In [ ]:
```
# Basic tools
import os, cv2, time, itertools
import numpy as np
import matplotlib.pyplot as plt

# Feature extraction
from skimage.feature import hog, local_binary_pattern   # ✅ only HOG + LBP

# Machine learning
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Deep learning
import tensorflow as tf
from tensorflow.keras import layers, models
```

In [15]:
```
# Path to dataset
data_dir = "/kaggle/input/chest-xray-pneumonia/chest_xray"

train_dir = os.path.join(data_dir, "train")
test_dir  = os.path.join(data_dir, "test")

# Function to load images from folder
def load_images_from_folder(folder, img_size=(128,128)):
    images = []
    labels = []
    for label, class_name in enumerate(["NORMAL", "PNEUMONIA"]):
        class_folder = os.path.join(folder, class_name)
        for filename in os.listdir(class_folder):
            img_path = os.path.join(class_folder, filename)
            img = cv2.imread(img_path)
            if img is not None:
                img = cv2.resize(img, img_size)
                images.append(img)
                labels.append(label)
    return np.array(images), np.array(labels)

# Load training and test data
X_train, y_train = load_images_from_folder(train_dir)
X_test, y_test   = load_images_from_folder(test_dir)

print("Train set:", X_train.shape, y_train.shape)
print("Test set:", X_test.shape, y_test.shape)
```

```
Train set: (5216, 128, 128, 3) (5216,)
Test set: (624, 128, 128, 3) (624,)
```

In [17]:
```
def extract_features(images):
    features = []
    for img in images:
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        # HOG features
        hog_features, _ = hog(
            gray, orientations=9, pixels_per_cell=(8, 8),
            cells_per_block=(2, 2), visualize=True
        )

        # LBP features
        lbp = local_binary_pattern(gray, P=8, R=1, method='uniform')
        lbp_hist, _ = np.histogram(
```

```
            lbp.ravel(), bins=np.arange(0, 11), range=(0, 10)
        )
        lbp_hist = lbp_hist.astype("float")
        lbp_hist /= (lbp_hist.sum() + 1e-7)

        # Combine features
        feature_vector = np.hstack([hog_features, lbp_hist])
        features.append(feature_vector)

    return np.array(features)

print("Extracting features for SVM...")
X_train_features = extract_features(X_train)
X_test_features  = extract_features(X_test)
```

```
Extracting features for SVM...
```

In [18]:

```
print("Training SVM with GridSearchCV...")
param_grid = {'C':[0.1, 1, 10], 'kernel':['linear','rbf']}
svm = GridSearchCV(SVC(), param_grid, cv=3)
svm.fit(X_train_features, y_train)

# Predictions
y_pred_svm = svm.predict(X_test_features)

# Accuracy
svm_acc = accuracy_score(y_test, y_pred_svm)
print("SVM Accuracy:", svm_acc)
print(classification_report(y_test, y_pred_svm))
```

```
Training SVM with GridSearchCV...
SVM Accuracy: 0.7451923076923077
              precision    recall  f1-score   support

           0       0.97      0.33      0.49       234
           1       0.71      0.99      0.83       390

    accuracy                           0.75       624
   macro avg       0.84      0.66      0.66       624
weighted avg       0.81      0.75      0.70       624
```

In [19]:

```
img_height, img_width = 128, 128

# Normalize images
X_train_norm = X_train / 255.0
X_test_norm  = X_test / 255.0

# CNN architecture
cnn_model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(img_height, img_width, 3
    layers.MaxPooling2D((2,2)),

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

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

    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(1, activation='sigmoid')   # binary classification
])
```

```
cnn_model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

cnn_model.summary()

# Train CNN
print("Training CNN...")
history = cnn_model.fit(
    X_train_norm, y_train,
    epochs=5,
    batch_size=32,
    validation_split=0.2,
    verbose=1
)

# Evaluate CNN
cnn_loss, cnn_acc = cnn_model.evaluate(X_test_norm, y_test)
print("CNN Accuracy:", cnn_acc)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
2025-09-11 05:38:03.249275: E external/local_xla/xla/stream_executor/cuda/cuda_driver.cc:152] failed call to cuInit: INTERNAL: CUDA error: Failed call to cuInit: UNKNOWN ERROR (303)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 128)	3,211,392
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Total params: 3,304,769 (12.61 MB)

Trainable params: 3,304,769 (12.61 MB)

Non-trainable params: 0 (0.00 B)

```
Training CNN...
Epoch 1/5
131/131 ━━━━━━━━━━━━━━━━━━━━ 75s 546ms/step - accuracy: 0.7503 - loss: 0.5123 - val_ac
curacy: 0.9272 - val_loss: 0.1868
Epoch 2/5
131/131 ━━━━━━━━━━━━━━━━━━━━ 71s 540ms/step - accuracy: 0.9426 - loss: 0.1562 - val_ac
curacy: 0.9511 - val_loss: 0.1275
Epoch 3/5
131/131 ━━━━━━━━━━━━━━━━━━━━ 70s 534ms/step - accuracy: 0.9597 - loss: 0.1115 - val_ac
curacy: 0.9732 - val_loss: 0.0547
Epoch 4/5
131/131 ━━━━━━━━━━━━━━━━━━━━ 69s 530ms/step - accuracy: 0.9697 - loss: 0.0895 - val_ac
curacy: 0.9511 - val_loss: 0.1181
Epoch 5/5
131/131 ━━━━━━━━━━━━━━━━━━━━ 68s 520ms/step - accuracy: 0.9740 - loss: 0.0783 - val_ac
curacy: 0.9607 - val_loss: 0.1098
20/20 ━━━━━━━━━━━━━━━━━━━━ 5s 260ms/step - accuracy: 0.6488 - loss: 1.0857
CNN Accuracy: 0.8028846383094788
```

In [20]:

```python
# Bar chart comparison
models = ['SVM', 'CNN']
accuracies = [svm_acc, cnn_acc]

plt.figure(figsize=(6,5))
plt.bar(models, accuracies, color=['blue','green'])
plt.ylim(0,1)
plt.ylabel("Accuracy")
plt.title("SVM vs CNN Accuracy on Chest X-Ray Dataset")

# Add values above bars
for i, v in enumerate(accuracies):
    plt.text(i, v + 0.02, f"{v:.2f}", ha='center', fontsize=12)

plt.show()
```