Transfer learning for Multiple classes and Multiple labels

In this notebook, Chest X-Ray dataset has 112120 gray scale images with 1024 by 1024 image size. The dataset contains x-ray images that shows one or more Thorax Disease and the total number of diseases is 14. This make the problem as multiple class and multiple labels problem. The volume of this dataset is around 43GB. In this example, transfer learning is used by training the model from scratch without freezing any layers. The first step is reading images from HDFS and then convert all images into SQL spark dataframe. You can refer to the convert images notebook which reads all images from HDFS and convert them into SQL spark dataframe. The code also assigned with labels for each image by reading CVS file that contains image indexes and their labels.

Vistualize the statistical Information about dataset

The dataset statistical information can be visualized in our dataset statistical data code.

Import the required packages

In this following cell all required packages are downloaded. If you got any error try to install the required package using pip command or download the required library such as BigDL.

In [1]:
```
import random
import time
from math import ceil
from bigdl.optim.optimizer import SGD, SequentialSchedule, Warmup, Poly, P
lateau, EveryEpoch, TrainSummary,\
    ValidationSummary, SeveralIteration, Step, L2Regularizer
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.sql import SparkSession, SQLContext
from pyspark.sql.functions import col, udf
from pyspark.sql.types import DoubleType
from pyspark.storagelevel import StorageLevel
from zoo.common.nncontext import *
from zoo.feature.image.imagePreprocessing import *
from zoo.feature.common import ChainedPreprocessing
from zoo.pipeline.api.keras.layers import Input, Flatten, Dense, GlobalAve
ragePooling2D, Dropout
from zoo.pipeline.api.keras.metrics import AUC
from zoo.pipeline.api.keras.optimizers import Adam
from zoo.pipeline.api.keras.models import Model
from zoo.pipeline.api.net import Net
from zoo.pipeline.nnframes import NNEstimator
from zoo.pipeline.api.keras.objectives import BinaryCrossEntropy
```

```
/usr/lib64/python2.7/site-packages/scipy/sparse/lil.py:16: RuntimeWarning:
numpy.dtype size changed, may indicate binary incompatibility. Expected 96,
```

```
got 88
  from . import _csparsetools
```

Inception pre-trained model

After downloading the pre-trained models and moving them to HDFS, now you can upload and use any of them as shown in the following cells

In [2]:
```
def get_inception_model(model_path, label_length):
    full_model = Net.load_bigdl(model_path)
    model = full_model.new_graph(["pool5/drop_7x7_s1"])  # this inception
    inputNode = Input(name="input", shape=(3, 224, 224))
    inception = model.to_keras()(inputNode)
    flatten = GlobalAveragePooling2D(dim_ordering='th')(inception)
    dropout = Dropout(0.25)(flatten)
    logits = Dense(label_length, W_regularizer=L2Regularizer(1e-1), b_regu
larizer=L2Regularizer(1e-1), activation="sigmoid")(dropout)
    lrModel = Model(inputNode, logits)
    return lrModel
```

In [3]:
```
def get_resnet_model(model_path, label_length):
    full_model = Net.load_bigdl(model_path)
    model = full_model.new_graph(["pool5"])
    print(('num of model layers: ', len(model.layers)))
    inputNode = Input(name="input", shape=(3, 224, 224))
    resnet = model.to_keras()(inputNode)
    flatten = GlobalAveragePooling2D(dim_ordering='th')(resnet)
    dropout = Dropout(0.2)(flatten)
    logits = Dense(label_length, W_regularizer=L2Regularizer(1e-1), b_regu
larizer=L2Regularizer(1e-1), activation="sigmoid")(dropout)
    lrModel = Model(inputNode, logits)
    return lrModel
```

In [4]:
```
def get_vgg_model(model_path, label_length):
    full_model = Net.load_bigdl(model_path)
    model = full_model.new_graph(["pool5"])
    print(('num of model layers: ', len(model.layers)))
    inputNode = Input(name="input", shape=(3, 224, 224))
    vgg_16 = model.to_keras()(inputNode)
    flatten = GlobalAveragePooling2D(dim_ordering='th')(vgg_16)
    dropout = Dropout(0.25)(flatten)
    logits = Dense(label_length, W_regularizer=L2Regularizer(1e-1), b_regu
larizer=L2Regularizer(1e-1), activation="sigmoid")(dropout)
    lrModel = Model(inputNode, logits)
    return lrModel
```

In [5]:
```
def get_densenet_model(model_path, label_length):
    full_model = Net.load_bigdl(model_path)
    model = full_model.new_graph(["pool5"])
    print(('num of model layers: ', len(model.layers)))
    inputNode = Input(name="input", shape=(3, 224, 224))
    densenet = model.to_keras()(inputNode)
    flatten = GlobalAveragePooling2D(dim_ordering='th')(densenet)
    dropout = Dropout(0.25)(flatten)
```

```
    logits = Dense(label_length, W_regularizer=L2Regularizer(1e-1), b_regu
larizer=L2Regularizer(1e-1), activation="sigmoid")(dropout)
    lrModel = Model(inputNode, logits)
    return lrModel
```

Learning Rate Scheduler for SGD optimizer

In [6]:
```
def get_sgd_optimMethod(num_epoch, trainingCount, batchSize):
    iterationPerEpoch = int(ceil(float(trainingCount) / batchSize))
    # maxIteration = num_epoch * iterationPerEpoch
    warmupEpoch = 10
    warmup_iteration = warmupEpoch * iterationPerEpoch
    init_lr = 1e-6
    maxlr = 0.001 * batch_size / 8
    print("peak lr is: ", maxlr)
    warmupDelta = (maxlr - init_lr) / warmup_iteration
    cooldownIteration = (num_epoch - warmupEpoch) * iterationPerEpoch

    lrSchedule = SequentialSchedule(iterationPerEpoch)
    lrSchedule.add(Warmup(warmupDelta), warmup_iteration)
    #lrSchedule.add(Step(iterationPerEpoch * 10, 0.1), cooldownIteration)
    lrSchedule.add(Plateau("Loss", factor=0.1, patience=1, mode="min", eps
ilon=0.01, cooldown=0, min_lr=1e-15 ), cooldownIteration)
    optim = SGD(learningrate=init_lr, momentum=0.9, dampening=0.0, nestero
v=True,
                leaningrate_schedule=lrSchedule)
    return optim
```

Learning Rate Scheduler for ADAM optimizer

In [7]:
```
def get_adam_optimMethod(num_epoch, trainingCount, batchSize):
    iterationPerEpoch = int(ceil(float(trainingCount) / batchSize))
    warmupEpoch = 5
    warmup_iteration = warmupEpoch * iterationPerEpoch
    init_lr = 0.0001 #1e-7
    maxlr = 0.001
    print("peak lr is: ", maxlr)
    warmupDelta = (maxlr - init_lr) / warmup_iteration
    cooldownIteration = (num_epoch - warmupEpoch) * iterationPerEpoch

    lrSchedule = SequentialSchedule(iterationPerEpoch)
    lrSchedule.add(Warmup(warmupDelta), warmup_iteration)
    lrSchedule.add(Plateau("Loss", factor=0.1, patience=1, mode="min", eps
ilon=0.01, cooldown=0, min_lr=1e-15 ),
                   cooldownIteration)
    optim = Adam(lr=init_lr, schedule=lrSchedule)
    return optim
```

Convert class labels into one hot encoding

This function will convert each labels into one hot encoding with multiple labels as sequence of 0's

and 1's

In [8]:
```
def get_auc_for_kth_class(k, df, label_col="label", prediction_col="predic
tion"):
    get_Kth = udf(lambda a: a[k], DoubleType())
    extracted_df = df.withColumn("kth_label", get_Kth(col(label_col))) \
        .withColumn("kth_prediction", get_Kth(col(prediction_col))) \
        .select('kth_label', 'kth_prediction')
    # areaUnderROC|areaUnderPR
    roc_score = BinaryClassificationEvaluator(rawPredictionCol='kth_predict
ion',
                                              labelCol='kth_label',
                                              metricName="areaUnderROC") \
        .evaluate(extracted_df)

    return roc_score
```

Plot AUC

This evaluate and printout auc for all 14 classes

In [9]:
```
%matplotlib notebook
%pylab inline
def plotingAuc(roc_auc_label):
    print ("plot of Area Under Curve for 14 classes ")
    lists=[]
    lists = sorted(roc_auc_label.items())
    label_texts = ["Atelectasis", "Cardiomegaly", "Effusion", "Infiltration
", "Mass", "Nodule", "Pneumonia",
                   "Pneumothorax", "Consolidation","Edema", "Emphysema", "F
ibrosis", "Pleural_Thickening", "Hernia"]
    x, y =zip(*lists)
    label_map = {k: v for v, k in enumerate(label_texts)}
    import numpy as np
    rng = np.random.RandomState(0)
    #matplotlib.use('Agg')
    fig, ax = plt.subplots(figsize=(10, 5))
    sizes = 500 * rng.rand(100)
    colors = ['#005249','#2300A8', '#00A658', '#00A6B8','#00A6BC', '#00AA5
8','#1805db', '#154406', '#631950','#000000','#850e04','#84b701','#adf802'
,'#042e60']
    #print (len(colors))
    plt.ylabel("AUC")
    plt.xlabel("Classes")
    plt.title("AUC for all 14 classes")
    plt.scatter(x, y,  alpha=0.50, color=colors,s=sizes,   cmap='viridis',
marker = '*')
    plt.grid(color='grey', linestyle='-', linewidth=0.5, alpha=0.5)
    ax.set_xticklabels(x, rotation=45 );
    plt.show()
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
/usr/lib/python2.7/site-packages/IPython/core/magics/pylab.py:161: UserWarn
ing: pylab import has clobbered these variables: ['random', 'ceil']
```

```
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```

Evaluating the model and calling plot AUC

In [10]:
```
def evaluate(testDF):
    predictionDF = nnModel.transform(testDF).persist(storageLevel=StorageL
evel.DISK_ONLY)
    label_texts= ["Atelectasis", "Cardiomegaly", "Effusion", "Infiltration"
, "Mass", "Nodule", "Pneumonia",
                  "Pneumothorax", "Consolidation", "Edema", "Emphysema", "
Fibrosis", "Pleural_Thickening", "Hernia"]
    label_map = {k: v for v, k in enumerate(label_texts)}
    total_auc = 0.0
    roc_auc_label =dict()
    for i in range(label_length):
        roc_score = get_auc_for_kth_class(i, predictionDF)
        total_auc += roc_score
        print('{:>12} {:>25} {:>5} {:<20}'.format('roc score for ', label_t
exts[i], ' is: ', roc_score))
        roc_auc_label[i]=(roc_score)
       # print roc_auc_label[i]
    print("Finished evaluation, average auc: ", total_auc / float(label_len
gth))
    plotingAuc(roc_auc_label)
    #plot_auc(total_score)
    print label_map
```

Visualizing training with Jupyter notebook

If you're using Jupyter notebook, you can also draw the training curves using popular plotting tools (e.g. matplotlib) and can display the plots inline.

First, retrieve the summaries as instructed in Retrieve Summary. The retrieved summary is a list of tuples. Each tuple is a recorded event in format (iteration count, recorded value, timestamp). You can convert it to numpy array or dataframe to plot it

Loading the pre-trained model and Chest X-ray image dataset

In [11]:
```
random.seed(1234)
batch_size = 1024 # int(sys.argv[1])
num_epoch = 15 # int(sys.argv[2])
```

Set the path for pre-trained model and Chest X-ray images

In [12]:
```
model_path ="hdfs:///datasets/xray_files/xray/analytics-zoo_resnet-50_imag
```

```
enet_0.1.0.model"
#data_path = "hdfs:///datasets/xray_files/All_ImageDFJan25"
data_path = "hdfs:///datasets/xray_files/stratified_samplingDF"
# save_path = sys.argv[5] #"./save_model"
```

Set the number of classes

In [13]:
```
label_length = 14
```

Initiate Spark session

In [14]:
```
sparkConf = create_spark_conf().setAppName("test_dell_x_ray")
sc = init_nncontext(sparkConf)
spark = SparkSession.builder.config(conf=sparkConf).getOrCreate()
print(sc.master)
```

```
yarn
```

Call the pre-trained model function

In [15]:
```
xray_model = get_resnet_model(model_path, label_length)
```

```
('num of model layers: ', 227)
creating: createZooKerasInput
creating: createZooKerasGlobalAveragePooling2D
creating: createZooKerasDropout
creating: createL2Regularizer
creating: createL2Regularizer
creating: createZooKerasDense
creating: createZooKerasModel
```

Load the Chest X-ray images

In [16]:
```
train_df = spark.read.load(data_path )
```

Splite the dataset into train and validation SQL spark dataframe

In [17]:
```
(trainingDF, validationDF) = train_df.randomSplit([0.7, 0.3])
trainingCount = trainingDF.count()
print("number of training images: ", trainingCount)
print("number of validation images: ", validationDF.count())
```

```
('number of training images: ', 78491)
('number of validation images: ', 33629)
```

Pre-process the iamges (dataset)

In [18]:
```
transformer = ChainedPreprocessing(
            [RowToImageFeature(), ImageCenterCrop(224, 224), ImageRandomPre
processing(ImageHFlip(), 0.5),
             ImageRandomPreprocessing(ImageBrightness(0.0, 32.0), 0.5),
             ImageChannelNormalize(123.68, 116.779, 103.939), ImageMatToTen
sor(), ImageFeatureToTensor()])
```

```
creating: createRowToImageFeature
creating: createImageCenterCrop
creating: createImageHFlip
creating: createImageRandomPreprocessing
creating: createImageBrightness
creating: createImageRandomPreprocessing
creating: createImageChannelNormalize
creating: createImageMatToTensor
creating: createImageFeatureToTensor
creating: createChainedPreprocessing
```

Save training and validation summary

In [19]:
```
train_summary = TrainSummary(log_dir="/home/mahmood/ChestXray/logDirectory/
logs", app_name="test_dell_x_ray")
val_summary = ValidationSummary(log_dir="/home/mahmood/ChestXray/logDirecto
ry/logs", app_name="test_dell_x_ray")
train_summary.set_summary_trigger("LearningRate", SeveralIteration(50))
train_summary.set_summary_trigger("Loss", SeveralIteration(50))
```

```
creating: createTrainSummary
creating: createValidationSummary
creating: createSeveralIteration
creating: createSeveralIteration
```

Out[19]:
```
JavaObject id=o494
```

Call optimizer function and here we are using Adam

In [20]:
```
optim_method = get_adam_optimMethod(num_epoch, trainingCount, batch_size)
```

```
('peak lr is: ', 0.001)
creating: createSequentialSchedule
creating: createWarmup
creating: createPlateau
creating: createZooKerasAdam
```

Set the classifier parameters such as batch size , loss function and validation data frame

In [21]:
```
classifier = NNEstimator(xray_model, BinaryCrossEntropy(), transformer) \
```

```
            .setBatchSize(batch_size) \
            .setMaxEpoch(num_epoch) \
            .setFeaturesCol("image") \
            .setCachingSample(False) \
            .setValidation(EveryEpoch(), validationDF, [AUC()], batch_size)
\
            .setTrainSummary(train_summary) \
            .setValidationSummary(val_summary) \
            .setOptimMethod(optim_method)
```

```
creating: createZooKerasBinaryCrossEntropy
creating: createSeqToTensor
creating: createFeatureLabelPreprocessing
creating: createNNEstimator
creating: createEveryEpoch
creating: createAUC
```

Train the model using fit and print how long may take

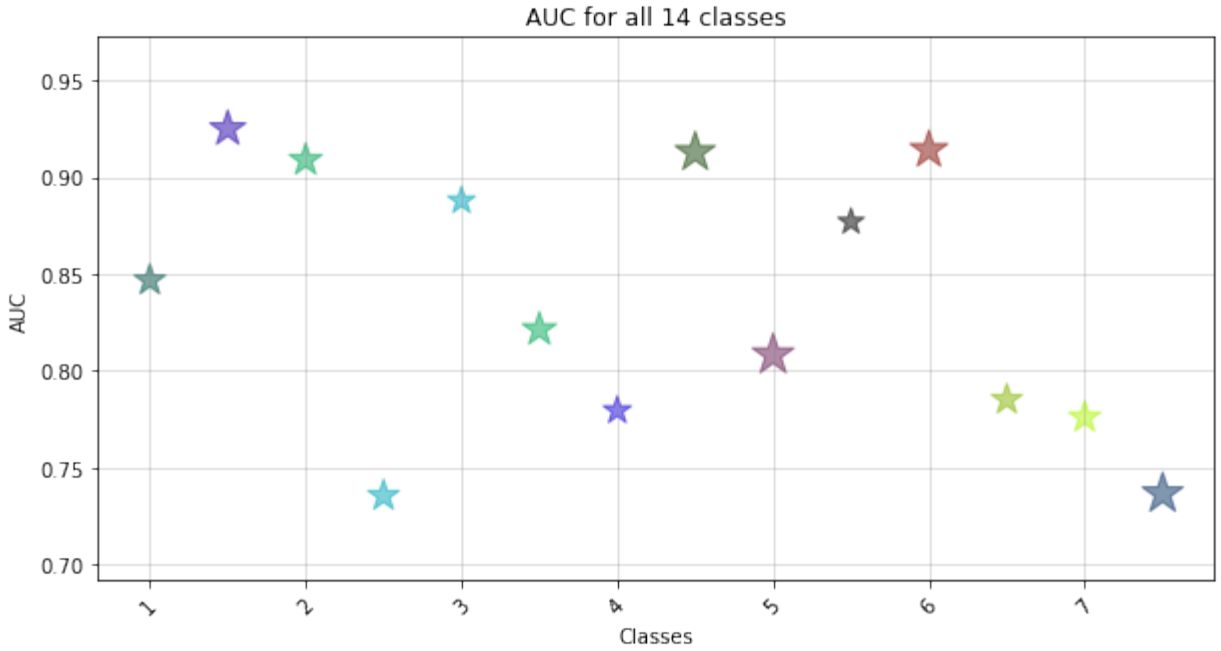
In [22]:
```
start = time.time()
nnModel = classifier.fit(trainingDF)
print("Finished training, took: ", time.time() - start)
```

```
creating: createToTuple
creating: createChainedPreprocessing
('Finished training, took: ', 6329.827677965164)
```

Evaluate the model and plot auc for training

In [23]:
```
print("evaluating on training data: ")
evaluate(trainingDF)
SQLContext(sc).clearCache()
```

```
evaluating on training data:
roc score for               Atelectasis  is:  0.846869636459
roc score for              Cardiomegaly  is:  0.925104862814
roc score for                  Effusion  is:  0.909006984498
roc score for              Infiltration  is:  0.735470437827
roc score for                      Mass  is:  0.887989906839
roc score for                    Nodule  is:  0.821416931477
roc score for                 Pneumonia  is:  0.779538113652
roc score for              Pneumothorax  is:  0.912982306119
roc score for             Consolidation  is:  0.808283912192
roc score for                     Edema  is:  0.877081440328
roc score for                 Emphysema  is:  0.914201681037
roc score for                  Fibrosis  is:  0.785155463572
roc score for        Pleural_Thickening  is:  0.776012375313
roc score for                    Hernia  is:  0.736516580398
('Finished evaluation, average auc: ', 0.8368307594662676)
plot of Area Under Curve for 14 classes
```

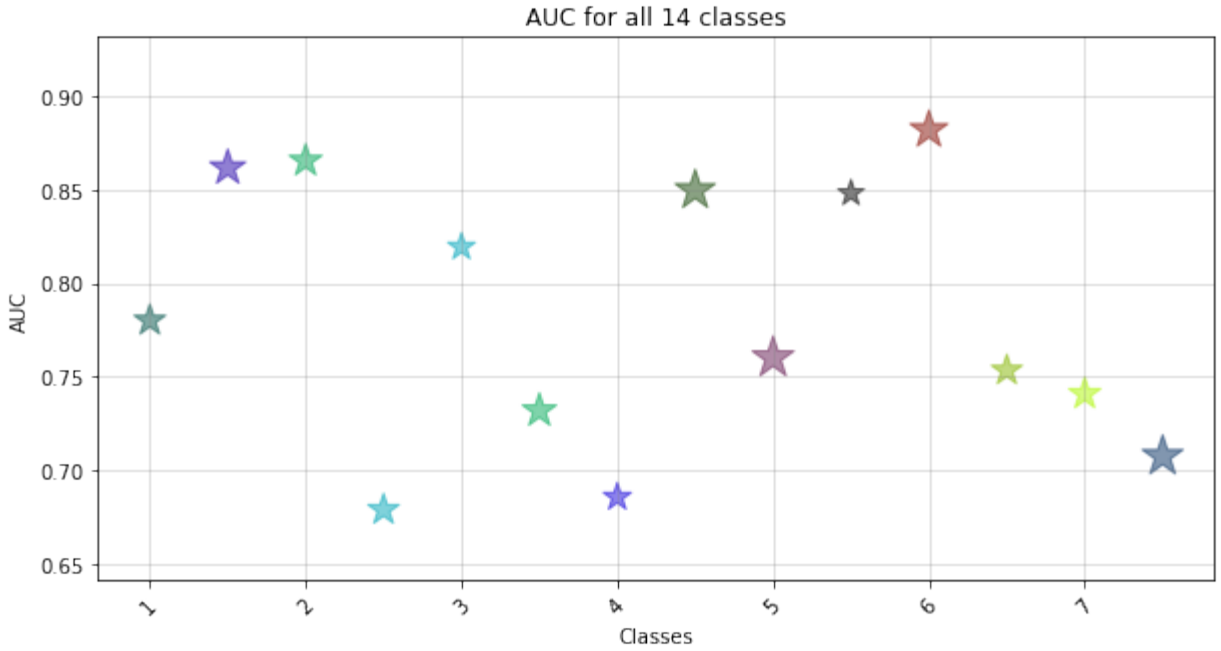



```
{'Effusion': 2, 'Pneumothorax': 7, 'Edema': 9, 'Cardiomegaly': 1, 'Pleural_
Thickening': 12, 'Atelectasis': 0, 'Consolidation': 8, 'Emphysema': 10, 'Pn
eumonia': 6, 'Nodule': 5, 'Mass': 4, 'Infiltration': 3, 'Hernia': 13, 'Fibr
osis': 11}
```

Evaluate the model based on the validation

In [24]:
```
print("\nevaluating on validation  data: ")
evaluate(validationDF)
```

```
evaluating on validation  data:
roc score for               Atelectasis  is:  0.780008270716
roc score for              Cardiomegaly  is:  0.861690840966
roc score for                  Effusion  is:  0.865470780768
roc score for              Infiltration  is:  0.678728872594
roc score for                      Mass  is:  0.81937252896
roc score for                    Nodule  is:  0.731773696509
roc score for                 Pneumonia  is:  0.685494899447
roc score for              Pneumothorax  is:  0.84966410216
roc score for             Consolidation  is:  0.760042028419
roc score for                     Edema  is:  0.848090533671
roc score for                 Emphysema  is:  0.881974314806
roc score for                  Fibrosis  is:  0.753294917453
roc score for        Pleural_Thickening  is:  0.740726466303
roc score for                    Hernia  is:  0.707347400953
('Finished evaluation, average auc: ', 0.7831199752660915)
plot of Area Under Curve for 14 classes
```



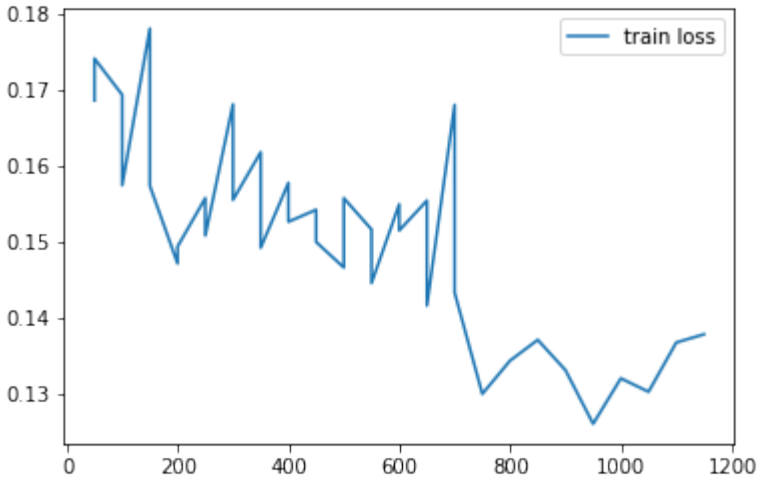

```
{'Effusion': 2, 'Pneumothorax': 7, 'Edema': 9, 'Cardiomegaly': 1, 'Pleural_
Thickening': 12, 'Atelectasis': 0, 'Consolidation': 8, 'Emphysema': 10, 'Pn
eumonia': 6, 'Nodule': 5, 'Mass': 4, 'Infiltration': 3, 'Hernia': 13, 'Fibr
osis': 11}
```

Loss function Graph

In [25]:
```
import matplotlib as plt
import numpy as np

#pylab
%matplotlib notebook
%pylab inline
#import matplotlib.pyplot as plt
pylab.figure()

#retrieve train and validation summary object and read the loss data into n
darray's.
loss = np.array(train_summary.read_scalar("Loss"))

plt.plot(loss[:,0],loss[:,1],label='train loss')
#plt.plot(val_loss[:,0],val_loss[:,1],label='val loss') #,color='green')
#plt.scatter(val_loss[:,0],val_loss[:,1],color='green')
plt.legend();
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
/usr/lib/python2.7/site-packages/IPython/core/magics/pylab.py:161: UserWarn
ing: pylab import has clobbered these variables: ['plt']
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```



In [ ]:
```
!tensorboard --logdir=/home/mahmood/ChestXray/logDirectory/logs/test_dell_x_ray/train --port 8082
```

```
/usr/lib64/python2.7/site-packages/h5py/__init__.py:36: RuntimeWarning: numpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  from ._conv import register_converters as _register_converters
/usr/lib64/python2.7/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
/usr/lib64/python2.7/site-packages/h5py/__init__.py:45: RuntimeWarning: numpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  from . import h5a, h5d, h5ds, h5f, h5fd, h5g, h5r, h5s, h5t, h5p, h5z
/usr/lib64/python2.7/site-packages/h5py/_hl/group.py:22: RuntimeWarning: numpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  from .. import h5g, h5i, h5o, h5r, h5t, h5l, h5p
/usr/lib64/python2.7/site-packages/scipy/sparse/lil.py:16: RuntimeWarning: numpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  from . import _csparsetools
/usr/lib64/python2.7/site-packages/scipy/linalg/basic.py:17: RuntimeWarning: numpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  from ._solve_toeplitz import levinson
/usr/lib64/python2.7/site-packages/scipy/linalg/__init__.py:202: RuntimeWarning: numpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  from ._decomp_update import *
/usr/lib64/python2.7/site-packages/scipy/special/__init__.py:640: RuntimeWarning: numpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  from ._ufuncs import *
/usr/lib64/python2.7/site-packages/scipy/special/_ellip_harm.py:7: RuntimeWarning: numpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  from ._ellip_harm_2 import _ellipsoid, _ellipsoid_norm
/usr/lib64/python2.7/site-packages/scipy/optimize/_trlib/__init__.py:1: Run
```

```
timeWarning: numpy.dtype size changed, may indicate binary incompatibility.
Expected 96, got 88
  from ._trlib import TRLIBQuadraticSubproblem
/usr/lib64/python2.7/site-packages/scipy/optimize/_numdiff.py:8: RuntimeWar
ning: numpy.dtype size changed, may indicate binary incompatibility. Expect
ed 96, got 88
  from ._group_columns import group_dense, group_sparse
/usr/lib64/python2.7/site-packages/scipy/interpolate/_bsplines.py:9: Runtim
eWarning: numpy.dtype size changed, may indicate binary incompatibility. Ex
pected 96, got 88
  from . import _bspl
/usr/lib64/python2.7/site-packages/scipy/spatial/__init__.py:94: RuntimeWar
ning: numpy.dtype size changed, may indicate binary incompatibility. Expect
ed 96, got 88
  from .ckdtree import *
/usr/lib64/python2.7/site-packages/scipy/spatial/__init__.py:95: RuntimeWar
ning: numpy.dtype size changed, may indicate binary incompatibility. Expect
ed 96, got 88
  from .qhull import *
/usr/lib64/python2.7/site-packages/scipy/spatial/_spherical_voronoi.py:18:
RuntimeWarning: numpy.dtype size changed, may indicate binary incompatibili
ty. Expected 96, got 88
  from . import _voronoi
/usr/lib64/python2.7/site-packages/scipy/spatial/distance.py:121: RuntimeWa
rning: numpy.dtype size changed, may indicate binary incompatibility. Expec
ted 96, got 88
  from . import _hausdorff
/usr/lib64/python2.7/site-packages/scipy/ndimage/measurements.py:36: Runtim
eWarning: numpy.dtype size changed, may indicate binary incompatibility. Ex
pected 96, got 88
  from . import _ni_label
/usr/lib64/python2.7/site-packages/pandas/_libs/__init__.py:4: RuntimeWarni
ng: numpy.dtype size changed, may indicate binary incompatibility. Expected
96, got 88
  from .tslib import iNaT, NaT, Timestamp, Timedelta, OutOfBoundsDatetime
/usr/lib64/python2.7/site-packages/pandas/__init__.py:26: RuntimeWarning: n
umpy.dtype size changed, may indicate binary incompatibility. Expected 96,
got 88
  from pandas._libs import (hashtable as _hashtable,
/usr/lib64/python2.7/site-packages/pandas/core/dtypes/common.py:6: RuntimeW
arning: numpy.dtype size changed, may indicate binary incompatibility. Expe
cted 96, got 88
  from pandas._libs import algos, lib
/usr/lib64/python2.7/site-packages/pandas/core/util/hashing.py:7: RuntimeWa
rning: numpy.dtype size changed, may indicate binary incompatibility. Expec
ted 96, got 88
  from pandas._libs import hashing, tslib
/usr/lib64/python2.7/site-packages/pandas/core/indexes/base.py:7: RuntimeWa
rning: numpy.dtype size changed, may indicate binary incompatibility. Expec
ted 96, got 88
  from pandas._libs import (lib, index as libindex, tslib as libts,
/usr/lib64/python2.7/site-packages/pandas/tseries/offsets.py:21: RuntimeWar
ning: numpy.dtype size changed, may indicate binary incompatibility. Expect
ed 96, got 88
  import pandas._libs.tslibs.offsets as liboffsets
/usr/lib64/python2.7/site-packages/pandas/core/ops.py:16: RuntimeWarning: n
umpy.dtype size changed, may indicate binary incompatibility. Expected 96,
```

```
got 88
  from pandas._libs import algos as libalgos, ops as libops
/usr/lib64/python2.7/site-packages/pandas/core/indexes/interval.py:32: Runt
imeWarning: numpy.dtype size changed, may indicate binary incompatibility.
Expected 96, got 88
  from pandas._libs.interval import (
/usr/lib64/python2.7/site-packages/pandas/core/internals.py:14: RuntimeWarn
ing: numpy.dtype size changed, may indicate binary incompatibility. Expecte
d 96, got 88
  from pandas._libs import internals as libinternals
/usr/lib64/python2.7/site-packages/pandas/core/sparse/array.py:33: RuntimeW
arning: numpy.dtype size changed, may indicate binary incompatibility. Expe
cted 96, got 88
  import pandas._libs.sparse as splib
/usr/lib64/python2.7/site-packages/pandas/core/window.py:36: RuntimeWarning
: numpy.dtype size changed, may indicate binary incompatibility. Expected 9
6, got 88
  import pandas._libs.window as _window
/usr/lib64/python2.7/site-packages/pandas/core/groupby/groupby.py:68: Runti
meWarning: numpy.dtype size changed, may indicate binary incompatibility. E
xpected 96, got 88
  from pandas._libs import (lib, reduction,
/usr/lib64/python2.7/site-packages/pandas/core/reshape/reshape.py:30: Runti
meWarning: numpy.dtype size changed, may indicate binary incompatibility. E
xpected 96, got 88
  from pandas._libs import algos as _algos, reshape as _reshape
/usr/lib64/python2.7/site-packages/pandas/io/parsers.py:45: RuntimeWarning:
numpy.dtype size changed, may indicate binary incompatibility. Expected 96,
got 88
  import pandas._libs.parsers as parsers
/usr/lib64/python2.7/site-packages/pandas/io/pytables.py:50: RuntimeWarning
: numpy.dtype size changed, may indicate binary incompatibility. Expected 9
6, got 88
  from pandas._libs import algos, lib, writers as libwriters
W0205 07:10:52.538840 Reloader plugin_event_accumulator.py:549] Detected ou
t of order event.step likely caused by a TensorFlow restart. Purging 738 ex
pired tensor events from Tensorboard display between the previous step: 738
(timestamp: 1549323785.16) and current step: 1 (timestamp: 1549336181.91).
W0205 07:10:52.538840 140037986817792 plugin_event_accumulator.py:549] Dete
cted out of order event.step likely caused by a TensorFlow restart. Purging
738 expired tensor events from Tensorboard display between the previous ste
p: 738 (timestamp: 1549323785.16) and current step: 1 (timestamp: 154933618
1.91).
TensorBoard 1.10.0 at http://pjupyter02.vcse.lab:8082 (Press CTRL+C to quit
)
```