FDA Submission

Your Name:: Kofi Ofuafor

Name of your Device:: Pneumonia detection algorithm

Algorithm Description

1. General Information

Intended Use Statement: This algorithm is intended for use in *assisting the clinician* in acute care setting with a speedy detection of pneumonia within a chest x-ray. The predicate device for this algorithm is a CADx device.

Indications for Use: This algorithm is used as a Pneumonia classifier in a chest x-ray taken in acute medical or surgical admission units, accident and emergency and acute care wards. It is intended for use in people aged 20-70 years with *no prior history* of presence of edema, pleural effusion and pulmonary infiltrations

Device Limitations: The device study was based on images from people of age 20- 70 and was significantly less accurate in the presence of concurrent disease and therefore not recommended for use in the catgory of patients were there is prior presence of disease in a coloumn num_prior_positive in the demographic data. The algorithm was not traing in patients over 80 years old.

Clinical Impact of Performance:

- Performance statistics
  - Accuracy: 81%
  - Precision: 0.16666666666666666
  - Sensitivity: 1
  - Specificity: 0

For this algorithm, missing a detection of Pneumonia is not acceptable as human life and depends on it. False negative may slow down clinicians urgency to look at the patient xray for a physical examination. Therefore False Negative results have more impact than a False positive. Accordingly, the threshold for the algorithm has been weighed in favour of recall(sensitivity).

When a test with high recall returns a negative result, you can be pretty confident that the result is truely negative. Recall does not take into account FP though, so you may still be labelling alot of negative cases as positive. So recall are good for screening tests.

2. Algorithm Design and Function

Algorithm Fow chart The check_dicom function reads in a .dcm file, checks the important fields for our device, and returns a numpy array of just the imaging data.

The function, check_dicom(), is required to check the image type(MODALITY), Body Part Examined (Chest), and Image position (PA or AP) for each DICOM image and check if the input to the algorithm is valid to be predicted by our algorithm or not.

DICOM Checking Steps: There are three steps:

1. Extract the desired attributes *Modality: DX The image type DX is a digital radiography *Body Part Examined: Must be Chest *Patient Position: AP or PA The image position could be AP or PA *Study Description: label This is the label for the chest x-ray.
2. Next the check_dicom function checks each image if the input to the algorithm is valid to be predicted upon by our device.
3. Finally, valid images are included as input and a numpy array of the image data is returned.

Preprocessing Steps: The preprocess_image function used here takes in the image value returned from check_dicom, and the image mean, standard deviation as well as image size corresponding with that of VGG16 input image size of (1,224,224, 3). These inputs are used to standardize and resize the image which is returned as a processed image.

Image augmentation techniques were also applied prior to training the model especially to training data.

CNN Architecture: The pretrained model was loaded and the first 17 layers were rendered untrainable Here is the pretrained model

```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
#1  input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
#2  block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
#3  block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
#4  block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
#5  block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
#6  block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
#7  block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
#8  block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
#9  block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
#10 block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
#11 block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
#12 block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
#13 block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
#14 block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
#15 block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
#16 block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
#17 block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
#18 block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
#19 block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

Con..Layer Block 1

Con. Layer Block2

Con Layer Block 3

Con Layer Block 4

Con Layer Block 5

Seven fully connected layers were added to the pretrained model as classifiers. Of the 7 layers, three were Dropout layers to prevent overfitting.

```
input_1 False
block1_conv1 False
block1_conv2 False
block1_pool False
block2_conv1 False
block2_conv2 False
block2_pool False
block3_conv1 False
block3_conv2 False
block3_conv3 False
block3_pool False
block4_conv1 False
block4_conv2 False
block4_conv3 False
block4_pool False
block5_conv1 False
block5_conv2 False
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
model_1 (Model)              (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              25089000
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               500500
_________________________________________________________________
dropout_3 (Dropout)          (None, 500)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 501
=================================================================
Total params: 40,304,689
Trainable params: 27,949,809
Non-trainable params: 12,354,880
_________________________________________________________________
```

Fully Connected classifier

3. Algorithm Training

Parameters:

- Types of augmentation used during training
  - horizontal_flip = True,
  - vertical_flip = False,
  - height_shift_range= 0.1,
  - width_shift_range=0.1,
  - rotation_range=20,
  - shear_range = 0.1,
  - zoom_range=0.1)
- Batch size
  - Training: 16
  - Validation: 32
  - Performance evaluation: 100
- Optimizer learning rate
  - Adam
  - Learning rate: 5e-6. For 5 trainable layers. Decaying rate per epoch: 0.5

Layers of pre-existing architecture that were frozen 17 Layers of pretrained model were frozen and therefore not trainable

```
block1_conv1 False
block1_conv2 False
block1_pool False
block2_conv1 False
block2_conv2 False
block2_pool False
block3_conv1 False
block3_conv2 False
block3_conv3 False
block3_pool False
block4_conv1 False
block4_conv2 False
block4_conv3 False
block4_pool False
block5_conv1 False
block5_conv2 False
```

Layers of pre-existing architecture that were fine-tuned * Fully connected layers were fine-tuned.

Layers added to pre-existing architecture

One one convulated was layers were fine tune and four layres were added for training

```
Layer (type)                 Output Shape              Param #
=================================================================
model_1 (Model)              (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 1000)              25089000
_________________________________________________________________
dropout_2 (Dropout)          (None, 1000)              0
_________________________________________________________________
dense_2 (Dense)              (None, 500)               500500
```

Fully Connected classifier