## Contents

1 **Installation**  
   1.1 Using virtualenv .................................................. 3  
   1.2 Using docker ....................................................... 4  

2 **Project Overview**  
   2.1 Hardware ........................................................... 5  
   2.2 Software ............................................................ 5  

3 **Usage**  
   3.1 Data Setup ......................................................... 7  
   3.2 Model setup ......................................................... 8  
   3.3 Training Process .................................................. 8  
   3.4 Evaluation Process ............................................... 10
This project contains our 10th place solution for the RSNA Pneumonia Detection Challenge. The team named DASA-FIDI-IARA is composed by: Alesson Scapinello MSc., Bernardo Henz MSc., Daniel Souza MSc, Felipe Kitamura MD MSc, Igor Santos MD and José Venson MSc.

Briefly, our solution is based on a Keras implementation of RetinaNet with resnet101 as backbone. We made some improvements to the original code, such as adding anchor ratios that better fit our training dataset, different data augmentation and hyperparameters.

Below, you have all the information that you need to install and reproduce the results.
We provided two methods to build and run this project: One by using virtualenv, and another one by using docker.

### 1.1 Using virtualenv

1. Firstly clone the projects:
   ```
   git clone https://github.com/alessonscap/keras-retinanet.git
   git clone https://github.com/alessonscap/rsna-challenge-2018.git
   
   export RSNA_PROJECT_PATH=$PWD/rsna-challenge-2018
   export RETINANET_PROJECT_PATH=$PWD/keras-retinanet
   ```

2. Install system-wide requirements to create the environment, we recommend using virtualenvwrapper. Installation guide can be found [here](#).

3. Install other system-wide requirements by running:
   ```
   sudo apt-get install python3-tk
   ```

4. Create an environment to install project dependencies (make sure that you are using python3, if your PYTHONPATH is /usr/bin/python3) by running:
   ```
   mkvirtualenv dfi-pneumonia-detection -p /usr/bin/python3
   ```

5. Activate your environment and install the required pip packages:
   ```
   cd $RSNA_PROJECT_PATH
   workon dfi-pneumonia-detection
   pip install cython numpy # avoids dependency issues
   pip install -r requirements.txt
   ```

6. Create a symbolic link in your `rsna-challenge-2018/retinanet` directory pointing to the `keras-retinanet` directory:
7. Compile Cython code provided by the keras-retinanet project:

```bash
cd keras_retinanet
python setup.py build_ext --inplace
```

### 1.2 Using docker

1. Firstly clone the projects:

```bash
git clone https://github.com/alessonscap/keras-retinanet.git
git clone https://github.com/alessonscap/rsna-challenge-2018.git
export RSNA_PROJECT_PATH=$PWD/rsna-challenge-2018
export RETINANET_PROJECT_PATH=$PWD/keras-retinanet
```

2. Install docker-CE. If you are using Ubuntu, you may follow these instructions.

3. Install the nvidia docker runtime following these instructions.

4. Build our keras-retinanet image by running:

```bash
cd $RETINANET_PROJECT_PATH
docker build -t dfi/keras-retinanet .
```

5. Build our rsna-challenge-2018 image by running:

```bash
cd $RSNA_PROJECT_PATH
docker build -t dfi/rsna-challenge-2018 .
```
Project Overview

Our solution is based on a keras implementation of retinanet, with some modifications to the original project. The project structure is shown below:

```
rsna-challenge-2018
    process_dataset/
        datasets/
            process_data.py
    retinanet/
        rsna_train.py
        rsna_evaluate.py
        rsna_eval.py
        rsna_generator.py
    docs/
        requirements.txt
```

2.1 Hardware

The hardware used for training has the following specifications:

- Ubuntu Desktop Linux
- 1 NVIDIA Tesla V100 32gb
- 5117 Processor 14-core 2.00GHz
- 256GB DDR4 System Memory

2.2 Software

The software used for training consists of, mainly:

- Python 3.6.6
• CUDA 9.0.176
• cuddn 7.1.4

The remaining Python requirements are described in requirements.txt.
Usage

Here we describe the general usage of the DFI Pneumonia Detection project.

3.1 Data Setup

Initially, you must download data from the RSNA Pneumonia Detection Challenge [1]. Assuming that Kaggle API is installed, the dataset can be downloaded using the commands below at the desired path:

```bash
export RSNA_DATA_PATH=<desired-path-to-keep-data>
mkdir -p $RSNA_DATA_PATH/
cd $RSNA_DATA_PATH/
kaggle competitions download -c rsna-pneumonia-detection-challenge
unzip "*.csv.zip"
unzip -d stage_2_test_images stage_2_test_images.zip
unzip -d stage_2_train_images stage_2_train_images.zip
sudo chmod 777 -R . # Allows us to read the csv files
```

The RSNA pneumonia detection challenge provided the training data as a set of patientIds, classes indicating pneumonia or non-pneumonia and bounding boxes for the positive cases.

This information is given in .csv files. However, to easily make multiple tests with different approaches, we adapted .csv files to the COCO format. To create the json files with the dataset information for training and testing you must run `process_data.py` passing some arguments, such as:

- **input_csv_train**: Path to stage_1_train_labels.csv file.
- **class_info_csv_path**: Path to class_info_csv_path.csv file.
- **input_train_images_path**: Path to rsna-challenge-2018 dataset.
- **output_train_json_name**: Name to json with train dataset.
- **output_validation_json_name**: Name to json with validation dataset.
- **output_test_json_name**: Name to json with test dataset.
• **split**: (Optional) Values to split dataset into train, val and test. (Default [70 20 10]).

The json files used in stage 1 and 2 can be found at rsna-challenge-2018/process_dataset/datasets.

If you wish to recreate the json file used in stage 2, you can use the commands below:

Using virtualenv

```
cd $RSNA_PROJECT_PATH/process_dataset
python process_data.py $RSNA_DATA_PATH/stage_2_train_labels.csv $RSNA_DATA_PATH/stage_2_detailed_class_info.csv $RSNA_DATA_PATH/stage_2_train_images train.json val.json test.json --split 80 20 0
```

Using docker

```
```

### 3.2 Model setup

The models used in the RSNA challenge can be downloaded by running:

```
export RSNA_MODELS_PATH=<desired-path-to-keep-models>
mkdir -p $RSNA_MODELS_PATH/
cd $RSNA_MODELS_PATH/
wget https://iarahealth.com/rsna/stage_1.h5
wget https://iarahealth.com/rsna/stage_2.h5
```

### 3.3 Training Process

Once you finished all installation steps and data setup, you can train your first model. To do this, you need to run `rsna_train.py` passing some arguments, such as:

- **epochs**: Number of epochs to train.
- **backbone**: Backbone model used by RetinaNet.
- **batch-size**: Size of the batches.
- **steps**: Number of steps per epoch.
- **data-aug**: Enables data augmentation.
- **noise_aug_std**: Defines the STD of the random noise added during training. If noise_aug_std=None, no noise is added.
- **anchor_boxes**: List of anchor boxes used during training.
- **dropout_rate**: Defines the dropout rate used during training.
- **snapshot-path**: Path to store snapshots of models during training.
- **tensorboard_dir**: Log directory for Tensorboard output.
- **multi-gpu**: (Optional) Number of GPUs to use for parallel processing.
• **multi-gpu-force**: Extra flag needed to enable multi-gpu support.

Also, you must pass the required arguments.

• **rsna**: Argument to define that RSNA dataset is used during training.

• **rsna_path**: Path to train dataset directory.

• **rsna_train_json**: Path to training json.

• **rsna_val_json**: Path to validation json.

To reproduce our stage 1 training process, you should use the commands below:

**Virtualenv**

```
cd $RSNA_PROJECT_PATH/retinanet/
python rsna_train.py --epochs 40 --backbone resnet101 --batch-size 14 --steps 1250 --
--data-aug --noise_aug_std 0.05 --anchor_boxes 0.25,0.33,0.5,0.75,1,1.33,2,3,4 --
--dropout_rate 0.1 --snapshot-path snapshots/ --tensorboard_dir logs/ rsna $RSNA_DATA_
--PATH/stage_1_train_images ..../process_dataset/datasets/stage_1_train.json ..../process_--
--dataset/datasets/stage_1_val.json
```

**Docker**

```
docker run --runtime=nvidia -it --rm -v $RSNA_DATA_PATH:/data dfi/rsna-challenge-2018
--python /rsna-challenge-2018/retinanet/rsna_train.py --epochs 40 --backbone
--resnet101 --batch-size 14 --steps 1250 --data-aug --noise_aug_std 0.05 --anchor_
--boxes 0.25,0.33,0.5,0.75,1,1.33,2,3,4 --dropout_rate 0.1 --snapshot-path snapshots/
----tensorboard_dir logs/ rsna /data/stage_1_train_images /rsna-challenge-2018/
--process_dataset/datasets/stage_1_train.json /rsna-challenge-2018/process_dataset/
--datasets/stage_1_val.json
```

To reproduce our stage 2 training process, you should use the commands below:

**Virtualenv**

```
cd $RSNA_PROJECT_PATH/retinanet/
python rsna_train.py --epochs 40 --backbone resnet101 --batch-size 14 --steps 1250 --
--data-aug --noise_aug_std 0.05 --anchor_boxes 0.25,0.33,0.5,0.75,1,1.33,2,3,4 --
--dropout_rate 0.1 --snapshot-path snapshots/ --tensorboard_dir logs/ rsna $RSNA_DATA_
--PATH/stage_2_train_images ..../process_dataset/datasets/stage_2_train.json ..../process_--
--dataset/datasets/stage_2_val.json
```

**Docker**

```
docker run --runtime=nvidia -it --rm -v $RSNA_DATA_PATH:/data dfi/rsna-challenge-2018
--python /rsna-challenge-2018/retinanet/rsna_train.py --epochs 40 --backbone
--resnet101 --batch-size 14 --steps 1250 --data-aug --noise_aug_std 0.05 --anchor_
--boxes 0.25,0.33,0.5,0.75,1,1.33,2,3,4 --dropout_rate 0.1 --snapshot-path snapshots/
----tensorboard_dir logs/ rsna /data/stage_2_train_images /rsna-challenge-2018/
--process_dataset/datasets/stage_2_train.json /rsna-challenge-2018/process_dataset/
--datasets/stage_2_val.json
```

If you want to quickly benchmark a model, you can change the steps and epochs parameters to: --steps 10 and --epochs 4.
3.4 Evaluation Process

Once you already have a trained model, you can evaluate and calculate the mAP score. To do this, you need to run `rsna_evaluate.py` passing some arguments, such as:

- **backbone**: Backbone model used by RetinaNet.
- **convert-model**: Convert the model to an inference model.
- **anchor_boxes**: Same list of anchor boxes used during training.
- **score-threshold**: Threshold on score to filter detections with.
- **nms_threshold**: Non maximum suppression threshold.
- **save-path**: (Optional) Path for saving images with detections.
- **kaggle_output_file**: Path to generate kaggle submission file.

Also, you must pass these required arguments:

- **rsna_path**: Path to test dataset directory.
- **rsna_test_json**: Path to test json.
- **model**: Path to trained model.

To reproduce our stage 1 evaluation process, you should use the commands below:

**Virtualenv**

```
cd $RSNA_PROJECT_PATH/retinanet/
python rsna_evaluate.py --backbone resnet101 --convert-model --score-threshold 0.2 --nms_threshold 0.1 --save-path out/stage_1 --kaggle_output_file stage_1.csv --anchor_boxes 0.25,0.33,0.5,0.75,1.33,2,3,4 $RSNA_DATA_PATH/stage_2_train_images ../process_dataset/datasets/rsna_test_stage_1.json $RSNA_MODELS_PATH/stage_1.h5
```

**Docker**

```
docker run --runtime=nvidia -it --rm -v $RSNA_DATA_PATH:/data -v $RSNA_MODELS_PATH:/models dfi/rsna-challenge-2018 python /rsna-challenge-2018/retinanet/rsna_evaluate.py --backbone resnet101 --convert-model --score-threshold 0.2 --nms_threshold 0.1 --save-path /data/out/stage_1 --kaggle_output_file /data/stage_1.csv --anchor_boxes 0.25,0.33,0.5,0.75,1.33,2,3,4 /data/stage_2_train_images /rsna-challenge-2018/process_dataset/datasets/rsna_test_stage_1.json /models/stage_1.h5
```

To reproduce our stage 2 evaluation process, you should use the commands below:

**Virtualenv**

```
cd $RSNA_PROJECT_PATH/retinanet/
python rsna_evaluate.py --backbone resnet101 --convert-model --score-threshold 0.2 --nms_threshold 0.1 --save-path out/stage_2 --kaggle_output_file stage_2.csv --anchor_boxes 0.25,0.33,0.5,0.75,1.33,2,3,4 $RSNA_DATA_PATH/stage_2_test_images ../process_dataset/datasets/rsna_test_stage_2.json $RSNA_MODELS_PATH/stage_2.h5
```

**Docker**

```
docker run --runtime=nvidia -it --rm -v $RSNA_DATA_PATH:/data -v $RSNA_MODELS_PATH:/models dfi/rsna-challenge-2018 python /rsna-challenge-2018/retinanet/rsna_evaluate.py --backbone resnet101 --convert-model --score-threshold 0.2 --nms_threshold 0.1 --save-path /data/out/stage_2 --kaggle_output_file /data/stage_2.csv --anchor_boxes 0.25,0.33,0.5,0.75,1.33,2,3,4 /data/stage_2_test_images /rsna-challenge-2018/process_dataset/datasets/rsna_test_stage_2.json /models/stage_2.h5
```
(continued from previous page)

3.4. Evaluation Process

11