baseline

April 24, 2025

[1]:
```
import os
import itertools
import random
import data_pipeline as pipeline
import pandas as pd
import numpy as np
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
from tensorflow.keras.metrics import BinaryAccuracy, Precision, Recall
from tensorflow.keras.callbacks import EarlyStopping
```

[2]:
```
PROCESSED_DIR = "../data/processed"
IMG_HEIGHT = IMG_WIDTH = 224
NUM_CHANNELS = 3
NUM_CLASSES = 2
EPOCHS = 30
BATCH_SIZE = 128
```

[3]:
```
SEED = 42
np.random.seed(SEED)
tf.random.set_seed(SEED)
random.seed(SEED)
```

[4]:
```
metrics = [
    BinaryAccuracy(name="accuracy"),
    Precision(name="precision"),
    Recall(name="recall"),
]
```

[5]:
```
train_dir = os.path.join(PROCESSED_DIR, "train")
val_dir = os.path.join(PROCESSED_DIR, "val")
test_dir = os.path.join(PROCESSED_DIR, "test")

all_paths = pipeline.get_image_paths(PROCESSED_DIR)
train_paths = [path for path in all_paths if "/train/" in path]
mean, std = pipeline.calc_mean_std(train_paths)
```

```
# use generators from data_pipeline for training, validation, and testing
print("loading train/val/test generators from data_pipeline")
train_data_gen, val_data_gen, test_data_gen, test_data_gen_raw = pipeline.
 ↪load_data(
    train_dir, val_dir, test_dir, mean, std
)
```

```
loading train/val/test generators from data_pipeline
creating train generator
Found 1600 images belonging to 2 classes.
creating validation generator
Found 400 images belonging to 2 classes.
creating test generator (normalized)
Found 200 images belonging to 2 classes.
creating test generator (raw)
Found 200 images belonging to 2 classes.
```

[6]:
```
image_size = (IMG_HEIGHT, IMG_WIDTH)
input_shape = (IMG_HEIGHT, IMG_WIDTH, NUM_CHANNELS)
```

[7]:
```
class_names = list(train_data_gen.class_indices.keys())
print(f"class names found: {class_names}")
```

```
class names found: ['NORMAL', 'COVID']
```

[8]:
```
def build_model(input_shape):
    """
    build keras sequential model

    params
    ------
    input_shape: tuple
        shape of input images (height, width, channels)

    returns
    -------
    model: tf.keras.Model
        compiled keras model
    """
    model = tf.keras.Sequential(
        [
            tf.keras.layers.Input(shape=input_shape),
            # convolutional
            tf.keras.layers.Conv2D(32, (3, 3), activation="relu",␣
 ↪padding="same"),
            tf.keras.layers.MaxPooling2D((2, 2)),
```

```
            tf.keras.layers.Conv2D(64, (3, 3), activation="relu",␣
↪padding="same"),
            tf.keras.layers.MaxPooling2D((2, 2)),
            tf.keras.layers.Conv2D(128, (3, 3), activation="relu",␣
↪padding="same"),
            tf.keras.layers.MaxPooling2D((2, 2)),
            # fully connected
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(128, activation="relu"),
            tf.keras.layers.Dropout(0.3),  # random value
            tf.keras.layers.Dense(64, activation="relu"),
            tf.keras.layers.Dropout(0.3),  # random value
            tf.keras.layers.Dense(1, activation="sigmoid"),
        ]
    )
    return model
```

[9]:
```
def train_model(model, train_data, val_data, epochs=EPOCHS):
    # compile the model
    model.compile(
        optimizer="adam",
        loss="binary_crossentropy",
        metrics=metrics,
    )
    # train the model
    history = model.fit(train_data, epochs=epochs, validation_data=val_data,␣
↪verbose=2)
    return history
```

[10]:
```
# build model
model = build_model(input_shape)

# print model summary
print("model architecture:")
model.summary()
```

```
model architecture:
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
max_pooling2d (MaxPooling2D)	(None, 112, 112, 32)	0
conv2d_1 (Conv2D)	(None, 112, 112, 64)	18,496

max_pooling2d_1 (MaxPooling2D)	(None, 56, 56, 64)	0
conv2d_2 (Conv2D)	(None, 56, 56, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 28, 28, 128)	0
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 128)	12,845,184
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 1)	65

Total params: 12,946,753 (49.39 MB)

Trainable params: 12,946,753 (49.39 MB)

Non-trainable params: 0 (0.00 B)

[11]:
```
# train the model
history = train_model(model, train_data_gen, val_data_gen, EPOCHS)
```

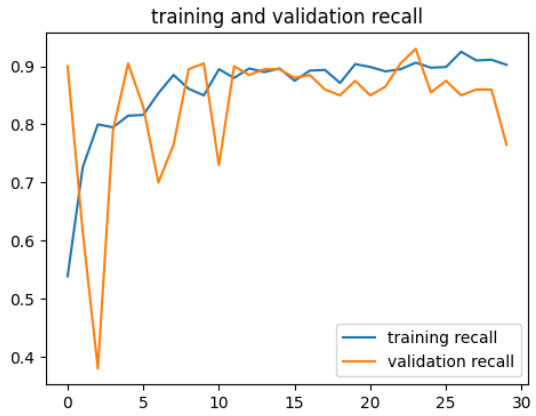
```
/opt/anaconda3/envs/ml-2025/lib/python3.12/site-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()
Epoch 1/30
13/13 - 27s - 2s/step - accuracy: 0.6137 - loss: 0.7341 - precision: 0.6338 -
recall: 0.5387 - val_accuracy: 0.6000 - val_loss: 0.7182 - val_precision: 0.5625
- val_recall: 0.9000
Epoch 2/30
13/13 - 28s - 2s/step - accuracy: 0.7206 - loss: 0.5636 - precision: 0.7182 -
recall: 0.7262 - val_accuracy: 0.6225 - val_loss: 0.6532 - val_precision: 0.6244
- val_recall: 0.6150
Epoch 3/30
```

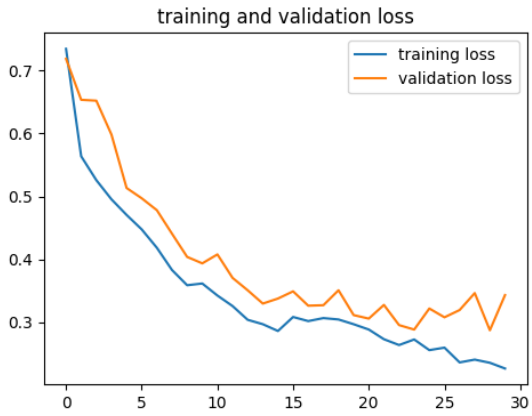
```
13/13 - 25s - 2s/step - accuracy: 0.7494 - loss: 0.5258 - precision: 0.7264 -
recall: 0.8000 - val_accuracy: 0.6075 - val_loss: 0.6518 - val_precision: 0.6972
- val_recall: 0.3800
Epoch 4/30
13/13 - 26s - 2s/step - accuracy: 0.7706 - loss: 0.4954 - precision: 0.7580 -
recall: 0.7950 - val_accuracy: 0.6900 - val_loss: 0.5980 - val_precision: 0.6583
- val_recall: 0.7900
Epoch 5/30
13/13 - 26s - 2s/step - accuracy: 0.7812 - loss: 0.4706 - precision: 0.7635 -
recall: 0.8150 - val_accuracy: 0.7775 - val_loss: 0.5133 - val_precision: 0.7211
- val_recall: 0.9050
Epoch 6/30
13/13 - 29s - 2s/step - accuracy: 0.7962 - loss: 0.4476 - precision: 0.7849 -
recall: 0.8163 - val_accuracy: 0.7550 - val_loss: 0.4970 - val_precision: 0.7217
- val_recall: 0.8300
Epoch 7/30
13/13 - 27s - 2s/step - accuracy: 0.8050 - loss: 0.4185 - precision: 0.7779 -
recall: 0.8537 - val_accuracy: 0.7775 - val_loss: 0.4781 - val_precision: 0.8284
- val_recall: 0.7000
Epoch 8/30
13/13 - 25s - 2s/step - accuracy: 0.8263 - loss: 0.3832 - precision: 0.7919 -
recall: 0.8850 - val_accuracy: 0.7875 - val_loss: 0.4408 - val_precision: 0.8010
- val_recall: 0.7650
Epoch 9/30
13/13 - 23s - 2s/step - accuracy: 0.8425 - loss: 0.3590 - precision: 0.8301 -
recall: 0.8612 - val_accuracy: 0.8275 - val_loss: 0.4040 - val_precision: 0.7885
- val_recall: 0.8950
Epoch 10/30
13/13 - 23s - 2s/step - accuracy: 0.8288 - loss: 0.3618 - precision: 0.8153 -
recall: 0.8500 - val_accuracy: 0.8275 - val_loss: 0.3938 - val_precision: 0.7835
- val_recall: 0.9050
Epoch 11/30
13/13 - 23s - 2s/step - accuracy: 0.8431 - loss: 0.3427 - precision: 0.8109 -
recall: 0.8950 - val_accuracy: 0.8075 - val_loss: 0.4080 - val_precision: 0.8639
- val_recall: 0.7300
Epoch 12/30
13/13 - 23s - 2s/step - accuracy: 0.8562 - loss: 0.3260 - precision: 0.8401 -
recall: 0.8800 - val_accuracy: 0.8275 - val_loss: 0.3707 - val_precision: 0.7860
- val_recall: 0.9000
Epoch 13/30
13/13 - 23s - 2s/step - accuracy: 0.8712 - loss: 0.3042 - precision: 0.8536 -
recall: 0.8963 - val_accuracy: 0.8775 - val_loss: 0.3510 - val_precision: 0.8719
- val_recall: 0.8850
Epoch 14/30
13/13 - 24s - 2s/step - accuracy: 0.8744 - loss: 0.2972 - precision: 0.8630 -
recall: 0.8900 - val_accuracy: 0.8700 - val_loss: 0.3299 - val_precision: 0.8524
- val_recall: 0.8950
Epoch 15/30
```

```
13/13 - 24s - 2s/step - accuracy: 0.8775 - loss: 0.2864 - precision: 0.8639 -
recall: 0.8963 - val_accuracy: 0.8600 - val_loss: 0.3379 - val_precision: 0.8364
- val_recall: 0.8950
Epoch 16/30
13/13 - 23s - 2s/step - accuracy: 0.8712 - loss: 0.3087 - precision: 0.8685 -
recall: 0.8750 - val_accuracy: 0.8700 - val_loss: 0.3494 - val_precision: 0.8627
- val_recall: 0.8800
Epoch 17/30
13/13 - 23s - 2s/step - accuracy: 0.8744 - loss: 0.3022 - precision: 0.8613 -
recall: 0.8925 - val_accuracy: 0.8800 - val_loss: 0.3266 - val_precision: 0.8762
- val_recall: 0.8850
Epoch 18/30
13/13 - 23s - 2s/step - accuracy: 0.8731 - loss: 0.3069 - precision: 0.8583 -
recall: 0.8938 - val_accuracy: 0.8725 - val_loss: 0.3273 - val_precision: 0.8821
- val_recall: 0.8600
Epoch 19/30
13/13 - 23s - 2s/step - accuracy: 0.8694 - loss: 0.3047 - precision: 0.8680 -
recall: 0.8712 - val_accuracy: 0.8650 - val_loss: 0.3511 - val_precision: 0.8763
- val_recall: 0.8500
Epoch 20/30
13/13 - 23s - 2s/step - accuracy: 0.8769 - loss: 0.2972 - precision: 0.8577 -
recall: 0.9038 - val_accuracy: 0.8800 - val_loss: 0.3117 - val_precision: 0.8838
- val_recall: 0.8750
Epoch 21/30
13/13 - 23s - 2s/step - accuracy: 0.8719 - loss: 0.2888 - precision: 0.8529 -
recall: 0.8988 - val_accuracy: 0.8800 - val_loss: 0.3061 - val_precision: 0.9043
- val_recall: 0.8500
Epoch 22/30
13/13 - 23s - 2s/step - accuracy: 0.8744 - loss: 0.2734 - precision: 0.8622 -
recall: 0.8913 - val_accuracy: 0.8750 - val_loss: 0.3278 - val_precision: 0.8827
- val_recall: 0.8650
Epoch 23/30
13/13 - 23s - 2s/step - accuracy: 0.8925 - loss: 0.2642 - precision: 0.8905 -
recall: 0.8950 - val_accuracy: 0.9000 - val_loss: 0.2958 - val_precision: 0.8960
- val_recall: 0.9050
Epoch 24/30
13/13 - 23s - 2s/step - accuracy: 0.8906 - loss: 0.2730 - precision: 0.8788 -
recall: 0.9062 - val_accuracy: 0.8900 - val_loss: 0.2887 - val_precision: 0.8611
- val_recall: 0.9300
Epoch 25/30
13/13 - 23s - 2s/step - accuracy: 0.8888 - loss: 0.2561 - precision: 0.8821 -
recall: 0.8975 - val_accuracy: 0.8750 - val_loss: 0.3221 - val_precision: 0.8906
- val_recall: 0.8550
Epoch 26/30
13/13 - 23s - 2s/step - accuracy: 0.8938 - loss: 0.2600 - precision: 0.8899 -
recall: 0.8988 - val_accuracy: 0.8875 - val_loss: 0.3082 - val_precision: 0.8974
- val_recall: 0.8750
Epoch 27/30
```

```
13/13 - 23s - 2s/step - accuracy: 0.9006 - loss: 0.2366 - precision: 0.8820 -
recall: 0.9250 - val_accuracy: 0.8775 - val_loss: 0.3197 - val_precision: 0.8995
- val_recall: 0.8500
Epoch 28/30
13/13 - 23s - 2s/step - accuracy: 0.9044 - loss: 0.2411 - precision: 0.8999 -
recall: 0.9100 - val_accuracy: 0.8700 - val_loss: 0.3466 - val_precision: 0.8776
- val_recall: 0.8600
Epoch 29/30
13/13 - 23s - 2s/step - accuracy: 0.9000 - loss: 0.2360 - precision: 0.8912 -
recall: 0.9112 - val_accuracy: 0.8850 - val_loss: 0.2876 - val_precision: 0.9053
- val_recall: 0.8600
Epoch 30/30
13/13 - 23s - 2s/step - accuracy: 0.8975 - loss: 0.2269 - precision: 0.8936 -
recall: 0.9025 - val_accuracy: 0.8550 - val_loss: 0.3434 - val_precision: 0.9329
- val_recall: 0.7650
```

[12]:
```python
# plot training history
recall = history.history["recall"]
val_recall = history.history["val_recall"]
loss = history.history["loss"]
val_loss = history.history["val_loss"]

epochs_range = range(EPOCHS)

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, recall, label="training recall")
plt.plot(epochs_range, val_recall, label="validation recall")
plt.legend(loc="lower right")
plt.title("training and validation recall")

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label="training loss")
plt.plot(epochs_range, val_loss, label="validation loss")
plt.legend(loc="upper right")
plt.title("training and validation loss")
plt.show()
```





[13]:
```
# evaluate the model on the validation set after training
print("\nevaluating model on validation data after training")
results = model.evaluate(val_data_gen, verbose=1)
print(f"final validation loss: {results[0]}")
print(f"final validation accuracy: {results[1]}")
print(f"final validation precision: {results[2]}")
print(f"final validation recall: {results[3]}")
```

```
evaluating model on validation data after training
4/4             2s 355ms/step -
accuracy: 0.8876 - loss: 0.2720 - precision: 0.7192 - recall: 0.6218
final validation loss: 0.3434081971645355
final validation accuracy: 0.8550000190734863
final validation precision: 0.9329268336296082
final validation recall: 0.7649999856948853
```

1 with hyperparam tuning

[14]:
```
def build_model(input_shape, filters_conv1, units_dense1, dropout_rate):
    """
    build keras sequential model

    params
    ------
    input_shape: tuple
        shape of input images (height, width, channels)
    filters_conv1: int
        number of filters in the first convolutional layer
    units_dense1: int
        number of units in the first dense layer
    dropout_rate: float
```

```
        dropout rate for dropout layers

    returns
    -------
    model: tf.keras.Model
        keras model (not compiled)
    """
    model = tf.keras.Sequential(
        [
            tf.keras.layers.Input(shape=input_shape),
            # convolutional
            tf.keras.layers.Conv2D(
                filters_conv1, (3, 3), activation="relu", padding="same"
            ),
            tf.keras.layers.MaxPooling2D((2, 2)),
            tf.keras.layers.Conv2D(64, (3, 3), activation="relu",␣
↪padding="same"),
            tf.keras.layers.MaxPooling2D((2, 2)),
            tf.keras.layers.Conv2D(128, (3, 3), activation="relu",␣
↪padding="same"),
            tf.keras.layers.MaxPooling2D((2, 2)),
            # fully connected
            tf.keras.layers.Flatten(),
            tf.keras.layers.Dense(units_dense1, activation="relu"),
            tf.keras.layers.Dropout(dropout_rate),
            tf.keras.layers.Dense(64, activation="relu"),
            tf.keras.layers.Dropout(dropout_rate),
            tf.keras.layers.Dense(1, activation="sigmoid"),
        ]
    )
    return model
```

[15]:
```
# define search space
learning_rates = [1e-3, 1e-4, 1e-5]
filters_conv1_list = [16, 32, 64]
units_dense1_list = [64, 128, 256]
dropout_rates = [0.2, 0.3, 0.4, 0.5]
```

[16]:
```
# create all possible combinations (full grid)
param_grid = list(
    itertools.product(
        learning_rates, filters_conv1_list, units_dense1_list, dropout_rates
    )
)
```

[17]:
```
# define number of combinations to randomly sample
num_combinations_to_test = 5
```

```
# randomly sample combinations
sampled_params = random.sample(param_grid, num_combinations_to_test)
print(
    f"randomly sampling {num_combinations_to_test} combinations from␣
 ↪{len(param_grid)} total."
)
```

```
randomly sampling 5 combinations from 108 total.
```

[18]:
```
# store results
results_list = []
```

[19]:
```
# early stopping callback
early_stopping = EarlyStopping(
    monitor="val_loss", patience=5, restore_best_weights=True, verbose=1
)
```

[20]:
```
print("starting hyperparameter tuning")

for lr, filters1, units1, dr in sampled_params:
    print(
        f"testing: lr={lr}, filters_conv1={filters1}, units_dense1={units1},␣
 ↪dropout_rate={dr}"
    )

    # build model
    input_shape = (IMG_HEIGHT, IMG_WIDTH, NUM_CHANNELS)
    model = build_model(
        input_shape,
        filters_conv1=filters1,
        units_dense1=units1,
        dropout_rate=dr,
    )

    # compile model
    optimizer = tf.keras.optimizers.Adam(learning_rate=lr)
    model.compile(optimizer=optimizer, loss="binary_crossentropy",␣
 ↪metrics=metrics)

    # train the model
    history = model.fit(
        train_data_gen,
        epochs=EPOCHS,
        validation_data=val_data_gen,
        callbacks=[early_stopping],
        verbose=2,
```

```
    )

    # evaluate the model on the validation set using best weights from early␣
 ↪stopping
    print("evaluating best model from this run")
    eval_results = model.evaluate(val_data_gen, verbose=1)

    run_results = {
        "learning_rate": lr,
        "filters_conv1": filters1,
        "units_dense1": units1,
        "dropout_rate": dr,
        "val_loss": eval_results[0],
        "val_accuracy": eval_results[1],
        "val_precision": eval_results[2],
        "val_recall": eval_results[3],
        "epochs_trained": len(history.epoch),
    }
    results_list.append(run_results)

print("hyperparameter tuning finished.")
```

```
starting hyperparameter tuning
testing: lr=0.001, filters_conv1=16, units_dense1=64, dropout_rate=0.2
Epoch 1/30
13/13 - 19s - 1s/step - accuracy: 0.7040 - loss: 0.6099 - precision: 0.7569 -
recall: 0.6010 - val_accuracy: 0.6275 - val_loss: 0.6790 - val_precision: 0.5889
- val_recall: 0.8450
Epoch 2/30
13/13 - 18s - 1s/step - accuracy: 0.7375 - loss: 0.5224 - precision: 0.7387 -
recall: 0.7350 - val_accuracy: 0.7150 - val_loss: 0.5928 - val_precision: 0.6807
- val_recall: 0.8100
Epoch 3/30
13/13 - 18s - 1s/step - accuracy: 0.7594 - loss: 0.4923 - precision: 0.7427 -
recall: 0.7937 - val_accuracy: 0.7275 - val_loss: 0.5966 - val_precision: 0.7358
- val_recall: 0.7100
Epoch 4/30
13/13 - 17s - 1s/step - accuracy: 0.8000 - loss: 0.4405 - precision: 0.7850 -
recall: 0.8263 - val_accuracy: 0.7575 - val_loss: 0.5186 - val_precision: 0.7725
- val_recall: 0.7300
Epoch 5/30
13/13 - 18s - 1s/step - accuracy: 0.8050 - loss: 0.4425 - precision: 0.7805 -
recall: 0.8487 - val_accuracy: 0.6600 - val_loss: 0.5973 - val_precision: 0.8333
- val_recall: 0.4000
Epoch 6/30
13/13 - 17s - 1s/step - accuracy: 0.8175 - loss: 0.4151 - precision: 0.7953 -
recall: 0.8550 - val_accuracy: 0.7575 - val_loss: 0.4948 - val_precision: 0.6914
- val_recall: 0.9300
```

```
Epoch 7/30
13/13 - 18s - 1s/step - accuracy: 0.8319 - loss: 0.3811 - precision: 0.8105 -
recall: 0.8662 - val_accuracy: 0.7825 - val_loss: 0.4821 - val_precision: 0.7198
- val_recall: 0.9250
Epoch 8/30
13/13 - 18s - 1s/step - accuracy: 0.8431 - loss: 0.3638 - precision: 0.8181 -
recall: 0.8825 - val_accuracy: 0.8125 - val_loss: 0.4028 - val_precision: 0.8019
- val_recall: 0.8300
Epoch 9/30
13/13 - 18s - 1s/step - accuracy: 0.8394 - loss: 0.3594 - precision: 0.8190 -
recall: 0.8712 - val_accuracy: 0.8000 - val_loss: 0.4389 - val_precision: 0.8000
- val_recall: 0.8000
Epoch 10/30
13/13 - 18s - 1s/step - accuracy: 0.8594 - loss: 0.3314 - precision: 0.8410 -
recall: 0.8863 - val_accuracy: 0.8025 - val_loss: 0.4235 - val_precision: 0.8954
- val_recall: 0.6850
Epoch 11/30
13/13 - 18s - 1s/step - accuracy: 0.8444 - loss: 0.3390 - precision: 0.8439 -
recall: 0.8450 - val_accuracy: 0.7925 - val_loss: 0.4465 - val_precision: 0.7773
- val_recall: 0.8200
Epoch 12/30
13/13 - 18s - 1s/step - accuracy: 0.8587 - loss: 0.3273 - precision: 0.8409 -
recall: 0.8850 - val_accuracy: 0.8125 - val_loss: 0.4230 - val_precision: 0.7729
- val_recall: 0.8850
Epoch 13/30
13/13 - 18s - 1s/step - accuracy: 0.8594 - loss: 0.3053 - precision: 0.8419 -
recall: 0.8850 - val_accuracy: 0.7375 - val_loss: 0.5317 - val_precision: 0.8276
- val_recall: 0.6000
Epoch 13: early stopping
Restoring model weights from the end of the best epoch: 8.
evaluating best model from this run
4/4             1s 246ms/step -
accuracy: 0.8078 - loss: 0.3984 - precision: 0.5809 - recall: 0.6519
testing: lr=0.0001, filters_conv1=16, units_dense1=128, dropout_rate=0.4
Epoch 1/30
13/13 - 19s - 1s/step - accuracy: 0.6555 - loss: 0.6638 - precision: 0.6794 -
recall: 0.5890 - val_accuracy: 0.6300 - val_loss: 0.6509 - val_precision: 0.6368
- val_recall: 0.6050
Epoch 2/30
13/13 - 18s - 1s/step - accuracy: 0.6913 - loss: 0.5836 - precision: 0.7068 -
recall: 0.6538 - val_accuracy: 0.6125 - val_loss: 0.6611 - val_precision: 0.6301
- val_recall: 0.5450
Epoch 3/30
13/13 - 18s - 1s/step - accuracy: 0.7169 - loss: 0.5537 - precision: 0.7188 -
recall: 0.7125 - val_accuracy: 0.6250 - val_loss: 0.6601 - val_precision: 0.6157
- val_recall: 0.6650
Epoch 4/30
13/13 - 18s - 1s/step - accuracy: 0.7369 - loss: 0.5473 - precision: 0.7297 -
```

```
recall: 0.7525 - val_accuracy: 0.6300 - val_loss: 0.6488 - val_precision: 0.6250
- val_recall: 0.6500
Epoch 5/30
13/13 - 18s - 1s/step - accuracy: 0.7487 - loss: 0.5285 - precision: 0.7403 -
recall: 0.7663 - val_accuracy: 0.6750 - val_loss: 0.6138 - val_precision: 0.6683
- val_recall: 0.6950
Epoch 6/30
13/13 - 18s - 1s/step - accuracy: 0.7475 - loss: 0.5288 - precision: 0.7409 -
recall: 0.7613 - val_accuracy: 0.6875 - val_loss: 0.6071 - val_precision: 0.6652
- val_recall: 0.7550
Epoch 7/30
13/13 - 18s - 1s/step - accuracy: 0.7619 - loss: 0.5119 - precision: 0.7422 -
recall: 0.8025 - val_accuracy: 0.6800 - val_loss: 0.5972 - val_precision: 0.7022
- val_recall: 0.6250
Epoch 8/30
13/13 - 18s - 1s/step - accuracy: 0.7706 - loss: 0.5020 - precision: 0.7599 -
recall: 0.7912 - val_accuracy: 0.6725 - val_loss: 0.5983 - val_precision: 0.7018
- val_recall: 0.6000
Epoch 9/30
13/13 - 18s - 1s/step - accuracy: 0.7756 - loss: 0.4908 - precision: 0.7603 -
recall: 0.8050 - val_accuracy: 0.7350 - val_loss: 0.5470 - val_precision: 0.7238
- val_recall: 0.7600
Epoch 10/30
13/13 - 18s - 1s/step - accuracy: 0.7738 - loss: 0.4827 - precision: 0.7724 -
recall: 0.7763 - val_accuracy: 0.7050 - val_loss: 0.5677 - val_precision: 0.7595
- val_recall: 0.6000
Epoch 11/30
13/13 - 18s - 1s/step - accuracy: 0.7931 - loss: 0.4689 - precision: 0.7795 -
recall: 0.8175 - val_accuracy: 0.7050 - val_loss: 0.5625 - val_precision: 0.7500
- val_recall: 0.6150
Epoch 12/30
13/13 - 18s - 1s/step - accuracy: 0.7856 - loss: 0.4704 - precision: 0.7698 -
recall: 0.8150 - val_accuracy: 0.6825 - val_loss: 0.5903 - val_precision: 0.7920
- val_recall: 0.4950
Epoch 13/30
13/13 - 18s - 1s/step - accuracy: 0.7894 - loss: 0.4552 - precision: 0.7689 -
recall: 0.8275 - val_accuracy: 0.6950 - val_loss: 0.5710 - val_precision: 0.7868
- val_recall: 0.5350
Epoch 14/30
13/13 - 18s - 1s/step - accuracy: 0.8044 - loss: 0.4358 - precision: 0.7944 -
recall: 0.8213 - val_accuracy: 0.7150 - val_loss: 0.5229 - val_precision: 0.7722
- val_recall: 0.6100
Epoch 15/30
13/13 - 18s - 1s/step - accuracy: 0.7981 - loss: 0.4419 - precision: 0.7891 -
recall: 0.8138 - val_accuracy: 0.7650 - val_loss: 0.4809 - val_precision: 0.7849
- val_recall: 0.7300
Epoch 16/30
13/13 - 18s - 1s/step - accuracy: 0.8112 - loss: 0.4299 - precision: 0.8007 -
```

```
recall: 0.8288 - val_accuracy: 0.7750 - val_loss: 0.4793 - val_precision: 0.8090
- val_recall: 0.7200
Epoch 17/30
13/13 - 18s - 1s/step - accuracy: 0.8087 - loss: 0.4212 - precision: 0.8119 -
recall: 0.8037 - val_accuracy: 0.7825 - val_loss: 0.4552 - val_precision: 0.8054
- val_recall: 0.7450
Epoch 18/30
13/13 - 18s - 1s/step - accuracy: 0.8031 - loss: 0.4284 - precision: 0.7843 -
recall: 0.8363 - val_accuracy: 0.7875 - val_loss: 0.4605 - val_precision: 0.7979
- val_recall: 0.7700
Epoch 19/30
13/13 - 18s - 1s/step - accuracy: 0.8181 - loss: 0.4062 - precision: 0.7991 -
recall: 0.8500 - val_accuracy: 0.7850 - val_loss: 0.4590 - val_precision: 0.8393
- val_recall: 0.7050
Epoch 20/30
13/13 - 18s - 1s/step - accuracy: 0.8131 - loss: 0.3989 - precision: 0.8051 -
recall: 0.8263 - val_accuracy: 0.8075 - val_loss: 0.4514 - val_precision: 0.7709
- val_recall: 0.8750
Epoch 21/30
13/13 - 18s - 1s/step - accuracy: 0.8244 - loss: 0.3950 - precision: 0.8161 -
recall: 0.8375 - val_accuracy: 0.7825 - val_loss: 0.4447 - val_precision: 0.7425
- val_recall: 0.8650
Epoch 22/30
13/13 - 18s - 1s/step - accuracy: 0.8131 - loss: 0.4110 - precision: 0.7923 -
recall: 0.8487 - val_accuracy: 0.7800 - val_loss: 0.4648 - val_precision: 0.7667
- val_recall: 0.8050
Epoch 23/30
13/13 - 18s - 1s/step - accuracy: 0.8294 - loss: 0.3818 - precision: 0.8249 -
recall: 0.8363 - val_accuracy: 0.7625 - val_loss: 0.4765 - val_precision: 0.9268
- val_recall: 0.5700
Epoch 24/30
13/13 - 18s - 1s/step - accuracy: 0.8294 - loss: 0.3761 - precision: 0.8194 -
recall: 0.8450 - val_accuracy: 0.8125 - val_loss: 0.4387 - val_precision: 0.8415
- val_recall: 0.7700
Epoch 25/30
13/13 - 18s - 1s/step - accuracy: 0.8350 - loss: 0.3689 - precision: 0.8198 -
recall: 0.8587 - val_accuracy: 0.8175 - val_loss: 0.4201 - val_precision: 0.8713
- val_recall: 0.7450
Epoch 26/30
13/13 - 18s - 1s/step - accuracy: 0.8250 - loss: 0.3819 - precision: 0.7961 -
recall: 0.8737 - val_accuracy: 0.7800 - val_loss: 0.4500 - val_precision: 0.8415
- val_recall: 0.6900
Epoch 27/30
13/13 - 18s - 1s/step - accuracy: 0.8219 - loss: 0.3824 - precision: 0.8129 -
recall: 0.8363 - val_accuracy: 0.8325 - val_loss: 0.4224 - val_precision: 0.8308
- val_recall: 0.8350
Epoch 28/30
13/13 - 18s - 1s/step - accuracy: 0.8375 - loss: 0.3568 - precision: 0.8140 -
```

```
recall: 0.8750 - val_accuracy: 0.8225 - val_loss: 0.4083 - val_precision: 0.7945
- val_recall: 0.8700
Epoch 29/30
13/13 - 18s - 1s/step - accuracy: 0.8275 - loss: 0.3730 - precision: 0.8126 -
recall: 0.8512 - val_accuracy: 0.8250 - val_loss: 0.3863 - val_precision: 0.7982
- val_recall: 0.8700
Epoch 30/30
13/13 - 18s - 1s/step - accuracy: 0.8275 - loss: 0.3761 - precision: 0.8018 -
recall: 0.8700 - val_accuracy: 0.8175 - val_loss: 0.4190 - val_precision: 0.8757
- val_recall: 0.7400
Restoring model weights from the end of the best epoch: 29.
evaluating best model from this run
4/4             1s 251ms/step -
accuracy: 0.8165 - loss: 0.3983 - precision: 0.5761 - recall: 0.6791
testing: lr=1e-05, filters_conv1=64, units_dense1=64, dropout_rate=0.4
Epoch 1/30
13/13 - 40s - 3s/step - accuracy: 0.5945 - loss: 0.6896 - precision: 0.5990 -
recall: 0.5720 - val_accuracy: 0.6050 - val_loss: 0.6728 - val_precision: 0.7500
- val_recall: 0.3150
Epoch 2/30
13/13 - 39s - 3s/step - accuracy: 0.6256 - loss: 0.6568 - precision: 0.6476 -
recall: 0.5512 - val_accuracy: 0.6350 - val_loss: 0.6574 - val_precision: 0.6274
- val_recall: 0.6650
Epoch 3/30
13/13 - 38s - 3s/step - accuracy: 0.6538 - loss: 0.6374 - precision: 0.6420 -
recall: 0.6950 - val_accuracy: 0.6275 - val_loss: 0.6477 - val_precision: 0.6474
- val_recall: 0.5600
Epoch 4/30
13/13 - 38s - 3s/step - accuracy: 0.6875 - loss: 0.6204 - precision: 0.6923 -
recall: 0.6750 - val_accuracy: 0.6500 - val_loss: 0.6454 - val_precision: 0.6351
- val_recall: 0.7050
Epoch 5/30
13/13 - 38s - 3s/step - accuracy: 0.6931 - loss: 0.6055 - precision: 0.6939 -
recall: 0.6913 - val_accuracy: 0.6225 - val_loss: 0.6406 - val_precision: 0.6244
- val_recall: 0.6150
Epoch 6/30
13/13 - 38s - 3s/step - accuracy: 0.6850 - loss: 0.6030 - precision: 0.6775 -
recall: 0.7063 - val_accuracy: 0.6375 - val_loss: 0.6422 - val_precision: 0.6291
- val_recall: 0.6700
Epoch 7/30
13/13 - 38s - 3s/step - accuracy: 0.7044 - loss: 0.5946 - precision: 0.7051 -
recall: 0.7025 - val_accuracy: 0.6300 - val_loss: 0.6372 - val_precision: 0.6398
- val_recall: 0.5950
Epoch 8/30
13/13 - 38s - 3s/step - accuracy: 0.7131 - loss: 0.5894 - precision: 0.7183 -
recall: 0.7013 - val_accuracy: 0.6500 - val_loss: 0.6341 - val_precision: 0.6429
- val_recall: 0.6750
Epoch 9/30
```

```
13/13 - 38s - 3s/step - accuracy: 0.6981 - loss: 0.5828 - precision: 0.6964 -
recall: 0.7025 - val_accuracy: 0.6350 - val_loss: 0.6321 - val_precision: 0.6552
- val_recall: 0.5700
Epoch 10/30
13/13 - 38s - 3s/step - accuracy: 0.7156 - loss: 0.5767 - precision: 0.7061 -
recall: 0.7387 - val_accuracy: 0.6475 - val_loss: 0.6252 - val_precision: 0.6595
- val_recall: 0.6100
Epoch 11/30
13/13 - 38s - 3s/step - accuracy: 0.7200 - loss: 0.5685 - precision: 0.7245 -
recall: 0.7100 - val_accuracy: 0.6450 - val_loss: 0.6243 - val_precision: 0.6629
- val_recall: 0.5900
Epoch 12/30
13/13 - 38s - 3s/step - accuracy: 0.7025 - loss: 0.5714 - precision: 0.6966 -
recall: 0.7175 - val_accuracy: 0.6450 - val_loss: 0.6196 - val_precision: 0.6629
- val_recall: 0.5900
Epoch 13/30
13/13 - 38s - 3s/step - accuracy: 0.7219 - loss: 0.5593 - precision: 0.7216 -
recall: 0.7225 - val_accuracy: 0.6450 - val_loss: 0.6167 - val_precision: 0.6686
- val_recall: 0.5750
Epoch 14/30
13/13 - 38s - 3s/step - accuracy: 0.7244 - loss: 0.5706 - precision: 0.7155 -
recall: 0.7450 - val_accuracy: 0.6525 - val_loss: 0.6136 - val_precision: 0.6743
- val_recall: 0.5900
Epoch 15/30
13/13 - 38s - 3s/step - accuracy: 0.7331 - loss: 0.5477 - precision: 0.7266 -
recall: 0.7475 - val_accuracy: 0.6450 - val_loss: 0.6196 - val_precision: 0.6859
- val_recall: 0.5350
Epoch 16/30
13/13 - 38s - 3s/step - accuracy: 0.7337 - loss: 0.5512 - precision: 0.7149 -
recall: 0.7775 - val_accuracy: 0.6400 - val_loss: 0.6172 - val_precision: 0.6591
- val_recall: 0.5800
Epoch 17/30
13/13 - 38s - 3s/step - accuracy: 0.7300 - loss: 0.5585 - precision: 0.7323 -
recall: 0.7250 - val_accuracy: 0.6400 - val_loss: 0.6202 - val_precision: 0.6750
- val_recall: 0.5400
Epoch 18/30
13/13 - 38s - 3s/step - accuracy: 0.7437 - loss: 0.5331 - precision: 0.7273 -
recall: 0.7800 - val_accuracy: 0.6600 - val_loss: 0.6107 - val_precision: 0.6667
- val_recall: 0.6400
Epoch 19/30
13/13 - 39s - 3s/step - accuracy: 0.7394 - loss: 0.5414 - precision: 0.7235 -
recall: 0.7750 - val_accuracy: 0.6700 - val_loss: 0.6031 - val_precision: 0.6889
- val_recall: 0.6200
Epoch 20/30
13/13 - 37s - 3s/step - accuracy: 0.7306 - loss: 0.5537 - precision: 0.7220 -
recall: 0.7500 - val_accuracy: 0.6425 - val_loss: 0.6039 - val_precision: 0.6770
- val_recall: 0.5450
Epoch 21/30
```

```
13/13 - 37s - 3s/step - accuracy: 0.7513 - loss: 0.5304 - precision: 0.7399 -
recall: 0.7750 - val_accuracy: 0.6625 - val_loss: 0.5943 - val_precision: 0.6836
- val_recall: 0.6050
Epoch 22/30
13/13 - 38s - 3s/step - accuracy: 0.7419 - loss: 0.5404 - precision: 0.7201 -
recall: 0.7912 - val_accuracy: 0.6400 - val_loss: 0.6025 - val_precision: 0.6772
- val_recall: 0.5350
Epoch 23/30
13/13 - 38s - 3s/step - accuracy: 0.7450 - loss: 0.5329 - precision: 0.7300 -
recall: 0.7775 - val_accuracy: 0.6600 - val_loss: 0.5972 - val_precision: 0.6839
- val_recall: 0.5950
Epoch 24/30
13/13 - 38s - 3s/step - accuracy: 0.7444 - loss: 0.5372 - precision: 0.7234 -
recall: 0.7912 - val_accuracy: 0.6500 - val_loss: 0.6000 - val_precision: 0.6852
- val_recall: 0.5550
Epoch 25/30
13/13 - 37s - 3s/step - accuracy: 0.7369 - loss: 0.5355 - precision: 0.7181 -
recall: 0.7800 - val_accuracy: 0.6400 - val_loss: 0.6004 - val_precision: 0.6842
- val_recall: 0.5200
Epoch 26/30
13/13 - 37s - 3s/step - accuracy: 0.7494 - loss: 0.5273 - precision: 0.7418 -
recall: 0.7650 - val_accuracy: 0.6650 - val_loss: 0.5865 - val_precision: 0.6875
- val_recall: 0.6050
Epoch 27/30
13/13 - 38s - 3s/step - accuracy: 0.7531 - loss: 0.5205 - precision: 0.7273 -
recall: 0.8100 - val_accuracy: 0.6500 - val_loss: 0.6006 - val_precision: 0.7027
- val_recall: 0.5200
Epoch 28/30
13/13 - 37s - 3s/step - accuracy: 0.7444 - loss: 0.5346 - precision: 0.7314 -
recall: 0.7725 - val_accuracy: 0.6575 - val_loss: 0.5960 - val_precision: 0.6864
- val_recall: 0.5800
Epoch 29/30
13/13 - 38s - 3s/step - accuracy: 0.7481 - loss: 0.5189 - precision: 0.7300 -
recall: 0.7875 - val_accuracy: 0.6675 - val_loss: 0.5848 - val_precision: 0.6914
- val_recall: 0.6050
Epoch 30/30
13/13 - 37s - 3s/step - accuracy: 0.7538 - loss: 0.5086 - precision: 0.7494 -
recall: 0.7625 - val_accuracy: 0.6425 - val_loss: 0.5860 - val_precision: 0.6727
- val_recall: 0.5550
Restoring model weights from the end of the best epoch: 29.
evaluating best model from this run
4/4             2s 492ms/step -
accuracy: 0.6938 - loss: 0.5609 - precision: 0.4950 - recall: 0.5019
testing: lr=0.001, filters_conv1=32, units_dense1=128, dropout_rate=0.2
Epoch 1/30
13/13 - 24s - 2s/step - accuracy: 0.5750 - loss: 1.0871 - precision: 0.6280 -
recall: 0.3680 - val_accuracy: 0.6225 - val_loss: 0.6555 - val_precision: 0.6269
- val_recall: 0.6050
```

```
Epoch 2/30
13/13 - 23s - 2s/step - accuracy: 0.7169 - loss: 0.5723 - precision: 0.7188 -
recall: 0.7125 - val_accuracy: 0.6400 - val_loss: 0.6373 - val_precision: 0.6818
- val_recall: 0.5250
Epoch 3/30
13/13 - 23s - 2s/step - accuracy: 0.7425 - loss: 0.5161 - precision: 0.7407 -
recall: 0.7462 - val_accuracy: 0.6775 - val_loss: 0.6110 - val_precision: 0.6437
- val_recall: 0.7950
Epoch 4/30
13/13 - 23s - 2s/step - accuracy: 0.7700 - loss: 0.4887 - precision: 0.7541 -
recall: 0.8012 - val_accuracy: 0.7025 - val_loss: 0.5842 - val_precision: 0.7263
- val_recall: 0.6500
Epoch 5/30
13/13 - 23s - 2s/step - accuracy: 0.7831 - loss: 0.4754 - precision: 0.7542 -
recall: 0.8400 - val_accuracy: 0.7525 - val_loss: 0.5188 - val_precision: 0.7538
- val_recall: 0.7500
Epoch 6/30
13/13 - 23s - 2s/step - accuracy: 0.7975 - loss: 0.4373 - precision: 0.7874 -
recall: 0.8150 - val_accuracy: 0.7850 - val_loss: 0.4912 - val_precision: 0.7500
- val_recall: 0.8550
Epoch 7/30
13/13 - 23s - 2s/step - accuracy: 0.7956 - loss: 0.4415 - precision: 0.7799 -
recall: 0.8238 - val_accuracy: 0.8025 - val_loss: 0.5136 - val_precision: 0.7410
- val_recall: 0.9300
Epoch 8/30
13/13 - 23s - 2s/step - accuracy: 0.8144 - loss: 0.4157 - precision: 0.7976 -
recall: 0.8425 - val_accuracy: 0.7000 - val_loss: 0.5387 - val_precision: 0.8846
- val_recall: 0.4600
Epoch 9/30
13/13 - 24s - 2s/step - accuracy: 0.8263 - loss: 0.3957 - precision: 0.8115 -
recall: 0.8500 - val_accuracy: 0.8100 - val_loss: 0.4560 - val_precision: 0.7627
- val_recall: 0.9000
Epoch 10/30
13/13 - 24s - 2s/step - accuracy: 0.8375 - loss: 0.3783 - precision: 0.8075 -
recall: 0.8863 - val_accuracy: 0.7975 - val_loss: 0.4482 - val_precision: 0.7990
- val_recall: 0.7950
Epoch 11/30
13/13 - 23s - 2s/step - accuracy: 0.8338 - loss: 0.3729 - precision: 0.8240 -
recall: 0.8487 - val_accuracy: 0.7875 - val_loss: 0.4441 - val_precision: 0.8402
- val_recall: 0.7100
Epoch 12/30
13/13 - 23s - 2s/step - accuracy: 0.8375 - loss: 0.3565 - precision: 0.8342 -
recall: 0.8425 - val_accuracy: 0.8300 - val_loss: 0.4056 - val_precision: 0.8267
- val_recall: 0.8350
Epoch 13/30
13/13 - 23s - 2s/step - accuracy: 0.8644 - loss: 0.3354 - precision: 0.8474 -
recall: 0.8888 - val_accuracy: 0.8425 - val_loss: 0.4078 - val_precision: 0.8703
- val_recall: 0.8050
```

```
Epoch 14/30
13/13 - 23s - 2s/step - accuracy: 0.8525 - loss: 0.3345 - precision: 0.8341 -
recall: 0.8800 - val_accuracy: 0.8275 - val_loss: 0.4005 - val_precision: 0.8701
- val_recall: 0.7700
Epoch 15/30
13/13 - 23s - 2s/step - accuracy: 0.8550 - loss: 0.3139 - precision: 0.8541 -
recall: 0.8562 - val_accuracy: 0.8275 - val_loss: 0.3906 - val_precision: 0.7991
- val_recall: 0.8750
Epoch 16/30
13/13 - 23s - 2s/step - accuracy: 0.8569 - loss: 0.3379 - precision: 0.8371 -
recall: 0.8863 - val_accuracy: 0.7025 - val_loss: 0.5101 - val_precision: 0.8857
- val_recall: 0.4650
Epoch 17/30
13/13 - 23s - 2s/step - accuracy: 0.8594 - loss: 0.3152 - precision: 0.8394 -
recall: 0.8888 - val_accuracy: 0.8225 - val_loss: 0.4209 - val_precision: 0.8862
- val_recall: 0.7400
Epoch 18/30
13/13 - 23s - 2s/step - accuracy: 0.8706 - loss: 0.2952 - precision: 0.8585 -
recall: 0.8875 - val_accuracy: 0.8625 - val_loss: 0.3538 - val_precision: 0.8836
- val_recall: 0.8350
Epoch 19/30
13/13 - 23s - 2s/step - accuracy: 0.8587 - loss: 0.3134 - precision: 0.8475 -
recall: 0.8750 - val_accuracy: 0.8575 - val_loss: 0.3632 - val_precision: 0.8667
- val_recall: 0.8450
Epoch 20/30
13/13 - 23s - 2s/step - accuracy: 0.8700 - loss: 0.2817 - precision: 0.8619 -
recall: 0.8813 - val_accuracy: 0.8375 - val_loss: 0.3865 - val_precision: 0.8814
- val_recall: 0.7800
Epoch 21/30
13/13 - 23s - 2s/step - accuracy: 0.8769 - loss: 0.2865 - precision: 0.8585 -
recall: 0.9025 - val_accuracy: 0.8500 - val_loss: 0.3639 - val_precision: 0.8431
- val_recall: 0.8600
Epoch 22/30
13/13 - 23s - 2s/step - accuracy: 0.8875 - loss: 0.2617 - precision: 0.8780 -
recall: 0.9000 - val_accuracy: 0.8600 - val_loss: 0.3447 - val_precision: 0.8600
- val_recall: 0.8600
Epoch 23/30
13/13 - 23s - 2s/step - accuracy: 0.8712 - loss: 0.2950 - precision: 0.8578 -
recall: 0.8900 - val_accuracy: 0.8675 - val_loss: 0.3319 - val_precision: 0.8517
- val_recall: 0.8900
Epoch 24/30
13/13 - 23s - 2s/step - accuracy: 0.8775 - loss: 0.2848 - precision: 0.8595 -
recall: 0.9025 - val_accuracy: 0.8650 - val_loss: 0.3723 - val_precision: 0.8883
- val_recall: 0.8350
Epoch 25/30
13/13 - 23s - 2s/step - accuracy: 0.8913 - loss: 0.2547 - precision: 0.8826 -
recall: 0.9025 - val_accuracy: 0.8850 - val_loss: 0.3166 - val_precision: 0.9010
- val_recall: 0.8650
```

```
Epoch 26/30
13/13 - 23s - 2s/step - accuracy: 0.9094 - loss: 0.2371 - precision: 0.8970 -
recall: 0.9250 - val_accuracy: 0.8700 - val_loss: 0.3426 - val_precision: 0.9205
- val_recall: 0.8100
Epoch 27/30
13/13 - 23s - 2s/step - accuracy: 0.8881 - loss: 0.2535 - precision: 0.8819 -
recall: 0.8963 - val_accuracy: 0.8750 - val_loss: 0.3154 - val_precision: 0.9213
- val_recall: 0.8200
Epoch 28/30
13/13 - 23s - 2s/step - accuracy: 0.8844 - loss: 0.2605 - precision: 0.8773 -
recall: 0.8938 - val_accuracy: 0.8600 - val_loss: 0.3395 - val_precision: 0.9235
- val_recall: 0.7850
Epoch 29/30
13/13 - 24s - 2s/step - accuracy: 0.8888 - loss: 0.2636 - precision: 0.8859 -
recall: 0.8925 - val_accuracy: 0.8700 - val_loss: 0.3254 - val_precision: 0.8895
- val_recall: 0.8450
Epoch 30/30
13/13 - 23s - 2s/step - accuracy: 0.8906 - loss: 0.2484 - precision: 0.8834 -
recall: 0.9000 - val_accuracy: 0.8825 - val_loss: 0.2855 - val_precision: 0.9005
- val_recall: 0.8600
Restoring model weights from the end of the best epoch: 30.
evaluating best model from this run
4/4             2s 338ms/step -
accuracy: 0.8871 - loss: 0.2705 - precision: 0.6804 - recall: 0.6811
testing: lr=1e-05, filters_conv1=16, units_dense1=256, dropout_rate=0.3
Epoch 1/30
13/13 - 20s - 2s/step - accuracy: 0.7115 - loss: 0.6194 - precision: 0.7311 -
recall: 0.6690 - val_accuracy: 0.6200 - val_loss: 0.6825 - val_precision: 0.6053
- val_recall: 0.6900
Epoch 2/30
13/13 - 18s - 1s/step - accuracy: 0.7381 - loss: 0.5477 - precision: 0.7408 -
recall: 0.7325 - val_accuracy: 0.6050 - val_loss: 0.6734 - val_precision: 0.6129
- val_recall: 0.5700
Epoch 3/30
13/13 - 18s - 1s/step - accuracy: 0.7212 - loss: 0.5481 - precision: 0.7287 -
recall: 0.7050 - val_accuracy: 0.6200 - val_loss: 0.6405 - val_precision: 0.6446
- val_recall: 0.5350
Epoch 4/30
13/13 - 18s - 1s/step - accuracy: 0.7631 - loss: 0.5091 - precision: 0.7615 -
recall: 0.7663 - val_accuracy: 0.6450 - val_loss: 0.6259 - val_precision: 0.6686
- val_recall: 0.5750
Epoch 5/30
13/13 - 18s - 1s/step - accuracy: 0.7625 - loss: 0.5044 - precision: 0.7672 -
recall: 0.7538 - val_accuracy: 0.6250 - val_loss: 0.6304 - val_precision: 0.6667
- val_recall: 0.5000
Epoch 6/30
13/13 - 18s - 1s/step - accuracy: 0.7581 - loss: 0.5139 - precision: 0.7604 -
recall: 0.7538 - val_accuracy: 0.6625 - val_loss: 0.6049 - val_precision: 0.6816
```

```
- val_recall: 0.6100
Epoch 7/30
13/13 - 18s - 1s/step - accuracy: 0.7669 - loss: 0.4867 - precision: 0.7515 -
recall: 0.7975 - val_accuracy: 0.6675 - val_loss: 0.5859 - val_precision: 0.7055
- val_recall: 0.5750
Epoch 8/30
13/13 - 18s - 1s/step - accuracy: 0.7769 - loss: 0.4920 - precision: 0.7646 -
recall: 0.8000 - val_accuracy: 0.6875 - val_loss: 0.5775 - val_precision: 0.7551
- val_recall: 0.5550
Epoch 9/30
13/13 - 18s - 1s/step - accuracy: 0.7675 - loss: 0.4895 - precision: 0.7716 -
recall: 0.7600 - val_accuracy: 0.7300 - val_loss: 0.5494 - val_precision: 0.7614
- val_recall: 0.6700
Epoch 10/30
13/13 - 18s - 1s/step - accuracy: 0.7769 - loss: 0.4775 - precision: 0.7597 -
recall: 0.8100 - val_accuracy: 0.7300 - val_loss: 0.5456 - val_precision: 0.7614
- val_recall: 0.6700
Epoch 11/30
13/13 - 18s - 1s/step - accuracy: 0.7844 - loss: 0.4676 - precision: 0.7771 -
recall: 0.7975 - val_accuracy: 0.7275 - val_loss: 0.5388 - val_precision: 0.7758
- val_recall: 0.6400
Epoch 12/30
13/13 - 18s - 1s/step - accuracy: 0.7962 - loss: 0.4533 - precision: 0.7985 -
recall: 0.7925 - val_accuracy: 0.7000 - val_loss: 0.5457 - val_precision: 0.7941
- val_recall: 0.5400
Epoch 13/30
13/13 - 18s - 1s/step - accuracy: 0.7744 - loss: 0.4698 - precision: 0.7654 -
recall: 0.7912 - val_accuracy: 0.6775 - val_loss: 0.5693 - val_precision: 0.7934
- val_recall: 0.4800
Epoch 14/30
13/13 - 18s - 1s/step - accuracy: 0.8012 - loss: 0.4560 - precision: 0.7783 -
recall: 0.8425 - val_accuracy: 0.6675 - val_loss: 0.6049 - val_precision: 0.8941
- val_recall: 0.3800
Epoch 15/30
13/13 - 18s - 1s/step - accuracy: 0.7906 - loss: 0.4614 - precision: 0.7888 -
recall: 0.7937 - val_accuracy: 0.7425 - val_loss: 0.5306 - val_precision: 0.7594
- val_recall: 0.7100
Epoch 16/30
13/13 - 18s - 1s/step - accuracy: 0.7950 - loss: 0.4453 - precision: 0.7843 -
recall: 0.8138 - val_accuracy: 0.7625 - val_loss: 0.5203 - val_precision: 0.7512
- val_recall: 0.7850
Epoch 17/30
13/13 - 18s - 1s/step - accuracy: 0.8106 - loss: 0.4408 - precision: 0.8027 -
recall: 0.8238 - val_accuracy: 0.7575 - val_loss: 0.5185 - val_precision: 0.7668
- val_recall: 0.7400
Epoch 18/30
13/13 - 18s - 1s/step - accuracy: 0.8081 - loss: 0.4325 - precision: 0.7938 -
recall: 0.8325 - val_accuracy: 0.7600 - val_loss: 0.5000 - val_precision: 0.7653
```

```
- val_recall: 0.7500
Epoch 19/30
13/13 - 18s - 1s/step - accuracy: 0.8006 - loss: 0.4410 - precision: 0.7929 -
recall: 0.8138 - val_accuracy: 0.7575 - val_loss: 0.5006 - val_precision: 0.8601
- val_recall: 0.6150
Epoch 20/30
13/13 - 18s - 1s/step - accuracy: 0.8125 - loss: 0.4181 - precision: 0.8109 -
recall: 0.8150 - val_accuracy: 0.7650 - val_loss: 0.4985 - val_precision: 0.8193
- val_recall: 0.6800
Epoch 21/30
13/13 - 18s - 1s/step - accuracy: 0.8125 - loss: 0.4332 - precision: 0.8086 -
recall: 0.8188 - val_accuracy: 0.7825 - val_loss: 0.4956 - val_precision: 0.8266
- val_recall: 0.7150
Epoch 22/30
13/13 - 18s - 1s/step - accuracy: 0.8131 - loss: 0.4204 - precision: 0.7944 -
recall: 0.8450 - val_accuracy: 0.7150 - val_loss: 0.5418 - val_precision: 0.8772
- val_recall: 0.5000
Epoch 23/30
13/13 - 18s - 1s/step - accuracy: 0.8075 - loss: 0.4351 - precision: 0.7874 -
recall: 0.8425 - val_accuracy: 0.7700 - val_loss: 0.4965 - val_precision: 0.8462
- val_recall: 0.6600
Epoch 24/30
13/13 - 18s - 1s/step - accuracy: 0.8087 - loss: 0.4152 - precision: 0.8159 -
recall: 0.7975 - val_accuracy: 0.7750 - val_loss: 0.4853 - val_precision: 0.7723
- val_recall: 0.7800
Epoch 25/30
13/13 - 18s - 1s/step - accuracy: 0.8125 - loss: 0.4127 - precision: 0.7990 -
recall: 0.8350 - val_accuracy: 0.7825 - val_loss: 0.4923 - val_precision: 0.8792
- val_recall: 0.6550
Epoch 26/30
13/13 - 18s - 1s/step - accuracy: 0.8150 - loss: 0.4060 - precision: 0.7986 -
recall: 0.8425 - val_accuracy: 0.7875 - val_loss: 0.4814 - val_precision: 0.8859
- val_recall: 0.6600
Epoch 27/30
13/13 - 18s - 1s/step - accuracy: 0.8275 - loss: 0.3966 - precision: 0.8164 -
recall: 0.8450 - val_accuracy: 0.7700 - val_loss: 0.4867 - val_precision: 0.8699
- val_recall: 0.6350
Epoch 28/30
13/13 - 18s - 1s/step - accuracy: 0.8281 - loss: 0.4007 - precision: 0.8285 -
recall: 0.8275 - val_accuracy: 0.7500 - val_loss: 0.5064 - val_precision: 0.8968
- val_recall: 0.5650
Epoch 29/30
13/13 - 18s - 1s/step - accuracy: 0.8188 - loss: 0.4003 - precision: 0.8087 -
recall: 0.8350 - val_accuracy: 0.7650 - val_loss: 0.4743 - val_precision: 0.7849
- val_recall: 0.7300
Epoch 30/30
13/13 - 18s - 1s/step - accuracy: 0.8175 - loss: 0.4019 - precision: 0.8151 -
recall: 0.8213 - val_accuracy: 0.7975 - val_loss: 0.4629 - val_precision: 0.8362
```

```
- val_recall: 0.7400
Restoring model weights from the end of the best epoch: 30.
evaluating best model from this run
4/4             1s 265ms/step -
accuracy: 0.8172 - loss: 0.4547 - precision: 0.6153 - recall: 0.5867
hyperparameter tuning finished.
```

[21]:
```python
# convert results to dataframe
results_df = pd.DataFrame(results_list)

# sort by validation accuracy (descending)
results_df = results_df.sort_values(by="val_recall", ascending=False)

# print results
print("\ntuning results summary:")
print(results_df)

# save results to csv
results_filename = "../results/hyperparameter_tuning_results_random_search.csv"
results_df.to_csv(results_filename, index=False)
print(f"\nresults saved to {results_filename}")
```

```
tuning results summary:
   learning_rate  filters_conv1  units_dense1  dropout_rate  val_loss  \
1        0.00010             16           128           0.4  0.386285
3        0.00100             32           128           0.2  0.285464
0        0.00100             16            64           0.2  0.402848
4        0.00001             16           256           0.3  0.462921
2        0.00001             64            64           0.4  0.584815

   val_accuracy  val_precision  val_recall  epochs_trained
1        0.8250       0.798165       0.870              30
3        0.8825       0.900524       0.860              30
0        0.8125       0.801932       0.830              13
4        0.7975       0.836158       0.740              30
2        0.6675       0.691429       0.605              30

results saved to ../results/hyperparameter_tuning_results_random_search.csv
```

[22]:
```python
# concatenate train and val data
X_all = []
y_all = []

train_data_gen.reset()
val_data_gen.reset()

for batch_x, batch_y in train_data_gen:
```

```
    X_all.append(batch_x)
    y_all.append(batch_y)
    if len(X_all) * BATCH_SIZE >= train_data_gen.samples:
        break

for batch_x, batch_y in val_data_gen:
    X_all.append(batch_x)
    y_all.append(batch_y)
    if len(X_all) * BATCH_SIZE >= val_data_gen.samples + train_data_gen.samples:
        break

X_all = np.concatenate(X_all)
y_all = np.concatenate(y_all)

# create dataset
train_val_ds = (
    tf.data.Dataset.from_tensor_slices((X_all, y_all))
    .shuffle(1000)
    .batch(BATCH_SIZE)
    .prefetch(tf.data.AUTOTUNE)
)
```

[23]:
```
print("\nretraining the best model on the combined train+validation data")
best_params = results_df.iloc[0]
best_lr = best_params["learning_rate"]
best_filters_conv1 = int(
    best_params["filters_conv1"]
)  # cast to int, wasn't working before
best_units_dense1 = int(
    best_params["units_dense1"]
)  # cast to int, wasn't working before
best_dropout_rate = best_params["dropout_rate"]
print(
    f"best hyperparameters found: lr={best_lr},␣
 ↪filters_conv1={best_filters_conv1}, units_dense1={best_units_dense1},␣
 ↪dropout_rate={best_dropout_rate}"
)
```

```
retraining the best model on the combined train+validation data
best hyperparameters found: lr=0.0001, filters_conv1=16, units_dense1=128,
dropout_rate=0.4
```

[24]:
```
# build the best model
final_model = build_model(
    input_shape,
    filters_conv1=best_filters_conv1,
```

```
    units_dense1=best_units_dense1,
    dropout_rate=best_dropout_rate,
)

# compile the best model
optimizer = tf.keras.optimizers.Adam(learning_rate=best_lr)
final_model.compile(optimizer=optimizer, loss="binary_crossentropy",␣
 ↪metrics=metrics)

# define steps per epoch
steps_per_epoch = (train_data_gen.samples + val_data_gen.samples) // BATCH_SIZE

# train the best model on the combined train+val data
final_history = final_model.fit(
    train_val_ds,
    epochs=EPOCHS,
    steps_per_epoch=steps_per_epoch,
    verbose=2,
)
```

```
Epoch 1/30
15/15 - 20s - 1s/step - accuracy: 0.6328 - loss: 0.6531 - precision: 0.6823 -
recall: 0.4826
Epoch 2/30
15/15 - 1s - 40ms/step - accuracy: 0.6719 - loss: 0.6383 - precision: 0.7143 -
recall: 0.6944
Epoch 3/30
```

```
/opt/anaconda3/envs/ml-2025/lib/python3.12/site-
packages/keras/src/trainers/epoch_iterator.py:107: UserWarning: Your input ran
out of data; interrupting training. Make sure that your dataset or generator can
generate at least `steps_per_epoch * epochs` batches. You may need to use the
`.repeat()` function when building your dataset.
  self._interrupted_warning()
```

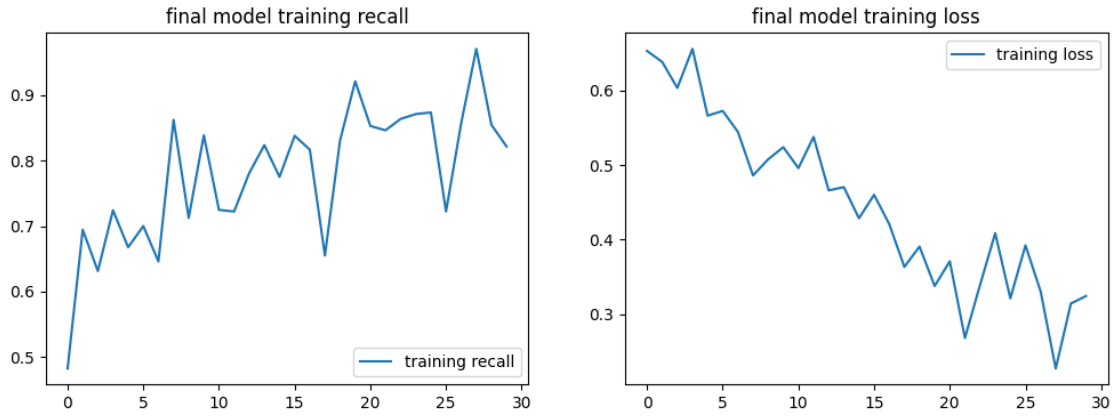
```
15/15 - 18s - 1s/step - accuracy: 0.6958 - loss: 0.6035 - precision: 0.7222 -
recall: 0.6314
Epoch 4/30
15/15 - 1s - 41ms/step - accuracy: 0.7031 - loss: 0.6559 - precision: 0.6562 -
recall: 0.7241
Epoch 5/30
15/15 - 18s - 1s/step - accuracy: 0.7130 - loss: 0.5663 - precision: 0.7313 -
recall: 0.6677
Epoch 6/30
15/15 - 1s - 41ms/step - accuracy: 0.7656 - loss: 0.5726 - precision: 0.7778 -
recall: 0.7000
Epoch 7/30
15/15 - 18s - 1s/step - accuracy: 0.7260 - loss: 0.5446 - precision: 0.7665 -
recall: 0.6461
```

```
Epoch 8/30
15/15 - 1s - 40ms/step - accuracy: 0.7656 - loss: 0.4862 - precision: 0.6944 -
recall: 0.8621
Epoch 9/30
15/15 - 18s - 1s/step - accuracy: 0.7391 - loss: 0.5077 - precision: 0.7494 -
recall: 0.7125
Epoch 10/30
15/15 - 1s - 41ms/step - accuracy: 0.7344 - loss: 0.5241 - precision: 0.6842 -
recall: 0.8387
Epoch 11/30
15/15 - 18s - 1s/step - accuracy: 0.7573 - loss: 0.4960 - precision: 0.7702 -
recall: 0.7247
Epoch 12/30
15/15 - 1s - 41ms/step - accuracy: 0.7188 - loss: 0.5379 - precision: 0.7647 -
recall: 0.7222
Epoch 13/30
15/15 - 18s - 1s/step - accuracy: 0.7896 - loss: 0.4661 - precision: 0.7910 -
recall: 0.7811
Epoch 14/30
15/15 - 1s - 41ms/step - accuracy: 0.8281 - loss: 0.4705 - precision: 0.8485 -
recall: 0.8235
Epoch 15/30
15/15 - 18s - 1s/step - accuracy: 0.8068 - loss: 0.4287 - precision: 0.8229 -
recall: 0.7751
Epoch 16/30
15/15 - 1s - 40ms/step - accuracy: 0.7812 - loss: 0.4602 - precision: 0.7949 -
recall: 0.8378
Epoch 17/30
15/15 - 18s - 1s/step - accuracy: 0.8125 - loss: 0.4210 - precision: 0.8083 -
recall: 0.8168
Epoch 18/30
15/15 - 1s - 42ms/step - accuracy: 0.7812 - loss: 0.3636 - precision: 0.8261 -
recall: 0.6552
Epoch 19/30
15/15 - 18s - 1s/step - accuracy: 0.8302 - loss: 0.3909 - precision: 0.8256 -
recall: 0.8309
Epoch 20/30
15/15 - 1s - 40ms/step - accuracy: 0.9062 - loss: 0.3380 - precision: 0.9211 -
recall: 0.9211
Epoch 21/30
15/15 - 18s - 1s/step - accuracy: 0.8458 - loss: 0.3711 - precision: 0.8370 -
recall: 0.8529
Epoch 22/30
15/15 - 1s - 40ms/step - accuracy: 0.8906 - loss: 0.2683 - precision: 0.9706 -
recall: 0.8462
Epoch 23/30
15/15 - 18s - 1s/step - accuracy: 0.8620 - loss: 0.3395 - precision: 0.8591 -
recall: 0.8636
```

```
Epoch 24/30
15/15 - 1s - 41ms/step - accuracy: 0.8594 - loss: 0.4087 - precision: 0.8438 -
recall: 0.8710
Epoch 25/30
15/15 - 18s - 1s/step - accuracy: 0.8698 - loss: 0.3215 - precision: 0.8643 -
recall: 0.8734
Epoch 26/30
15/15 - 1s - 42ms/step - accuracy: 0.7812 - loss: 0.3924 - precision: 0.8667 -
recall: 0.7222
Epoch 27/30
15/15 - 19s - 1s/step - accuracy: 0.8630 - loss: 0.3308 - precision: 0.8650 -
recall: 0.8568
Epoch 28/30
15/15 - 1s - 41ms/step - accuracy: 0.9375 - loss: 0.2274 - precision: 0.9167 -
recall: 0.9706
Epoch 29/30
15/15 - 18s - 1s/step - accuracy: 0.8719 - loss: 0.3145 - precision: 0.8842 -
recall: 0.8546
Epoch 30/30
15/15 - 1s - 41ms/step - accuracy: 0.8438 - loss: 0.3246 - precision: 0.8214 -
recall: 0.8214
```

[25]:
```
print("\nplotting final training history")
final_recall = final_history.history["recall"]
final_loss = final_history.history["loss"]

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, final_recall, label="training recall")
plt.legend(loc="lower right")
plt.title("final model training recall")

plt.subplot(1, 2, 2)
plt.plot(epochs_range, final_loss, label="training loss")
plt.legend(loc="upper right")
plt.title("final model training loss")
plt.show()
```

```
plotting final training history
```



[26]:
```
model_save_path = "../models/baseline_model.keras"
print(f"\nsaving final model to {model_save_path}")
final_model.save(model_save_path)

print("loading model back")
loaded_model = tf.keras.models.load_model(model_save_path)
print("model loaded successfully")
```

```
saving final model to ../models/baseline_model.keras
loading model back
model loaded successfully
```

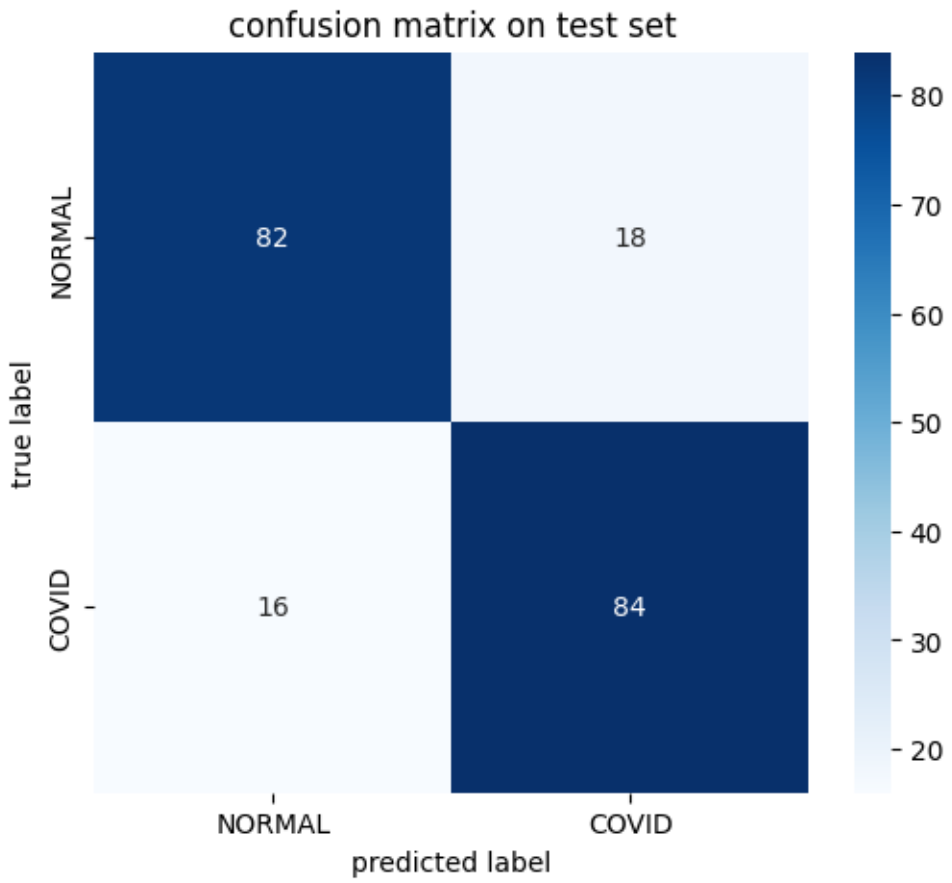
[27]:
```
print("\nevaluating loaded model on test data")
test_results = loaded_model.evaluate(test_data_gen, verbose=1)
print(f"test loss: {test_results[0]}")
print(f"test accuracy: {test_results[1]}")
print(f"test precision: {test_results[2]}")
print(f"test recall: {test_results[3]}")
```

```
evaluating loaded model on test data

/opt/anaconda3/envs/ml-2025/lib/python3.12/site-
packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in
its constructor. `**kwargs` can include `workers`, `use_multiprocessing`,
`max_queue_size`. Do not pass these arguments to `fit()`, as they will be
ignored.
  self._warn_if_super_not_called()

2/2             1s 220ms/step -
accuracy: 0.8346 - loss: 0.3719 - precision: 0.7460 - recall: 0.8695
test loss: 0.372536838054657
```

```
test accuracy: 0.8299999833106995
test precision: 0.8235294222831726
test recall: 0.8399999737739563
```

[28]:
```python
print("\ngenerating confusion matrix")
# get predictions (probabilities)
y_pred_prob = loaded_model.predict(test_data_gen)
# convert probabilities to binary predictions
y_pred = (y_pred_prob > 0.5).astype(int).flatten()
# get true labels
y_true = test_data_gen.classes
# calculate confusion matrix
cm = confusion_matrix(y_true, y_pred)
# plot confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(
    cm,
    annot=True,
    fmt="d",
    cmap="Blues",
    xticklabels=class_names,
    yticklabels=class_names,
)
plt.xlabel("predicted label")
plt.ylabel("true label")
plt.title("confusion matrix on test set")
plt.show()
```

```
generating confusion matrix
2/2             1s 229ms/step
```



[29]:
```
print("\nplotting sample test images with predictions")
# get a batch of raw (unnormalized) images and labels
images_raw, labels_raw = next(iter(test_data_gen_raw))
# get predictions for this batch using the normalized data generator
# important: ensure we use the *same batch* for predictions
# we need to reset the generator to be sure we get the same batch
test_data_gen.reset()
images_norm, _ = next(iter(test_data_gen))  # get normalized images for␣
 ↪prediction
batch_pred_prob = loaded_model.predict(images_norm)
batch_pred = (batch_pred_prob > 0.5).astype(int).flatten()

plt.figure(figsize=(10, 10))
for i in range(9):  # display 9 samples
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images_raw[i].astype("uint8"))  # display raw image
    true_label = class_names[int(labels_raw[i])]  # cast to int
    pred_label = class_names[batch_pred[i]]
    prob = batch_pred_prob[i][0]
```

```
    plt.title(f"true: {true_label}\npred: {pred_label} ({prob:.2f})")
    plt.axis("off")
plt.tight_layout()
plt.show()
```

```
plotting sample test images with predictions
4/4             0s 88ms/step
```