CNN Models

By Shray Komal

Data Set

Kidney Cancer Dataset

-Normal: 5077 images

-Tumor: 2283 images

-Stone: 1377 images

-Cyst: 3709 images

2 Classes

-Normal: 5077 images

-Abnormal: 7369 images

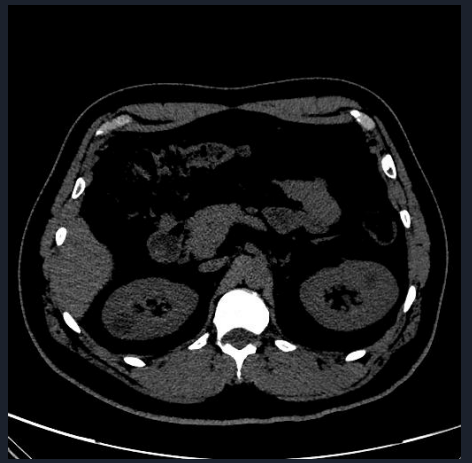
Sample Data

Normal

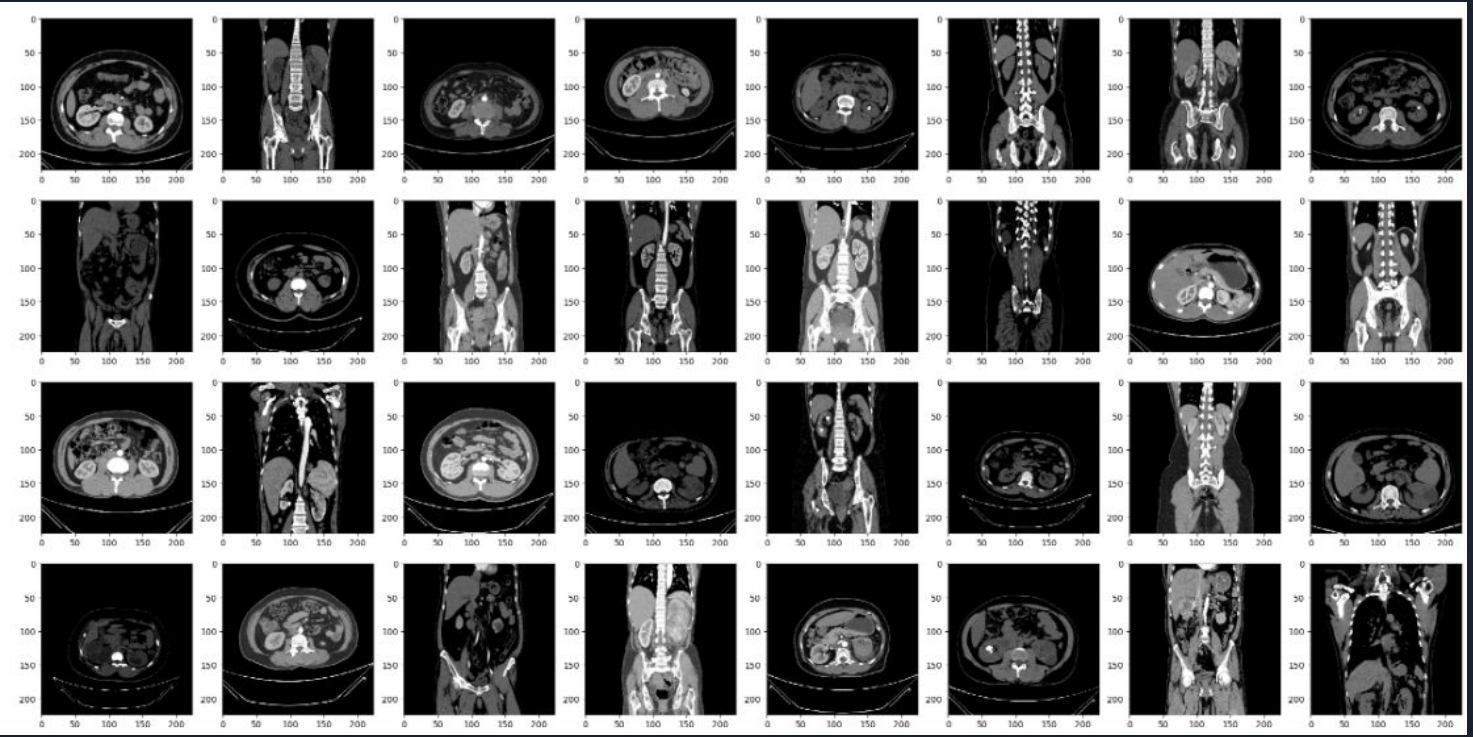
Tumor

Stone

Cyst



Sample Data



Split Code for Train, Test, Validation

```python
def train_test_validation_split(src_folder: pathlib.PosixPath, class_name: str) -> dict:
    # For tracking
    n_train, n_valid, n_test = 0, 0, 0

    # Random seed for reproducibility
    random.seed(42)

    # Iterate over every image
    for file in src_folder.iterdir():
        img_name = str(file).split('\\')[-1]

        # Make sure it's JPG
        if file.suffix == '.jpg':
            # Generate a random number
            x = random.random()

            # Where should the image go?
            tgt_dir = ''

            # .80 or below
            if x <= pct_train:
                tgt_dir = 'train'
                n_train += 1

            # Between .80 and .90
            elif pct_train < x <= (pct_train + pct_valid):
                tgt_dir = 'validation'
                n_valid += 1

            # Above .90
            else:
                tgt_dir = 'test'
                n_test += 1

            # Copy the image
            shutil.copy(
                src=file,
                # data/<train|valid|test>/<cat\dog>/<something>.jpg
                dst=f'{str(dir_data)}/{tgt_dir}/{class_name}/{img_name}'
            )

    return {
        'source': str(src_folder),
        'target': str(dir_data),
         'n_train': n_train,
        'n_validaiton': n_valid,
        'n_test': n_test
    }
```

Model 1

```
model_1 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(filters=16, kernel_size=(3, 3), input_shape=(224, 224, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(2, 2), padding='same'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])

model_1.compile(
    loss=tf.keras.losses.categorical_crossentropy,
    optimizer=tf.keras.optimizers.Adam(),
    metrics=[tf.keras.metrics.BinaryAccuracy(name='accuracy')]
)

history_1 = model_1.fit(
    train_data,
    validation_data=valid_data,
    epochs=10
)
```

Results

```
Epoch 1/10
157/157 [==============================] - 344s 2s/step - loss: 0.3198 - accuracy: 0.9476 - val_loss: 0.0018 - val_accuracy: 1.
0000
Epoch 2/10
157/157 [==============================] - 248s 2s/step - loss: 6.1482e-04 - accuracy: 1.0000 - val_loss: 2.0156e-04 - val_accu
racy: 1.0000
Epoch 3/10
157/157 [==============================] - 246s 2s/step - loss: 8.8530e-05 - accuracy: 1.0000 - val_loss: 8.4984e-05 - val_accu
racy: 1.0000
Epoch 4/10
157/157 [==============================] - 240s 2s/step - loss: 3.9572e-05 - accuracy: 1.0000 - val_loss: 4.6502e-05 - val_accu
racy: 1.0000
Epoch 5/10
157/157 [==============================] - 252s 2s/step - loss: 2.2388e-05 - accuracy: 1.0000 - val_loss: 3.0255e-05 - val_accu
racy: 1.0000
Epoch 6/10
157/157 [==============================] - 241s 2s/step - loss: 1.4246e-05 - accuracy: 1.0000 - val_loss: 2.2376e-05 - val_accu
racy: 1.0000
Epoch 7/10
157/157 [==============================] - 238s 2s/step - loss: 9.8275e-06 - accuracy: 1.0000 - val_loss: 1.5069e-05 - val_accu
racy: 1.0000
Epoch 8/10
157/157 [==============================] - 288s 2s/step - loss: 7.1694e-06 - accuracy: 1.0000 - val_loss: 1.1973e-05 - val_accu
racy: 1.0000
Epoch 9/10
157/157 [==============================] - 242s 2s/step - loss: 5.4386e-06 - accuracy: 1.0000 - val_loss: 9.1219e-06 - val_accu
racy: 1.0000
Epoch 10/10
157/157 [==============================] - 235s 1s/step - loss: 4.2489e-06 - accuracy: 1.0000 - val_loss: 7.4094e-06 - val_accu
racy: 1.0000
```

Model 2

```
model_2 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), input_shape=(224, 224, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(2, 2), padding='same'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])

model_2.compile(
    loss=tf.keras.losses.categorical_crossentropy,
    optimizer=tf.keras.optimizers.Adam(),
    metrics=[tf.keras.metrics.BinaryAccuracy(name='accuracy')]
)

history_2 = model_2.fit(
    train_data,
    validation_data=valid_data,
    epochs=10
)
```

Results

```
Epoch 1/10
157/157 [==============================] - 660s 4s/step - loss: 0.7617 - accuracy: 0.8631 - val_loss: 0.1119 - val_accuracy: 0.
9666
Epoch 2/10
157/157 [==============================] - 592s 4s/step - loss: 0.1090 - accuracy: 0.9831 - val_loss: 0.0936 - val_accuracy: 0.
9886
Epoch 3/10
157/157 [==============================] - 549s 3s/step - loss: 0.0911 - accuracy: 0.9937 - val_loss: 0.0850 - val_accuracy: 0.
9764
Epoch 4/10
157/157 [==============================] - 4971s 32s/step - loss: 0.0724 - accuracy: 0.9996 - val_loss: 0.0621 - val_accuracy:
0.9984
Epoch 5/10
157/157 [==============================] - 469s 3s/step - loss: 0.0598 - accuracy: 0.9999 - val_loss: 0.0519 - val_accuracy: 1.
0000
Epoch 6/10
157/157 [==============================] - 386s 2s/step - loss: 0.0507 - accuracy: 1.0000 - val_loss: 0.0443 - val_accuracy: 1.
0000
Epoch 7/10
157/157 [==============================] - 392s 2s/step - loss: 0.0430 - accuracy: 1.0000 - val_loss: 0.0375 - val_accuracy: 1.
0000
Epoch 8/10
157/157 [==============================] - 414s 3s/step - loss: 0.0367 - accuracy: 1.0000 - val_loss: 0.0322 - val_accuracy: 1.
0000
Epoch 9/10
157/157 [==============================] - 354s 2s/step - loss: 0.0316 - accuracy: 1.0000 - val_loss: 0.0278 - val_accuracy: 1.
0000
Epoch 10/10
157/157 [==============================] - 380s 2s/step - loss: 0.0274 - accuracy: 1.0000 - val_loss: 0.0242 - val_accuracy: 1.
0000
```

Model 3

```
model_3 = tf.keras.Sequential([
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), input_shape=(224, 224, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(2, 2), padding='same'),
    tf.keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu'),
    tf.keras.layers.MaxPool2D(pool_size=(2, 2), padding='same'),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(2, activation='softmax')
])

model_3.compile(
    loss=tf.keras.losses.categorical_crossentropy,
    optimizer=tf.keras.optimizers.Adam(),
    metrics=[tf.keras.metrics.BinaryAccuracy(name='accuracy')]
)

history_3 = model_3.fit(
    train_data,
    validation_data=valid_data,
    epochs=10
)
```

Results

```
Epoch 1/10
157/157 [==============================] - 841s 5s/step - loss: 0.1307 - accuracy: 0.9494 - val_loss: 3.3676e-04 - val_accurac
y: 1.0000
Epoch 2/10
157/157 [==============================] - 548s 3s/step - loss: 1.1737e-04 - accuracy: 1.0000 - val_loss: 6.0396e-05 - val_accu
racy: 1.0000
Epoch 3/10
157/157 [==============================] - 408s 3s/step - loss: 3.2695e-05 - accuracy: 1.0000 - val_loss: 2.7961e-05 - val_accu
racy: 1.0000
Epoch 4/10
157/157 [==============================] - 397s 3s/step - loss: 1.6887e-05 - accuracy: 1.0000 - val_loss: 1.7390e-05 - val_accu
racy: 1.0000
Epoch 5/10
157/157 [==============================] - 403s 3s/step - loss: 1.0452e-05 - accuracy: 1.0000 - val_loss: 1.2356e-05 - val_accu
racy: 1.0000
Epoch 6/10
157/157 [==============================] - 405s 3s/step - loss: 7.0864e-06 - accuracy: 1.0000 - val_loss: 9.2751e-06 - val_accu
racy: 1.0000
Epoch 7/10
157/157 [==============================] - 398s 3s/step - loss: 5.1282e-06 - accuracy: 1.0000 - val_loss: 6.8062e-06 - val_accu
racy: 1.0000
Epoch 8/10
157/157 [==============================] - 409s 3s/step - loss: 3.8520e-06 - accuracy: 1.0000 - val_loss: 5.3335e-06 - val_accu
racy: 1.0000
Epoch 9/10
157/157 [==============================] - 411s 3s/step - loss: 2.9968e-06 - accuracy: 1.0000 - val_loss: 4.3884e-06 - val_accu
racy: 1.0000
Epoch 10/10
157/157 [==============================] - 399s 3s/step - loss: 2.3799e-06 - accuracy: 1.0000 - val_loss: 3.6500e-06 - val_accu
racy: 1.0000
```