```
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report, roc_auc_score, roc_curve, auc
import matplotlib.pyplot as plt
import numpy as np
import os
```

```
from google.colab import files
files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving medical-mnist.zip to medical-mnist.zip
```

```
!unzip medical-mnist.zip -d medical-mnist
```

Show hidden output

```
import os
DATA_DIR = "./medical-mnist"

IMG_SIZE = (224, 224)
BATCH_SIZE = 32

datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=15,
    zoom_range=0.1,
    horizontal_flip=True
)

train_gen = datagen.flow_from_directory(
    DATA_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    subset='training',
    class_mode='categorical'
)

val_gen = datagen.flow_from_directory(
    DATA_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    subset='validation',
    class_mode='categorical',
    shuffle=False
)
```

```
Found 47164 images belonging to 6 classes.
Found 11790 images belonging to 6 classes.
```

```
print("Num GPUs Available:", len(tf.config.list_physical_devices('GPU')))
tf.config.list_physical_devices()
```

```
Num GPUs Available: 1
[PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU'),
 PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(224,224,3))

# Freeze the base layers first
for layer in base_model.layers:
    layer.trainable = False

x = GlobalAveragePooling2D()(base_model.output)
x = Dropout(0.4)(x)
output = Dense(train_gen.num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)
```

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-3),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
```

Show hidden output

```
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=5
)
```

Show hidden output

```
# Training first layer first

for layer in base_model.layers[-100:]:  # unfreeze last 100 layers
    layer.trainable = True

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history_fine = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=3
)
```

Show hidden output

```
val_loss, val_acc = model.evaluate(val_gen)
print(f"Validation Accuracy: {val_acc*100:.2f}%")
```

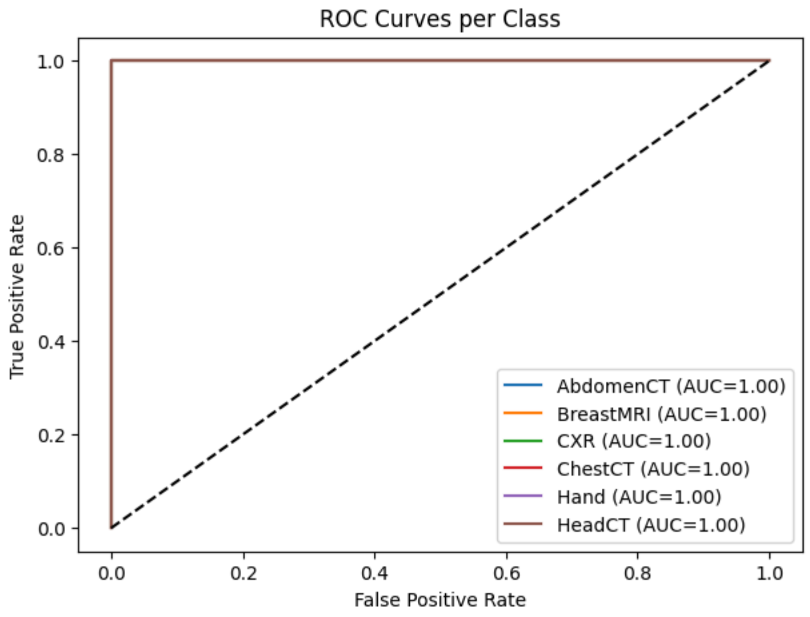
```
369/369 ━━━━━━━━━━━━━━━━━━━━ 137s 371ms/step - accuracy: 0.9999 - loss: 3.6757e-04
Validation Accuracy: 99.97%
```

```
y_true = val_gen.classes
y_pred = model.predict(val_gen)
y_score = np.array(y_pred)

fpr, tpr, roc_auc = {}, {}, {}
for i, c in enumerate(val_gen.class_indices):
    fpr[c], tpr[c], _ = roc_curve((y_true == i).astype(int), y_score[:, i])
    roc_auc[c] = auc(fpr[c], tpr[c])

plt.figure(figsize=(7,5))
for c in roc_auc:
    plt.plot(fpr[c], tpr[c], label=f"{c} (AUC={roc_auc[c]:.2f})")
plt.plot([0,1],[0,1],'k--')
plt.xlabel('False Positive Rate'); plt.ylabel('True Positive Rate')
plt.legend(); plt.title('ROC Curves per Class')
plt.show()
```

```
369/369 ━━━━━━━━━━━━━━━━━━━━ 144s 375ms/step
```



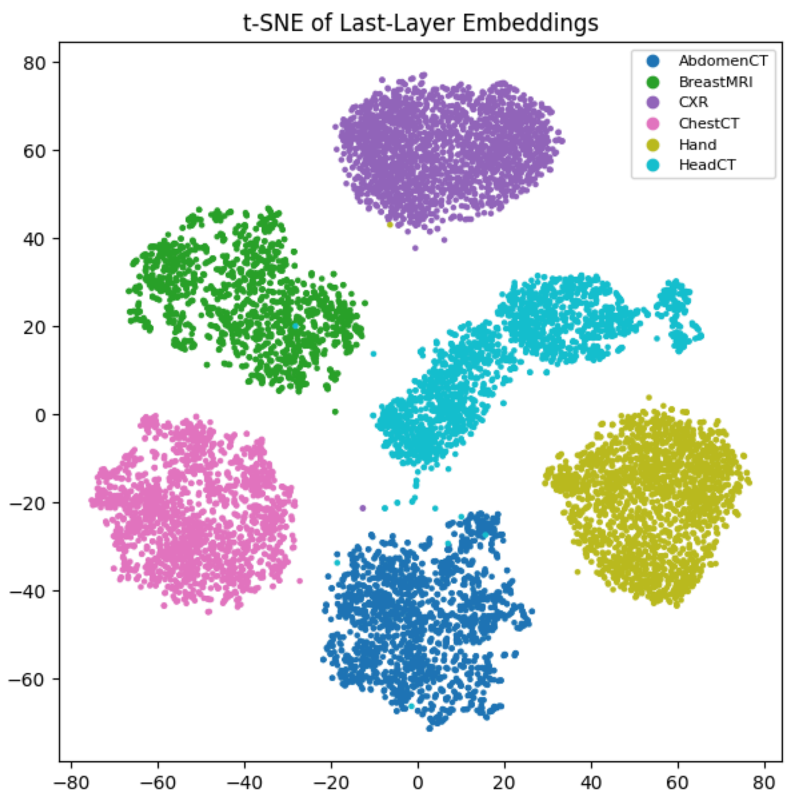
```
from sklearn.manifold import TSNE

# Extract features from the penultimate layer
feature_model = Model(inputs=model.input, outputs=model.layers[-2].output)
features = feature_model.predict(val_gen)
labels = y_true

tsne = TSNE(n_components=2, perplexity=30, random_state=42)
emb = tsne.fit_transform(features)

plt.figure(figsize=(7,7))
scatter = plt.scatter(emb[:,0], emb[:,1], c=labels, cmap='tab10', s=5)
plt.legend(handles=scatter.legend_elements()[0], labels=val_gen.class_indices.keys(), fontsize=8)
plt.title('t-SNE of Last-Layer Embeddings')
plt.show()
```

```
369/369 ━━━━━━━━━━━━━━━━━━━━ 145s 376ms/step
```



Writeup:

1. Model Selection

I chose InceptionV3, becuase_

- It was one of the first ImageNet architectures successfully adapted to medical imaging tasks.
- It trains efficiently on a Google Colab GPU.
- Keras provides a simple and stable API for transfer learning and fine-tuning.

---

2. Dataset Selection

I used the **Medical MNIST** dataset from Kaggle. It contains 10,000 grayscale medical images across six classes:

- Abdomen CT
- Breast MRI
- Chest X-ray
- Chest CT
- Hand X-ray
- Head CT

Reasons for choosing this dataset:

- Small size and quick to train.
- Balanced classes.
- Well-suited for transfer learning demonstrations.
- Simple directory structure that works directly with Keras data generators.

All images were resized to **224×224×3** to match InceptionV3 input requirements.

---

3. Methodology

Preprocessing

- Loaded dataset using `ImageDataGenerator`.

- Rescaled pixel intensities to [0, 1].
- Applied data augmentation (rotation, zoom, horizontal flip).
- Used an 80/20 training/validation split.

Model Architecture

- Loaded InceptionV3 with `weights="imagenet"` and `include_top=False`