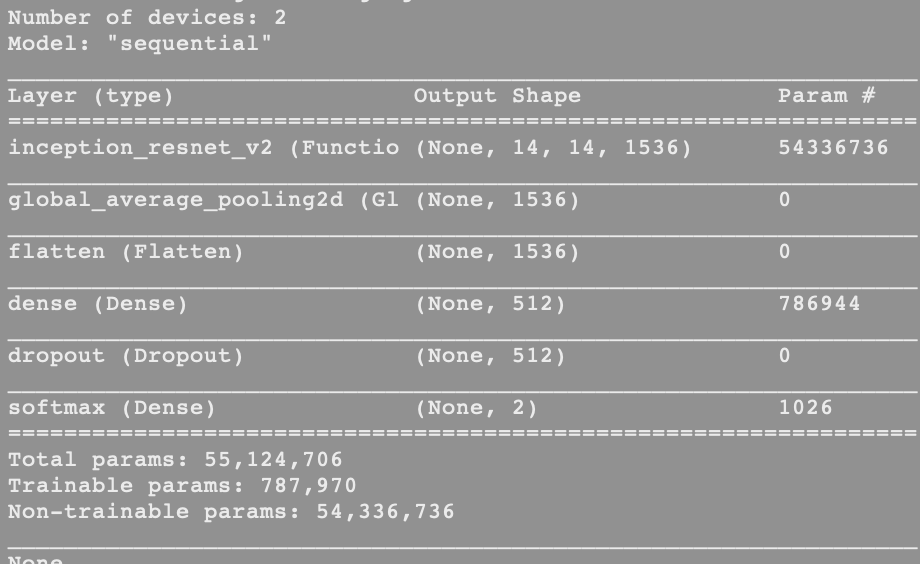
~~class\_weight="balanced"~~

~~Other over-sampling method: SMOTE (Synthetic Minority Over-sampling Technique)~~

Data Augmentation with Keras

Image\_genenator

1. Prep dataset and directories for the use of image generator. – DONE.
2. Adding another dense layer!!

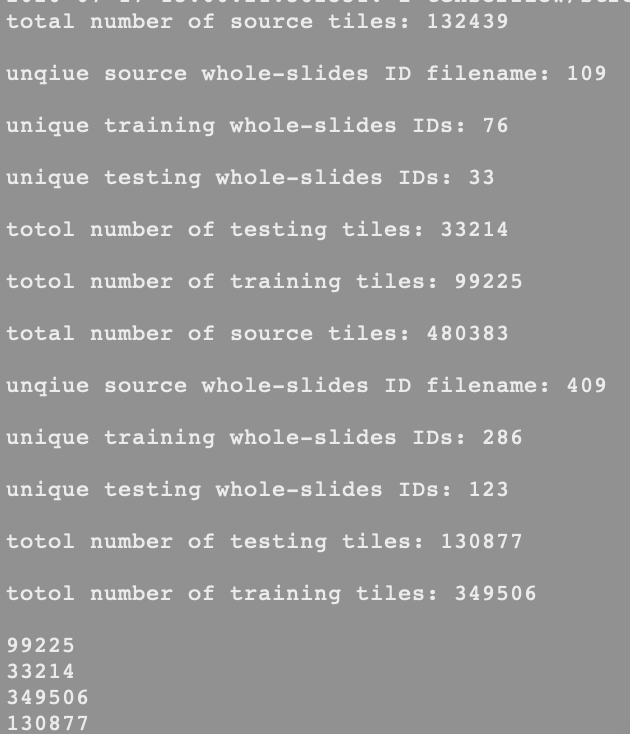


#Change the used pre-trained model?

* Check training tiles are belong to the whole-slide that are not in the validation set.
  + RE-DO SPLIT DATASET => PREP DATASET FOR TRAIN! – DONE
* **Micro-auc DEEPPATH ?**
* **Use inception V3, pre-trained Keras Imagenet model,**
* **Try VGG**
* **Compare the per-tile auc from the paper.**
* Compare at least 3 different models.

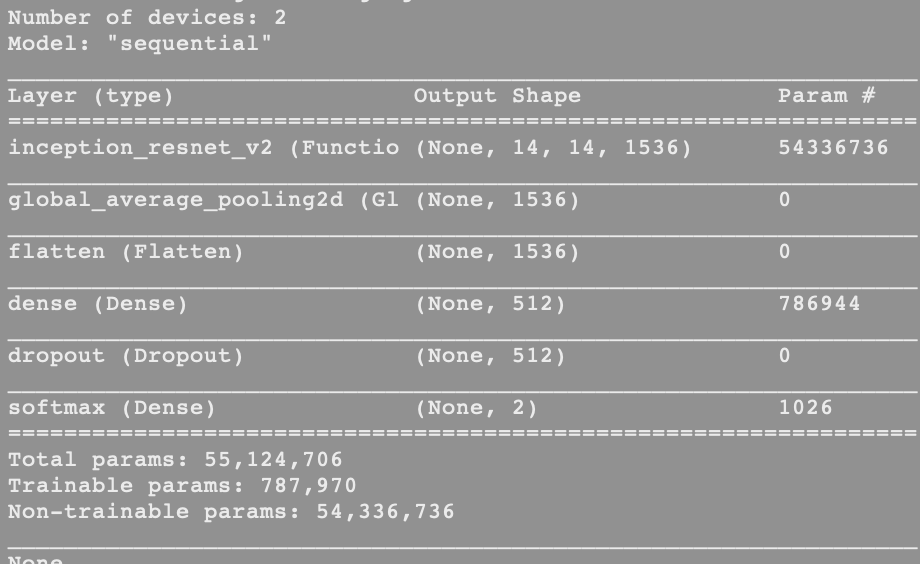
Split by Slides results: (0.7)

Output:



Re-train , oda-compute-0-11,

Batch size = 400, with data augmentation, ResNetV2 pre-trained model + a dense layer with output layer.

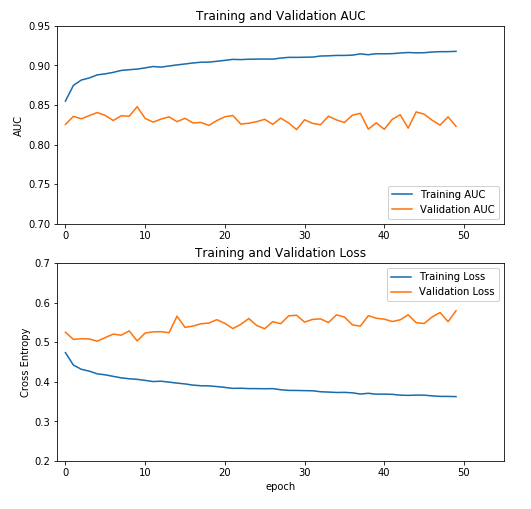


Batch size = 400, with data augmentation, ResNetV2 pre-trained model + a dense layer with output layer. Split by Slides results: (0.7), epochs = 50

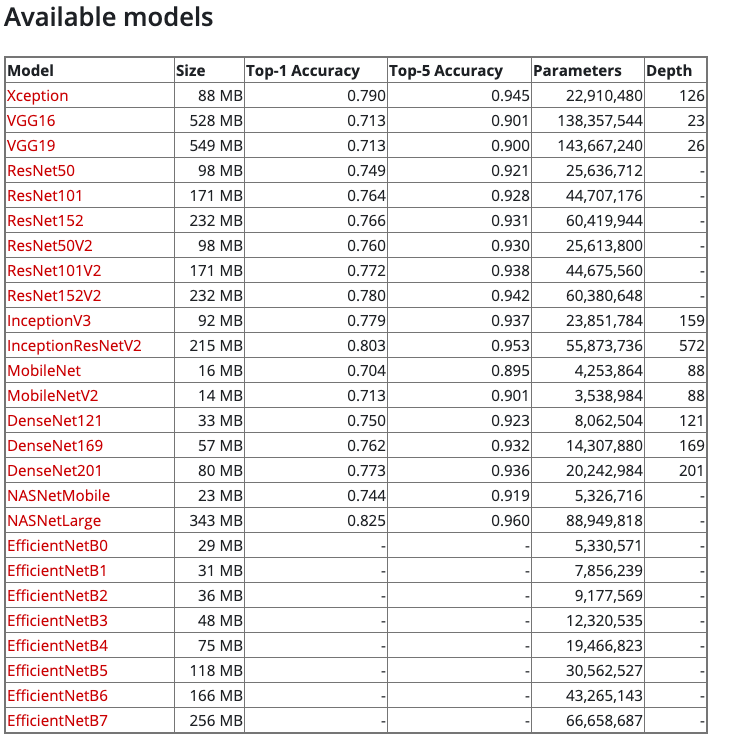
Results: Graph 1 [y-axis from 0 – 1 ]



Graph 2 [y-axis is between smaller interval to zoom the changes]



* Compare at least 3 different models.



Aug 3rd, 2020

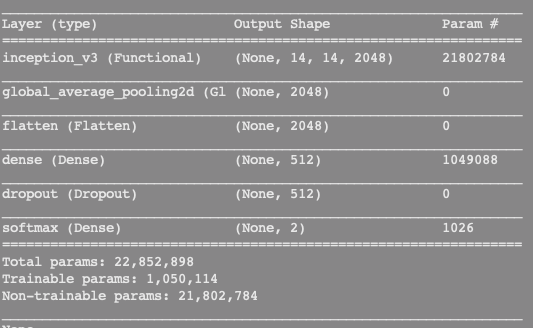
Try with Inception V3 – same dataset used for InceptionResNetV2

22s/step 1100 steps => 1 epoch [with data augmentation step]

50GB (before split ) -> (50GB after split)

Batch size = 400, with data augmentation, InceptionV3 pre-trained model + a dense layer with output layer. Split by Slides results: (0.7), epochs = 50 , shuffle is by default.

Optimizer is RMSprop; InceptionResNetV2 used Adam optimizer.



Aug14, 2020

Inception V3 still running

