```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.
#import kagglehub
#paultimothymooney_chest_xray_pneumonia_path = kagglehub.dataset_download('paultimothymooney
#awsaf49_pneumonia_chest_xray_npy_path = kagglehub.dataset_download('awsaf49/pneumonia-chest

#print('Data source import complete.')
```

Import Libraries

```
import numpy as np # linear algebra
import pandas as pd
from tqdm import tqdm
from glob import glob
import os
import numpy as np

import matplotlib.pyplot as plt

#import tensorflow as tf
```

```
!pip install -U tensorflow
```

```
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-
Collecting tensorboard~=2.19.0 (from tensorflow)
```

```
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (fro
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/li
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages
Downloading tensorflow-2.19.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 644.9/644.9 MB 2.7 MB/s eta 0:00:00
Downloading ml_dtypes-0.5.1-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.wh
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.7/4.7 MB 64.3 MB/s eta 0:00:00
Downloading tensorboard-2.19.0-py3-none-any.whl (5.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 75.4 MB/s eta 0:00:00
Installing collected packages: ml-dtypes, tensorboard, tensorflow
  Attempting uninstall: ml-dtypes
    Found existing installation: ml-dtypes 0.4.1
    Uninstalling ml-dtypes-0.4.1:
      Successfully uninstalled ml-dtypes-0.4.1
  Attempting uninstall: tensorboard
    Found existing installation: tensorboard 2.18.0
    Uninstalling tensorboard-2.18.0:
      Successfully uninstalled tensorboard-2.18.0
  Attempting uninstall: tensorflow
    Found existing installation: tensorflow 2.18.0
    Uninstalling tensorflow-2.18.0:
      Successfully uninstalled tensorflow-2.18.0
ERROR: pip's dependency resolver does not currently take into account all the package
tensorflow-text 2.18.1 requires tensorflow<2.19,>=2.18.0, but you have tensorflow 2.1
tensorflow-decision-forests 1.11.0 requires tensorflow==2.18.0, but you have tensorfl
tf-keras 2.18.0 requires tensorflow<2.19,>=2.18, but you have tensorflow 2.19.0 which
Successfully installed ml-dtypes-0.5.1 tensorboard-2.19.0 tensorflow-2.19.0
```

Extrating files

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
!ls /content/drive/MyDrive/pfc2T1
```

```
archive
```

```
import numpy as np

base_path = '/content/drive/MyDrive/pfc2T1/archive/Array128/Array128'
!ls '/content/drive/MyDrive/pfc2T1/Array128/Array128'
normal = np.load(f'{base_path}/train_Normal_128.npy')
viral = np.load(f'{base_path}/train_Virus_128.npy')
bacterial = np.load(f'{base_path}/train_bacteria_128.npy')
```

```
ls: cannot access '/content/drive/MyDrive/pfc2T1/Array128/Array128': No such file or dir
```

Loading Training Files

I have converted all images to numpy array to boost speed

```
#normal = np.load('../input/pneumonia-chest-xray-npy/Array128/Array128/train_Normal_128.npy'
#viral = np.load('../input/pneumonia-chest-xray-npy/Array128/Array128/train_Virus_128.npy')
#bacterial = np.load('../input/pneumonia-chest-xray-npy/Array128/Array128/train_bacteria_128
```

```
normal.shape, viral.shape, bacterial.shape
```

```
((1341, 128, 128, 1), (1345, 128, 128, 1), (2530, 128, 128, 1))
```

Create Labels

```
label_normal = np.zeros(len(normal))
label_bacterial = np.ones(len(bacterial))
label_viral = np.full(len(viral),2, dtype = int)
```

```
train_data = np.concatenate((normal,bacterial,viral),axis=0)
train_label = np.concatenate((label_normal,label_bacterial,label_viral),axis=0)
```

```
train_label.shape, train_data.shape
```

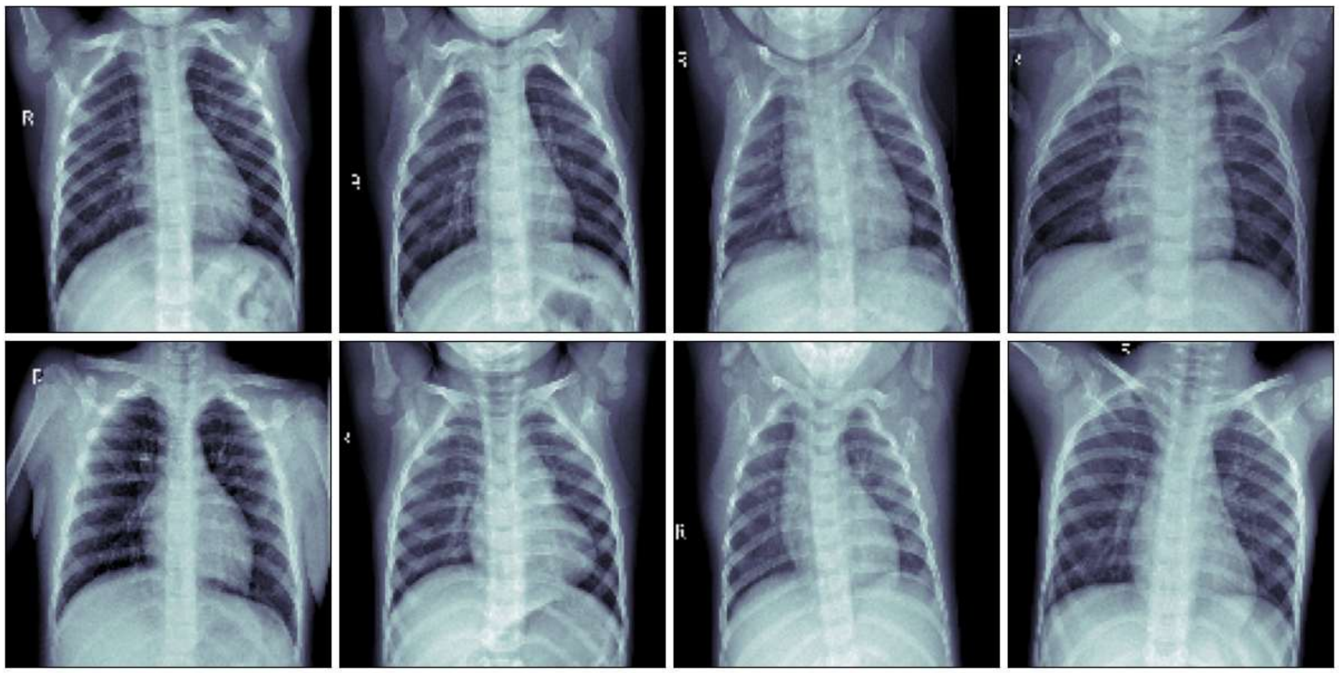
```
((5216,), (5216, 128, 128, 1))
```

Visualization

Normal

```
n_row = 2
n_col = 4

fig, ax = plt.subplots(n_row, n_col, figsize = (n_col*3, n_row*3), constrained_layout = True

for row in tqdm(range(n_row)):

    for col in range(n_col):

        ax[row][col].imshow(normal[row*n_col + col,:,:,0], cmap = 'bone')
        ax[row][col].set_xticks([])
        ax[row][col].set_yticks([])
```

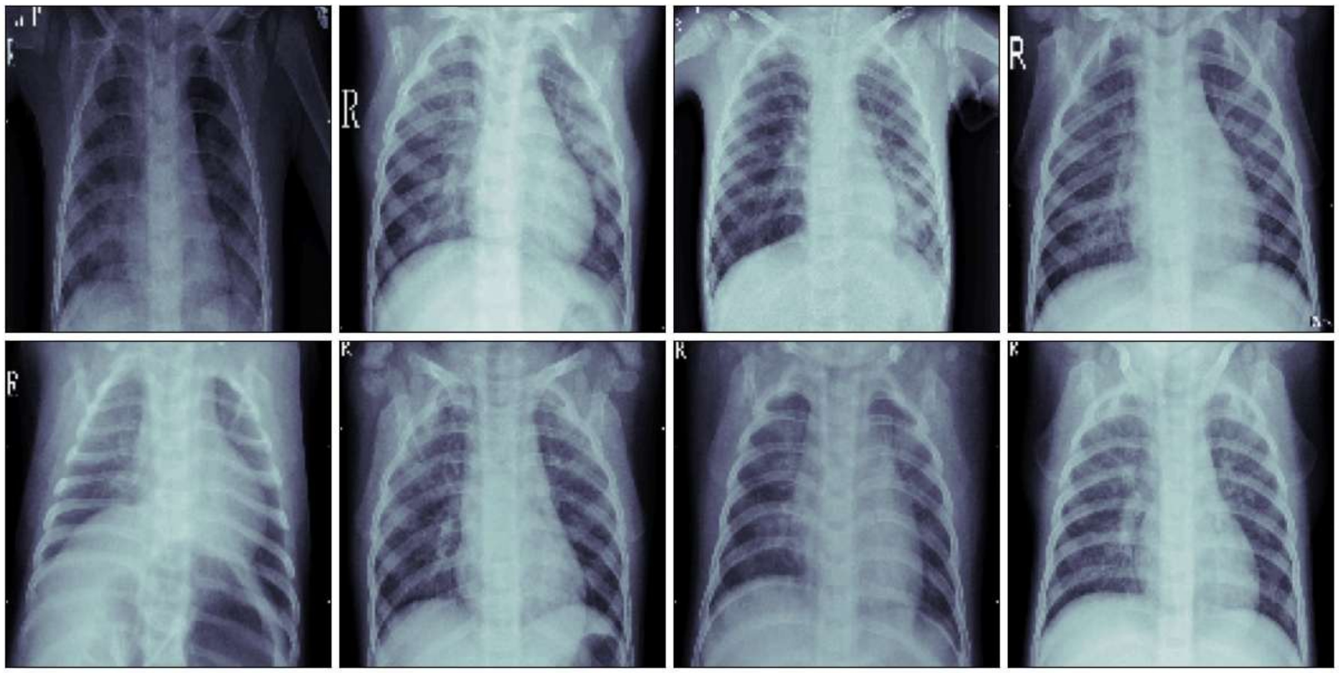
```
100%|██████████| 2/2 [00:00<00:00, 165.34it/s]
```



Viral Pneumonia

```
n_row = 2
n_col = 4

fig, ax = plt.subplots(n_row, n_col, figsize = (n_col*3, n_row*3), constrained_layout = True

for row in tqdm(range(n_row)):

    for col in range(n_col):

        ax[row][col].imshow(viral[row*n_col + col,:,:,0], cmap = 'bone')
        ax[row][col].set_xticks([])
        ax[row][col].set_yticks([])
```

```
100%|██████████| 2/2 [00:00<00:00, 146.58it/s]
```

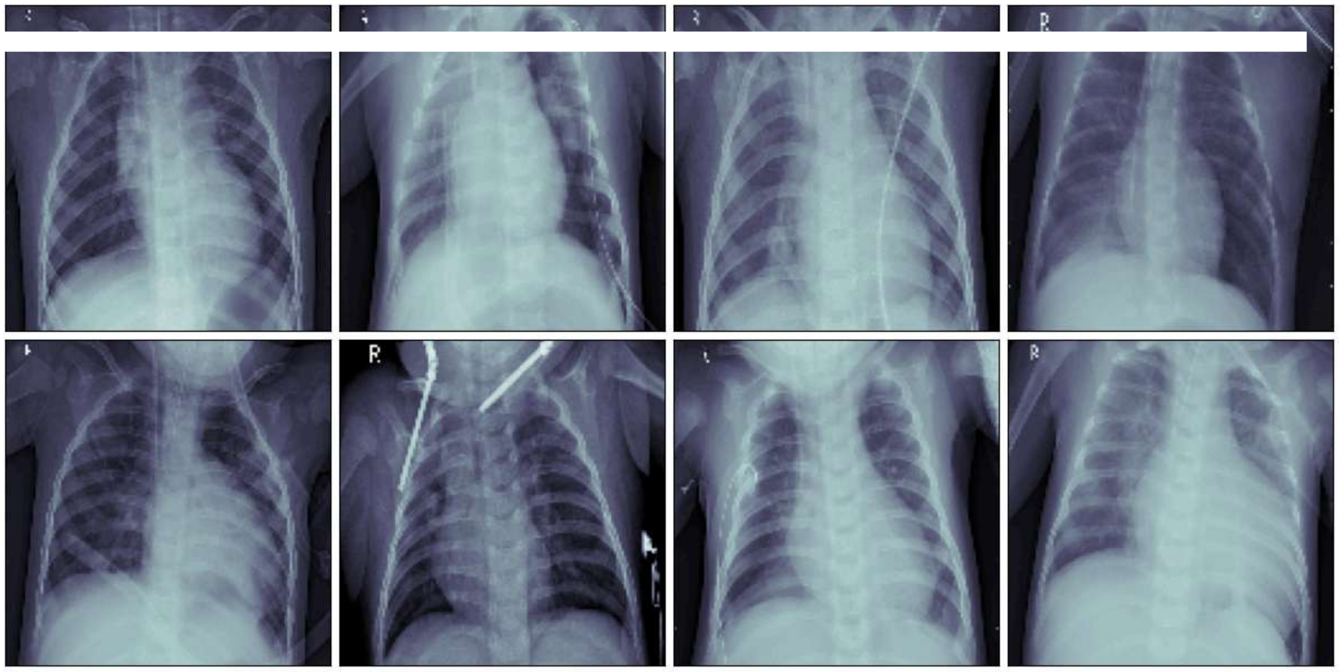


Bacterial Pneumonia

```
n_row = 2
n_col = 4
```

```
fig, ax = plt.subplots(n_row, n_col, figsize = (n_col*3, n_row*3), constrained_layout = True

for row in tqdm(range(n_row)):

    for col in range(n_col):

        ax[row][col].imshow(bacterial[row*n_col + col,:,:,0], cmap = 'bone')
        ax[row][col].set_xticks([])
        ax[row][col].set_yticks([])
```

```
100%|██████████| 2/2 [00:00<00:00, 265.87it/s]
```



Loading Test Data

```
#test_normal = np.load('../input/pneumonia-chest-xray-npy/Array128/Array128/test_Normal_128.
#test_viral = np.load('../input/pneumonia-chest-xray-npy/Array128/Array128/test_Virus_128.np
#test_bacterial = np.load('../input/pneumonia-chest-xray-npy/Array128/Array128/test_bacteria
test_normal = np.load(f'{base_path}/test_Normal_128.npy')
test_viral = np.load(f'{base_path}/test_Virus_128.npy')
test_bacterial = np.load(f'{base_path}/test_bacteria_128.npy')
```

```
test_normal.shape, test_viral.shape , test_bacterial.shape
```

```
((234, 128, 128, 1), (148, 128, 128, 1), (242, 128, 128, 1))
```

Create Labels

```
label_test_normal = np.zeros(len(test_normal))
label_test_bacterial = np.ones(len(test_bacterial))
label_test_viral = np.full(len(test_viral),2, dtype = int)
```

```
test_data = np.concatenate((test_normal, test_bacterial, test_viral),axis=0)
test_label = np.concatenate((label_test_normal,label_test_bacterial,label_test_viral),axis=
```

```
test_label.shape, test_data.shape
```

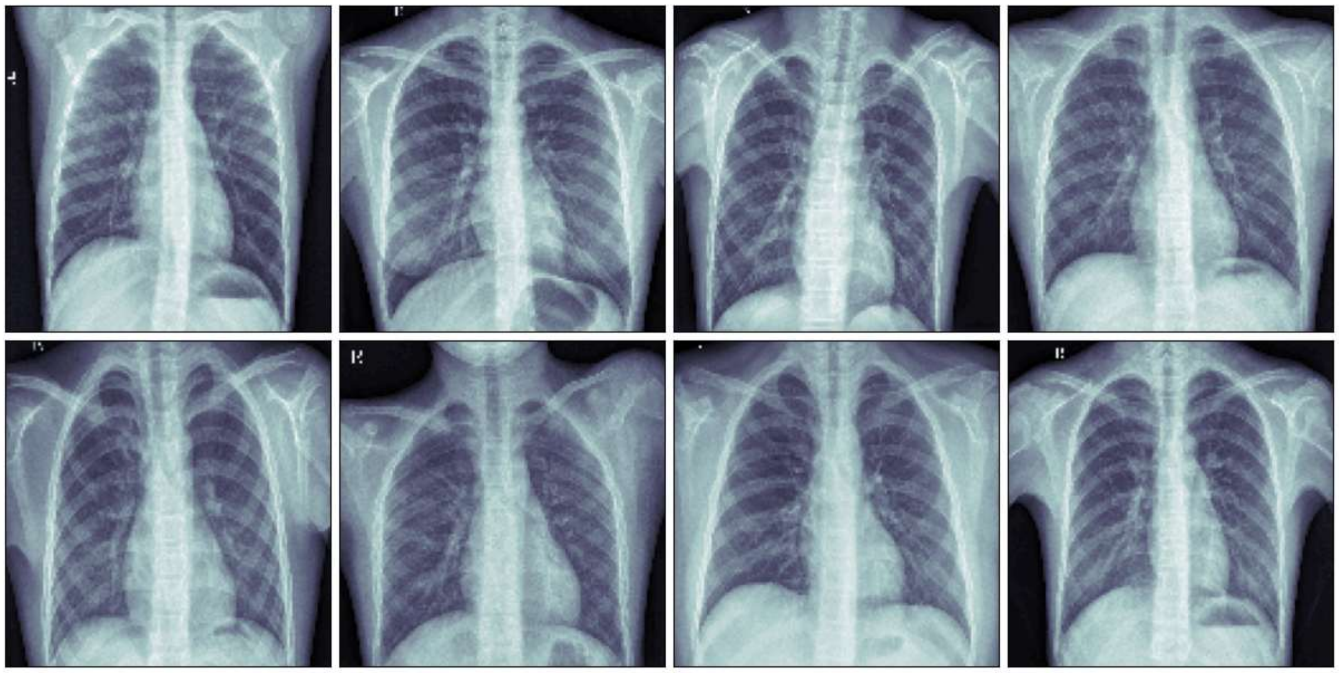
```
((624,), (624, 128, 128, 1))
```

Visualization

Normal

```
n_row = 2
n_col = 4

fig, ax = plt.subplots(n_row, n_col, figsize = (n_col*3, n_row*3), constrained_layout = True

for row in tqdm(range(n_row)):

    for col in range(n_col):

        ax[row][col].imshow(test_normal[row*n_col + col,:,:,0], cmap = 'bone')
        ax[row][col].set_xticks([])
        ax[row][col].set_yticks([])
```

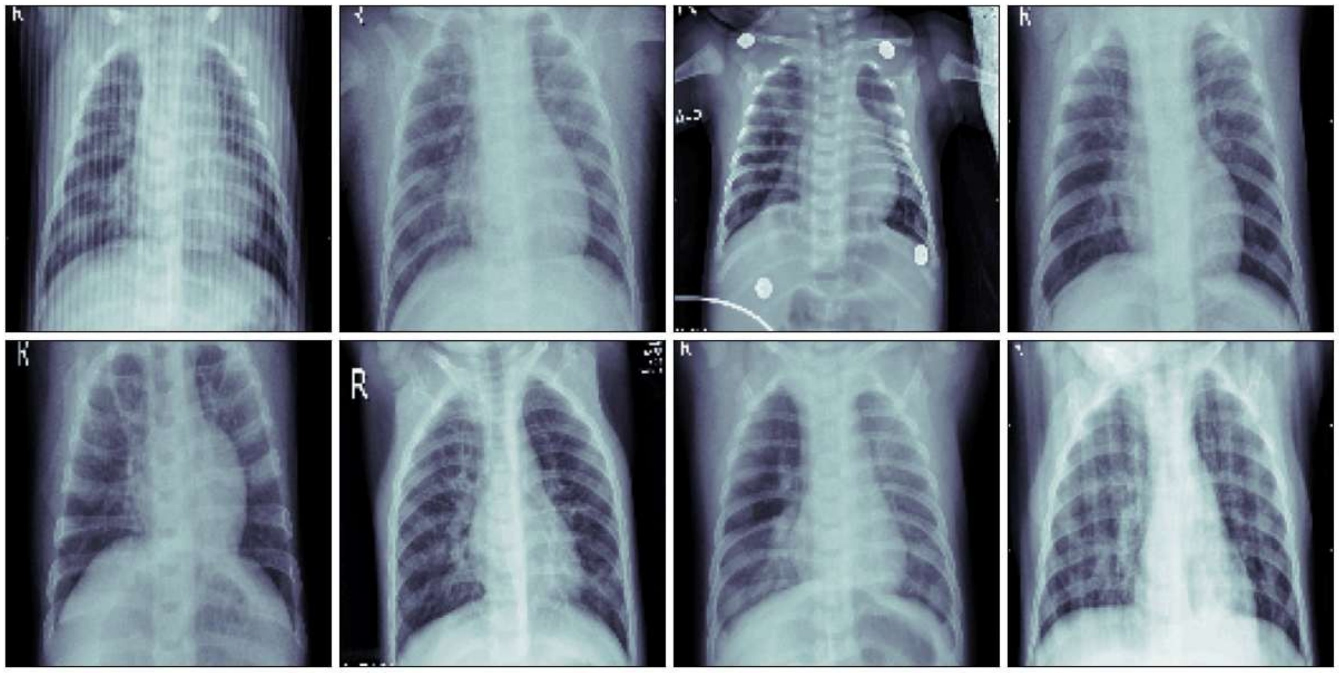
```
100%|██████████| 2/2 [00:00<00:00, 205.41it/s]
```



Viral Pneumonia

```
n_row = 2
n_col = 4

fig, ax = plt.subplots(n_row, n_col, figsize = (n_col*3, n_row*3), constrained_layout = True

for row in tqdm(range(n_row)):

    for col in range(n_col):

        ax[row][col].imshow(test_viral[row*n_col + col,:,:,0], cmap = 'bone')
        ax[row][col].set_xticks([])
        ax[row][col].set_yticks([])
```

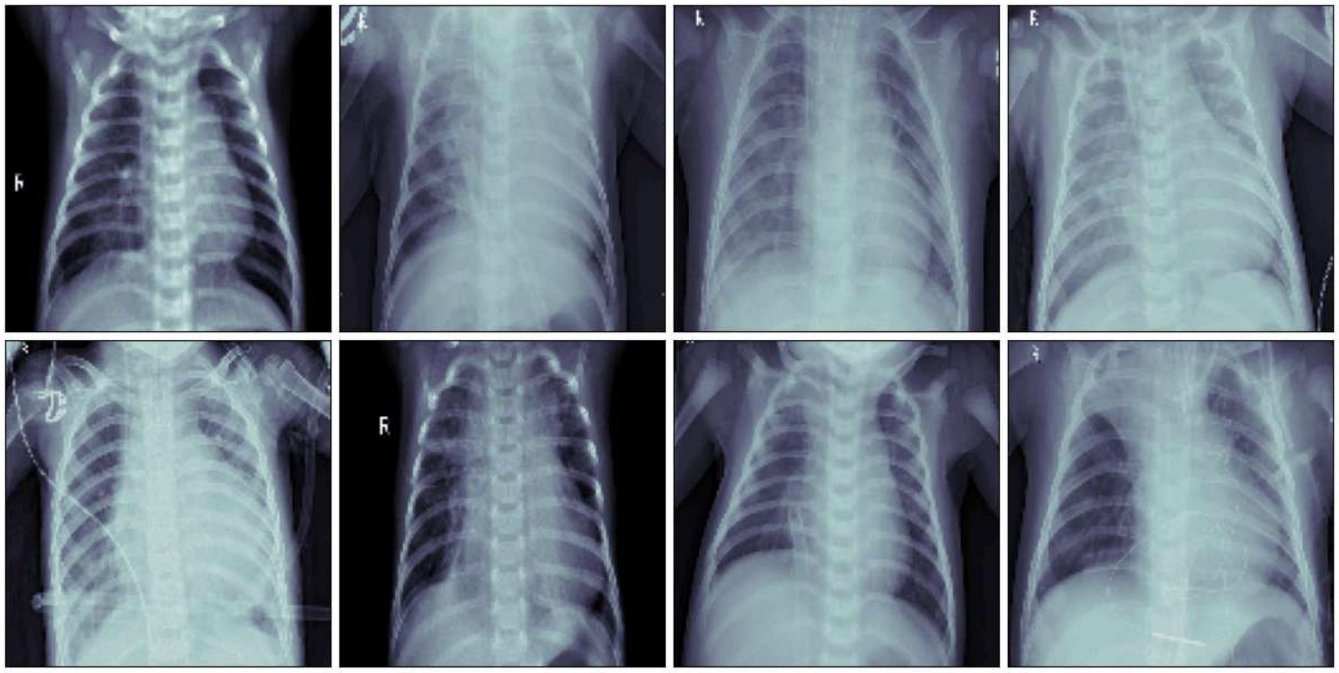
```
100%|██████████| 2/2 [00:00<00:00, 252.11it/s]
```



Bacterial Pneumonia

```
n_row = 2
n_col = 4

fig, ax = plt.subplots(n_row, n_col, figsize = (n_col*3, n_row*3), constrained_layout = True

for row in tqdm(range(n_row)):

    for col in range(n_col):

        ax[row][col].imshow(test_bacterial[row*n_col + col,:,:,0], cmap = 'bone')
        ax[row][col].set_xticks([])
        ax[row][col].set_yticks([])
```

```
100%|██████████| 2/2 [00:00<00:00, 313.29it/s]
```



Class Imbalance

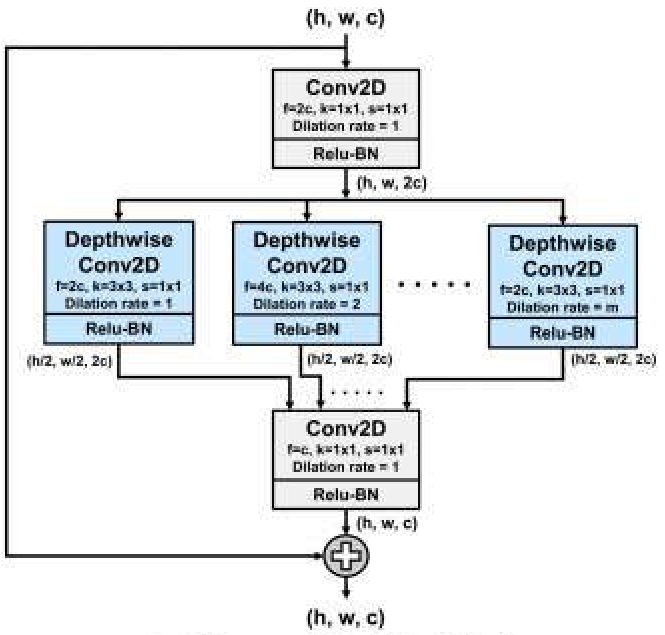
```
# from sklearn.utils import class_weight

# class_weights = class_weight.compute_class_weight('balanced',
#                                                    np.unique(train_label),
#                                                    train_label)
# class_weights
```

[CovXNet]

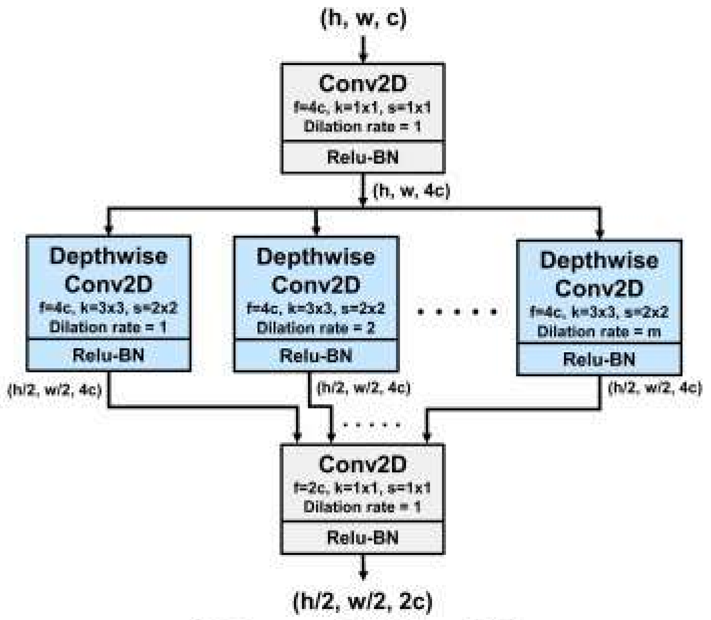
Title: CovXNet: A multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization

Code: [here]()

Residual & Shifter Unit:

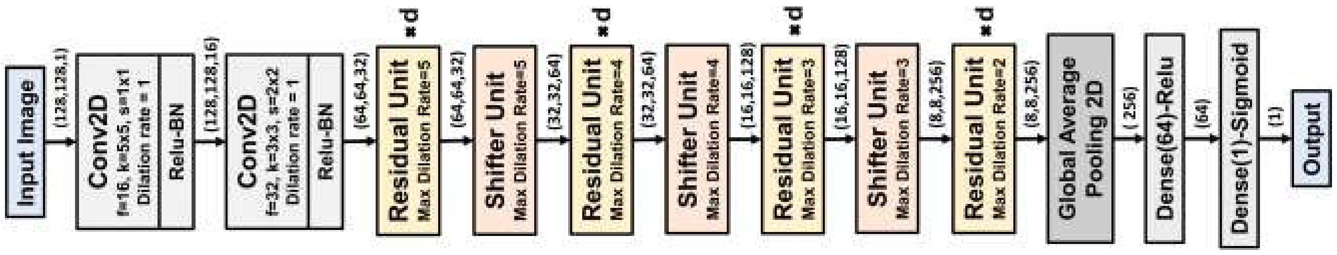


(a) Proposed Residual Unit



(b) Proposed Shifter Unit

Model:



CovXNet128

```
!pip install -q covxnet --no-deps
```

```
from covxnet import CovXNet128

model = CovXNet128(input_shape = (128, 128, 1), num_classes = 3)
model.compile(loss='categorical_crossentropy',
              optimizer = 'adam',
              metrics = ['accuracy'])
```

```
# model.summary()
# plot_model(model, show_shapes=True)
```

One Hot Encoding the labels

```
from tensorflow.keras.utils import to_categorical
train_label = to_categorical(train_label, num_classes= 3)
test_label  = to_categorical(test_label, num_classes = 3)
```

ImageDataGenerator

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale = 1/255,
                                  width_shift_range = 0.1,
                                  height_shift_range = 0.1,
                                  fill_mode = 'constant',
                                  zoom_range = 0.2,
                                  rotation_range = 30)

val_datagen = ImageDataGenerator(rescale = 1/255)

train_ds = train_datagen.flow(train_data,
                                      train_label,
                                      batch_size = 16,
                                      shuffle = True)

val_ds = val_datagen.flow(test_data,
                                 test_label,
                                 batch_size = 16,
                                 shuffle = False)
```

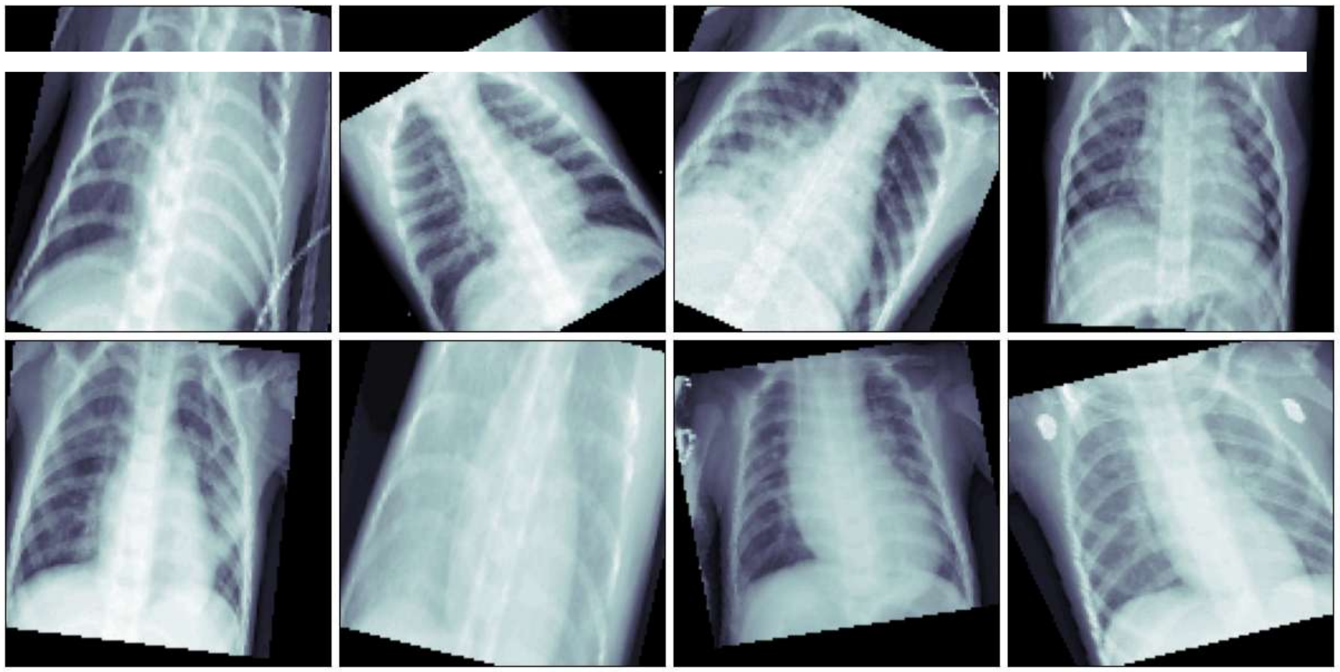
Vizualization After Augmentation

```
#images, labels = train_ds.next()
images, labels = next(train_ds)
```

```
n_row = 2
n_col = 4
```

```
fig, ax = plt.subplots(n_row, n_col, figsize = (n_col*3, n_row*3), constrained_layout = True

for row in tqdm(range(n_row)):

    for col in range(n_col):

        ax[row][col].imshow(images[row*n_col + col,:,:,0], cmap = 'bone')
        ax[row][col].set_xticks([])
        ax[row][col].set_yticks([])
```

```
100%|██████████| 2/2 [00:00<00:00, 242.25it/s]
```



Callback

```
from tensorflow.keras.callbacks import ModelCheckpoint, CSVLogger
from tensorflow.keras.models import load_model

def get_callbacks():

    filepath = 'covxnet128.h5'
    callback1 = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=0,
                                save_best_only=True, mode='max')
    callback2 = CSVLogger('covxnet128_log.csv')

    return [callback1 ,callback2]
```

Training

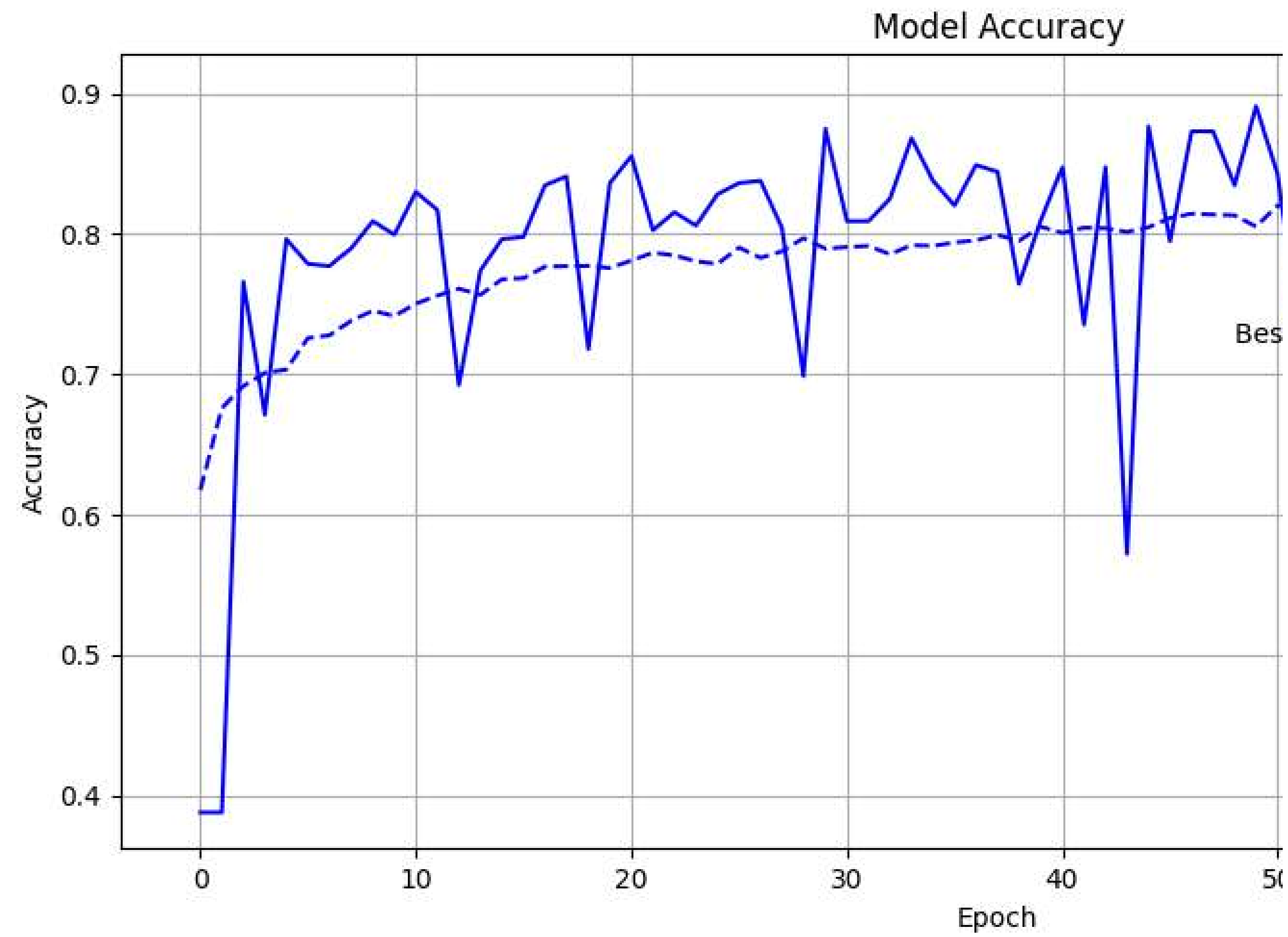
```
history = model.fit(train_ds,
                    validation_data=val_ds,
                    epochs = 75,
                    callbacks = get_callbacks(),
                    verbose = 1
                    )
```

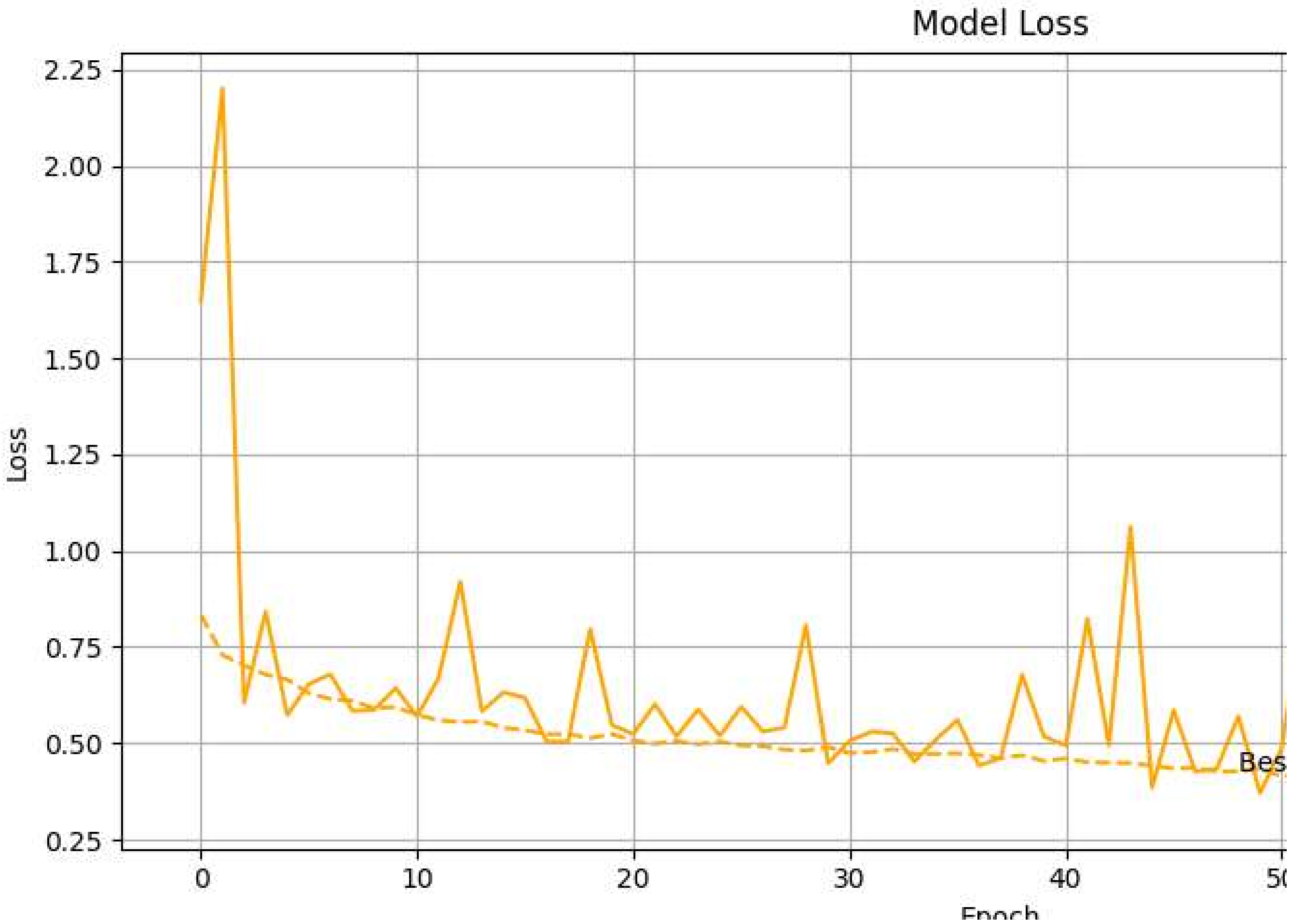
```
Epoch 66/75
326/326 ━━━━━━━━━━━━━━━━━━━━ 78s 238ms/step - accuracy: 0.8443 - loss: 0.3544 - val_a
Epoch 67/75
326/326 ━━━━━━━━━━━━━━━━━━━━ 78s 238ms/step - accuracy: 0.8589 - loss: 0.3434 - val_a
Epoch 68/75
326/326 ━━━━━━━━━━━━━━━━━━━━ 78s 239ms/step - accuracy: 0.8396 - loss: 0.3673 - val_a
Epoch 69/75
326/326 ━━━━━━━━━━━━━━━━━━━━ 78s 240ms/step - accuracy: 0.8558 - loss: 0.3432 - val_a
Epoch 70/75
326/326 ━━━━━━━━━━━━━━━━━━━━ 82s 240ms/step - accuracy: 0.8454 - loss: 0.3440 - val_a
Epoch 71/75
326/326 ━━━━━━━━━━━━━━━━━━━━ 78s 239ms/step - accuracy: 0.8613 - loss: 0.3329 - val_a
Epoch 72/75
326/326 ━━━━━━━━━━━━━━━━━━━━ 78s 239ms/step - accuracy: 0.8619 - loss: 0.3231 - val_a
Epoch 73/75
326/326 ━━━━━━━━━━━━━━━━━━━━ 78s 239ms/step - accuracy: 0.8531 - loss: 0.3445 - val_a
Epoch 74/75
326/326 ━━━━━━━━━━━━━━━━━━━━ 82s 239ms/step - accuracy: 0.8537 - loss: 0.3387 - val_a
Epoch 75/75
326/326 ━━━━━━━━━━━━━━━━━━━━ 78s 240ms/step - accuracy: 0.8598 - loss: 0.3290 - val_a
```

Plotting History

```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(2, 1, figsize=(10, 10))

# Plot training & validation accuracy values
axs[0].plot(history.history['accuracy'], label='Train Accuracy', color='blue', linestyle='da
axs[0].plot(history.history['val_accuracy'], label='Test Accuracy', color='blue')
axs[0].set_title('Model Accuracy')
axs[0].set_xlabel('Epoch')
axs[0].set_ylabel('Accuracy')
axs[0].legend(loc='lower right')
axs[0].grid(True)

max_val_acc = np.max(history.history['val_accuracy'])
max_val_acc_epoch = np.argmax(history.history['val_accuracy'])
axs[0].annotate(f'Best Acc: {max_val_acc:.2f}',
                xy=(max_val_acc_epoch, max_val_acc),
                xytext=(max_val_acc_epoch*0.8, max_val_acc*0.8),
                arrowprops=dict(facecolor='red', shrink=0.05))

# Plot training & validation loss values
axs[1].plot(history.history['loss'], label='Train Loss', color='orange', linestyle='dashed')
axs[1].plot(history.history['val_loss'], label='Test Loss', color='orange')
axs[1].set_title('Model Loss')
axs[1].set_xlabel('Epoch')
axs[1].set_ylabel('Loss')
```

```
axs[1].legend(loc='upper right')
axs[1].grid(True)

min_val_loss = history.history['val_loss'][max_val_acc_epoch]
min_val_loss_epoch = max_val_acc_epoch
axs[1].annotate(f'Best Loss: {min_val_loss:.2f}',
                xy=(min_val_loss_epoch, min_val_loss),
                xytext=(min_val_loss_epoch*0.8, min_val_loss*1.3),
                arrowprops=dict(facecolor='red', shrink=0.05))

fig.tight_layout()
plt.show()
```


Model Accuracy
Accuracy
Epoch
Bes
0.9
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50


Model Loss
Loss
Epoch
Bes
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
10
20
30
40
50