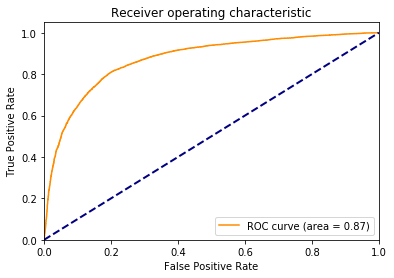
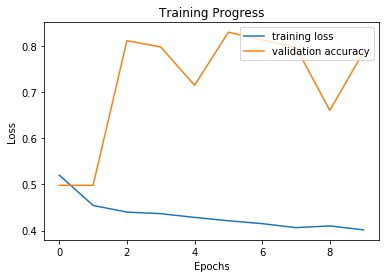
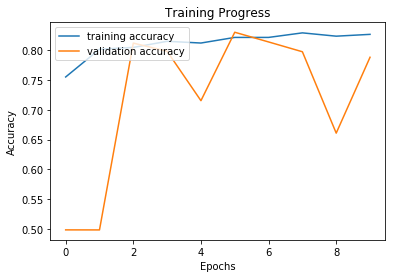
**Results Models**

**ModelRyan2019\_3\_16\_17\_10\_50**

AUC=0.20 dus kan je met np.ones() omzetten naar AUC=0.80 maar niet helemaal zoals het hoort.

**ModelRyan2019\_3\_16\_17\_39\_17**



Submission to Kaggle provided a AUC of 0.8954

**Code bovenste model**

input = Input(shape=(IMAGE\_SIZE,IMAGE\_SIZE,3))

conv1 = Conv2D(64,kernel\_size=(4,4),padding='same',activation='relu',name='conv1')(input)

Batch1=BatchNormalization(name='Batch1')(conv1)

conv2a=Conv2D(32,kernel\_size=(4,4),padding='same',activation='relu',name='conv2a')(Batch1)

Pool2a=MaxPool2D(pool\_size=(4, 4),name='Pool2a')(conv2a)

Batch2a=BatchNormalization(name='Batch2a')(Pool2a)

conv2b=Conv2D(64,kernel\_size=(2,2),padding='same',activation='relu',name='conv2b')(Batch1)

Batch2b=BatchNormalization(name='Batch2b')(conv2b)

Pool2b=MaxPool2D(pool\_size=(2, 2),name='Pool2b')(Batch2b)

conv3b=Conv2D(32,kernel\_size=(8,8),padding='same',activation='relu',name='conv3b')(Pool2b)

Pool3b=MaxPool2D(pool\_size=(2, 2),name='Pool3b')(conv3b)

Batch3b=BatchNormalization(name='Batch3b')(Pool3b)

avg1=average([Batch2a, Batch3b])

conv4=Conv2D(32,kernel\_size=(2,2),padding='valid',activation='relu',name='conv4')(avg1)

Pool4=MaxPool2D(pool\_size=(2, 2),name='Pool4')(conv4)

conv5=Conv2D(32,kernel\_size=(3,3),padding='valid',activation='relu',name='conv5')(Pool4)

Pool5=MaxPool2D(pool\_size=(2, 2),name='Pool5')(conv5)

Glob5=GlobalAveragePooling2D(name='Glob5')(Pool5)

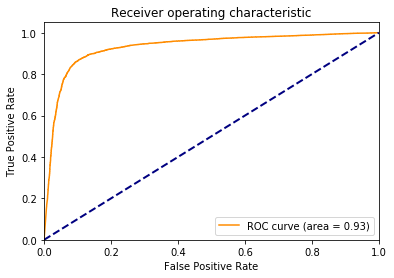
Drop = Dropout(0.5,name='Drop')(Glob5)

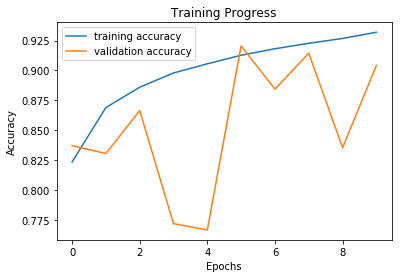
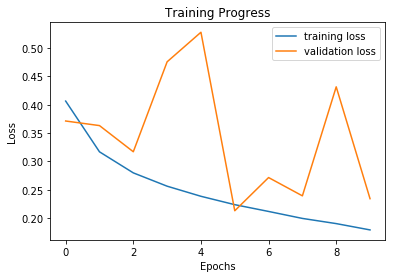
output = Dense(1, activation='sigmoid')(Drop)

model = Model(input=input, output=output)

**ModelRyan2019\_3\_17\_9\_28\_31**

Het vroige model getraind met alle train images levert de volgende resultaten op(dus train\_steps = train\_gen.n//train\_gen.batch\_size):



Ingeleverd bij Kaggle geeft dit een AUC van 0.9400 op voor de test set.