cnn-model

April 2, 2025

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
 ↪installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/
 ↪docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
 ↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that␣
 ↪gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved␣
 ↪outside of the current session
```

```
/kaggle/input/chest-xray-pneumonia/chest_xray/chest_xray/.DS_Store
/kaggle/input/chest-xray-pneumonia/chest_xray/chest_xray/val/.DS_Store
/kaggle/input/chest-xray-
pneumonia/chest_xray/chest_xray/val/PNEUMONIA/person1947_bacteria_4876.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/chest_xray/val/PNEUMONIA/person1946_bacteria_4875.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/chest_xray/val/PNEUMONIA/person1952_bacteria_4883.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/chest_xray/val/PNEUMONIA/person1954_bacteria_4886.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/chest_xray/val/PNEUMONIA/person1951_bacteria_4882.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/chest_xray/val/PNEUMONIA/person1946_bacteria_4874.jpeg
/kaggle/input/chest-xray-
```

```
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-0463-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0671-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-1385-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-0580-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0234-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-1266-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-1338-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0691-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0539-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0650-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-0657-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-1288-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0193-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-0880-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/IM-0545-0001-0002.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-0911-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0664-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-0596-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/IM-0656-0001-0002.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0125-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/IM-0435-0001-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-1197-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0440-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-0885-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0516-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/IM-0523-0001-0003.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0205-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0552-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0277-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-0522-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0354-0001.jpeg
```

```
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-1160-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-0389-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0162-0001.jpeg
/kaggle/input/chest-xray-
pneumonia/chest_xray/train/NORMAL/NORMAL2-IM-1247-0001.jpeg
/kaggle/input/chest-xray-pneumonia/chest_xray/train/NORMAL/IM-0219-0001.jpeg
```

[4]:
```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
  ↪Dropout, BatchNormalization
import os
```

[5]:
```python
dataset_path = "/kaggle/input/chest-xray-pneumonia/chest_xray/"  # Update with␣
  ↪actual dataset path
train_dir = os.path.join(dataset_path, "train")
val_dir = os.path.join(dataset_path, "val")
test_dir = os.path.join(dataset_path, "test")
```

[21]:
```python
img_size = (150, 150)
batch_size = 64
```

[22]:
```python
train_datagen = ImageDataGenerator(
    rescale=1.0/255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    brightness_range=[0.8, 1.2],
    contrast_range=[0.8, 1.2]
)

val_test_datagen = ImageDataGenerator(rescale=1.0/255)
```

[23]:
```python
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

```
val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary'
)
```

```
Found 5216 images belonging to 2 classes.
Found 16 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

[25]:
```
# CNN Model (Deeper Architecture)
model = keras.Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(150,150,3)),
    BatchNormalization(),
    MaxPooling2D(2,2),

    Conv2D(64, (3,3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(2,2),

    Conv2D(128, (3,3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(2,2),

    Conv2D(256, (3,3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(2,2),

    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),  # Prevent overfitting
    Dense(128, activation='relu'),
    Dropout(0.3),  # Prevent overfitting
    Dense(1, activation='sigmoid')  # Binary classification
])
```

[26]:
```
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

[27]:
```
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ReduceLROnPlateau
```

[28]:
```
early_stopping = EarlyStopping(monitor='val_loss', patience=5,␣
  ↪restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3,␣
  ↪min_lr=1e-6)
```

[29]:
```
# Train Model
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=20,
    callbacks=[early_stopping, reduce_lr]
)
```

```
Epoch 1/20
82/82                 75s 747ms/step -
accuracy: 0.7663 - loss: 0.9861 - val_accuracy: 0.5000 - val_loss: 15.6292 -
learning_rate: 0.0010
Epoch 2/20
82/82                 63s 706ms/step -
accuracy: 0.8498 - loss: 0.3694 - val_accuracy: 0.5000 - val_loss: 19.3237 -
learning_rate: 0.0010
Epoch 3/20
82/82                 63s 706ms/step -
accuracy: 0.8726 - loss: 0.3318 - val_accuracy: 0.5000 - val_loss: 25.1221 -
learning_rate: 0.0010
Epoch 4/20
82/82                 64s 707ms/step -
accuracy: 0.8860 - loss: 0.3012 - val_accuracy: 0.5000 - val_loss: 22.1412 -
learning_rate: 0.0010
Epoch 5/20
82/82                 64s 714ms/step -
accuracy: 0.9061 - loss: 0.2286 - val_accuracy: 0.5000 - val_loss: 17.6029 -
learning_rate: 2.0000e-04
Epoch 6/20
82/82                 64s 708ms/step -
accuracy: 0.9117 - loss: 0.2082 - val_accuracy: 0.5000 - val_loss: 12.1762 -
learning_rate: 2.0000e-04
Epoch 7/20
82/82                 64s 713ms/step -
accuracy: 0.9238 - loss: 0.1999 - val_accuracy: 0.5000 - val_loss: 37.2418 -
learning_rate: 2.0000e-04
Epoch 8/20
82/82                 63s 699ms/step -
accuracy: 0.9160 - loss: 0.2035 - val_accuracy: 0.5000 - val_loss: 5.4114 -
learning_rate: 2.0000e-04
Epoch 9/20
82/82                 65s 727ms/step -
```

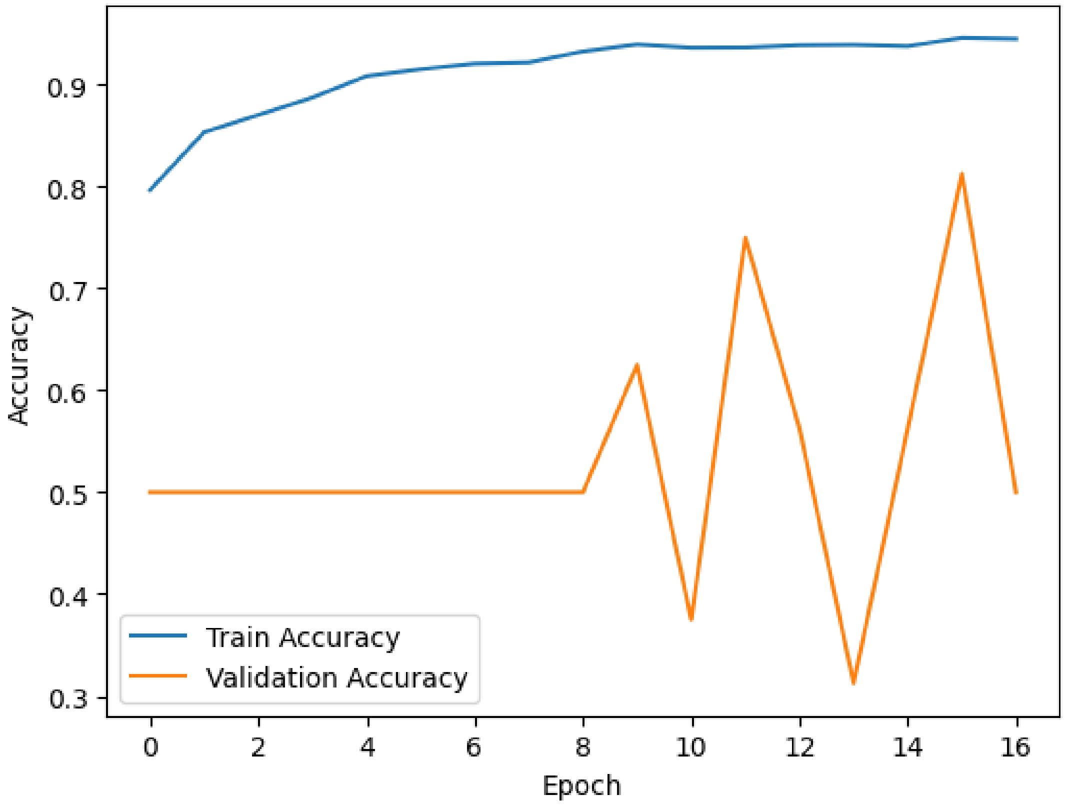
```
accuracy: 0.9323 - loss: 0.1953 - val_accuracy: 0.5000 - val_loss: 1.4358 -
learning_rate: 2.0000e-04
Epoch 10/20
82/82                64s 715ms/step -
accuracy: 0.9365 - loss: 0.1721 - val_accuracy: 0.6250 - val_loss: 0.6757 -
learning_rate: 2.0000e-04
Epoch 11/20
82/82                64s 707ms/step -
accuracy: 0.9405 - loss: 0.1559 - val_accuracy: 0.3750 - val_loss: 0.9060 -
learning_rate: 2.0000e-04
Epoch 12/20
82/82                65s 720ms/step -
accuracy: 0.9309 - loss: 0.1804 - val_accuracy: 0.7500 - val_loss: 0.4052 -
learning_rate: 2.0000e-04
Epoch 13/20
82/82                64s 713ms/step -
accuracy: 0.9362 - loss: 0.1655 - val_accuracy: 0.5625 - val_loss: 0.7815 -
learning_rate: 2.0000e-04
Epoch 14/20
82/82                64s 712ms/step -
accuracy: 0.9451 - loss: 0.1633 - val_accuracy: 0.3125 - val_loss: 1.2365 -
learning_rate: 2.0000e-04
Epoch 15/20
82/82                66s 732ms/step -
accuracy: 0.9346 - loss: 0.1617 - val_accuracy: 0.5625 - val_loss: 2.4196 -
learning_rate: 2.0000e-04
Epoch 16/20
82/82                65s 722ms/step -
accuracy: 0.9523 - loss: 0.1369 - val_accuracy: 0.8125 - val_loss: 0.4800 -
learning_rate: 4.0000e-05
Epoch 17/20
82/82                65s 727ms/step -
accuracy: 0.9412 - loss: 0.1460 - val_accuracy: 0.5000 - val_loss: 0.7982 -
learning_rate: 4.0000e-05
```

[30]:
```
#### Evaluate Model
test_loss, test_acc = model.evaluate(test_generator)
print(f"Test accuracy: {test_acc:.4f}")

# Plot Accuracy
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
10/10                8s 690ms/step -
```

```
accuracy: 0.8954 - loss: 0.3020
Test accuracy: 0.8782
```


Accuracy
Train Accuracy
Validation Accuracy
Epoch