


POST LAB DL – Sumit Sonar

 3D_image_classification

File Edit View Insert Runtime Tools Help Cannot save changes

+ Code + Text Copy to Drive

✓ 4s

[1]

```
import os
import zipfile
import numpy as np
import tensorflow as tf # for data preprocessing

import keras
from keras import layers
```

✓ 34s

[2]

```
# Download url of normal CT scans.
url = "https://github.com/hasibzunair/3D-image-classification-tutorial/releases/download/v0.2/CT-0.zip"
filename = os.path.join(os.getcwd(), "CT-0.zip")
keras.utils.get_file(filename, url)

# Download url of abnormal CT scans.
url = "https://github.com/hasibzunair/3D-image-classification-tutorial/releases/download/v0.2/CT-23.zip"
filename = os.path.join(os.getcwd(), "CT-23.zip")
keras.utils.get_file(filename, url)

# Make a directory to store the data.
os.makedirs("MosMedData")

# Unzip data in the newly created directory.
with zipfile.ZipFile("CT-0.zip", "r") as z_fp:
    z_fp.extractall("./MosMedData/")

with zipfile.ZipFile("CT-23.zip", "r") as z_fp:
    z_fp.extractall("./MosMedData/")
```

Downloading data from <https://github.com/hasibzunair/3D-image-classification-tutorial/releases/download/v0.2/CT-0.zip>
1065471431/1065471431 [=====] - 7s 0us/step
Downloading data from <https://github.com/hasibzunair/3D-image-classification-tutorial/releases/download/v0.2/CT-23.zip>
1045162547/1045162547 [=====] - 11s 0us/step

✓ 2s completed at 1:12 PM

✓ 0s

[3]

```
import nibabel as nib
from scipy import ndimage

def read_nifti_file(filepath):
    """Read and load volume"""
    # Read file
    scan = nib.load(filepath)
    # Get raw data
    scan = scan.get_fdata()
    return scan

def normalize(volume):
    """Normalize the volume"""
    min = -1000
    max = 400
    volume[volume < min] = min
    volume[volume > max] = max
    volume = (volume - min) / (max - min)
    volume = volume.astype("float32")
    return volume

def resize_volume(img):
```

✓ 0s

[4]

```
# Folder "CT-0" consist of CT scans having normal lung tissue,
# no CT-signs of viral pneumonia.
normal_scan_paths = [
    os.path.join(os.getcwd(), "MosMedData/CT-0", x)
    for x in os.listdir("MosMedData/CT-0")
]

# Folder "CT-23" consist of CT scans having several ground-glass opacifications,
# involvement of lung parenchyma.
abnormal_scan_paths = [
    os.path.join(os.getcwd(), "MosMedData/CT-23", x)
    for x in os.listdir("MosMedData/CT-23")
]

print("CT scans with normal lung tissue: " + str(len(normal_scan_paths)))
print("CT scans with abnormal lung tissue: " + str(len(abnormal_scan_paths)))
```

CT scans with normal lung tissue: 100
CT scans with abnormal lung tissue: 100

✓ 2s completed at 1:12 PM

POST LAB DL – Sumit Sonar

CO

3D_image_classification

File Edit View Insert Runtime Tools Help Cannot save changes

Share

+ Code + Text Copy to Drive

✓ T4 RAM Disk

7m

[5]

```
# Read and process the scans.
# Each scan is resized across height, width, and depth and rescaled.
abnormal_scans = np.array([process_scan(path) for path in abnormal_scan_paths])
normal_scans = np.array([process_scan(path) for path in normal_scan_paths])

# For the CT scans having presence of viral pneumonia
# assign 1, for the normal ones assign 0.
abnormal_labels = np.array([1 for _ in range(len(abnormal_scans))])
normal_labels = np.array([0 for _ in range(len(normal_scans))])

# Split data in the ratio 70-30 for training and validation.
x_train = np.concatenate((abnormal_scans[:70], normal_scans[:70]), axis=0)
y_train = np.concatenate((abnormal_labels[:70], normal_labels[:70]), axis=0)
x_val = np.concatenate((abnormal_scans[70:], normal_scans[70:]), axis=0)
y_val = np.concatenate((abnormal_labels[70:], normal_labels[70:]), axis=0)
print(
    "Number of samples in train and validation are %d and %d."
    % (x_train.shape[0], x_val.shape[0])
)
```

Number of samples in train and validation are 140 and 60.

0s

[6]

```
import random

from scipy import ndimage

def rotate(volume):
    """Rotate the volume by a few degrees"""

    def scipy_rotate(volume):
```

✓ 2s completed at 1:12 PM

CO

3D_image_classification

File Edit View Insert Runtime Tools Help Cannot save changes

Share

+ Code + Text Copy to Drive

✓ T4 RAM Disk

3s

[7]

```
batch_size = 2
# Augment the on the fly during training.
train_dataset = (
    train_loader.shuffle(len(x_train))
    .map(train_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
# Only rescale.
validation_dataset = (
    validation_loader.shuffle(len(x_val))
    .map(validation_preprocessing)
    .batch(batch_size)
    .prefetch(2)
)
```

3s

[8]

```
import matplotlib.pyplot as plt

data = train_dataset.take(1)
images, labels = list(data)[0]
images = images.numpy()
image = images[0]
print("Dimension of the CT scan is:", image.shape)
plt.imshow(np.squeeze(image[:, :, 30]), cmap="gray")
```

Dimension of the CT scan is: (128, 128, 64, 1)
<matplotlib.image.AxesImage at 0x7c1758b24c70>

0

✓ 2s completed at 1:12 PM

CO

3D_image_classification

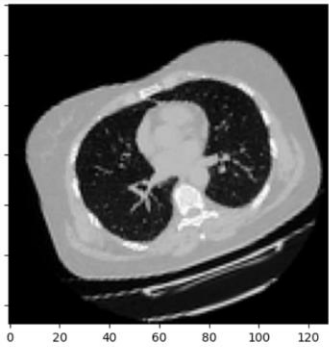
File Edit View Insert Runtime Tools Help Cannot save changes

Share

+ Code + Text Copy to Drive

✓ T4 RAM Disk

3s



3s

[9]

```
def plot_slices(num_rows, num_columns, width, height, data):
    """plot a montage of 20 CT slices"""
    data = np.rot90(np.array(data))
    data = np.transpose(data)
    data = np.reshape(data, (num_rows, num_columns, width, height))
    rows_data, columns_data = data.shape[0], data.shape[1]
    heights = [slc[0].shape[0] for slc in data]
```

✓ 2s completed at 1:12 PM

POST LAB DL – Sumit Sonar

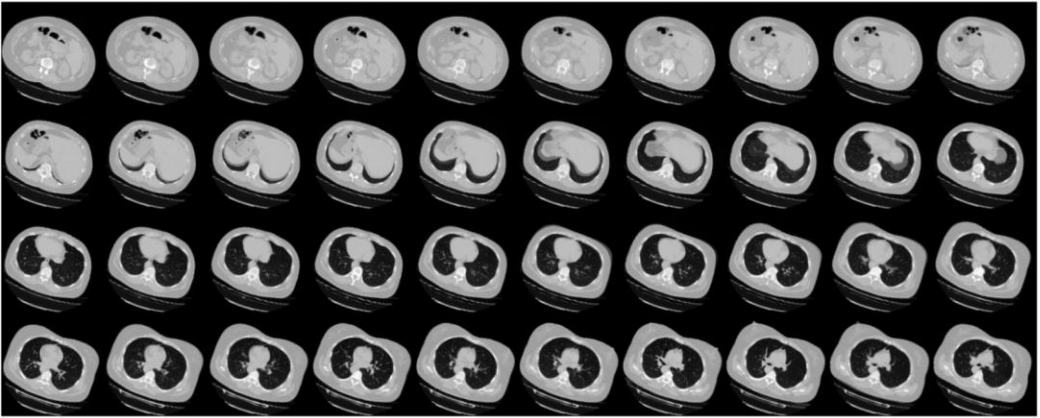
3D_image_classification

File Edit View Insert Runtime Tools Help Cannot save changes

+ Code + Text Copy to Drive

✓ T4 RAM Disk

```
# Visualize montage of slices.
# 4 rows and 10 columns for 100 slices of the CT scan.
plot_slices(4, 10, 128, 128, image[:, :, :40])
```



✓ 2s completed at 1:12 PM

3D_image_classification

File Edit View Insert Runtime Tools Help Cannot save changes

+ Code + Text Copy to Drive

✓ T4 RAM Disk

```
[10] def get_model(width=128, height=128, depth=64):
    """Build a 3D convolutional neural network model."""
    inputs = keras.Input((width, height, depth, 1))
    x = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(inputs)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)
    x = layers.Conv3D(filters=64, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)
    x = layers.Conv3D(filters=128, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)
    x = layers.Conv3D(filters=256, kernel_size=3, activation="relu")(x)
    x = layers.MaxPool3D(pool_size=2)(x)
    x = layers.BatchNormalization()(x)
    x = layers.GlobalAveragePooling3D()(x)
    x = layers.Dense(units=512, activation="relu")(x)
    x = layers.Dropout(0.3)(x)
    outputs = layers.Dense(units=1, activation="sigmoid")(x)
    # Define the model.
    model = keras.Model(inputs, outputs, name="3dcnn")
    return model
```

✓ 2s completed at 1:12 PM

3D_image_classification

File Edit View Insert Runtime Tools Help Cannot save changes

+ Code + Text Copy to Drive

✓ T4 RAM Disk

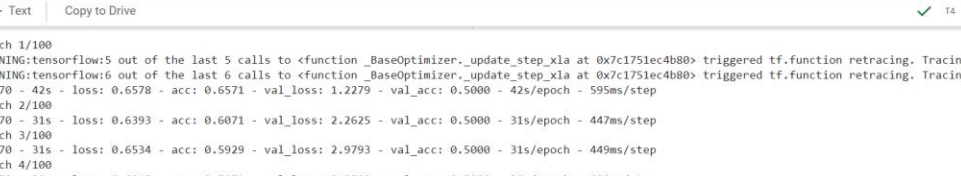
```
[10] Total params: 1352897 (5.16 MB)
Trainable params: 1351873 (5.16 MB)
Non-trainable params: 1024 (4.00 KB)
```

Train model

```
[11] # Compile model.
initial_learning_rate = 0.0001
lr_schedule = keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate, decay_steps=100000, decay_rate=0.96, staircase=True
)
model.compile(
    loss="binary_crossentropy",
    optimizer=keras.optimizers.Adam(learning_rate=lr_schedule),
    metrics=["acc"],
    run_eagerly=True,
)
# Define callbacks.
checkpoint_cb = keras.callbacks.ModelCheckpoint(
    "3d_image_classification.keras", save_best_only=True
)
early_stopping_cb = keras.callbacks.EarlyStopping(monitor="val_acc", patience=15)
# Train the model, doing validation at the end of each epoch
epochs = 100
model.fit(
    train_dataset,
```

✓ 2s completed at 1:12 PM

POST LAB DL – Sumit Sonar



The image shows a Google Colab interface. At the top, the title bar reads "3D_image_classification". Below it are tabs for "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". The "Runtime" tab is active, showing a "Copy to Drive" button. The main area contains a Jupyter notebook with a single cell. The cell's output shows a training log for 16 epochs. The log includes warnings about TensorFlow's BaseOptimizer and a summary of training metrics (loss, accuracy, validation loss, validation accuracy) for each epoch. The training time is 1:12 PM, and the total time taken for the cell is 2s.

```
[11] Epoch 1/100
WARNING:tensorflow:5 out of the last 5 calls to <function _BaseOptimizer._update_step_xla at 0x7c1751ec4b80> triggered tf.function retracing. Tracing is expensive and will significantly slow down your code. Please see https://www.tensorflow.org/api_guides/python/tracing for more details.
WARNING:tensorflow:6 out of the last 6 calls to <function _BaseOptimizer._update_step_xla at 0x7c1751ec4b80> triggered tf.function retracing. Tracing is expensive and will significantly slow down your code. Please see https://www.tensorflow.org/api_guides/python/tracing for more details.
70/70 - 42s - loss: 0.6578 - acc: 0.6571 - val_loss: 1.2279 - val_acc: 0.5000 - 42s/epoch - 595ms/step
Epoch 2/100
70/70 - 31s - loss: 0.6393 - acc: 0.6071 - val_loss: 2.2625 - val_acc: 0.5000 - 31s/epoch - 447ms/step
Epoch 3/100
70/70 - 31s - loss: 0.6534 - acc: 0.5929 - val_loss: 2.9793 - val_acc: 0.5000 - 31s/epoch - 449ms/step
Epoch 4/100
70/70 - 30s - loss: 0.6215 - acc: 0.7071 - val_loss: 3.0509 - val_acc: 0.5000 - 30s/epoch - 432ms/step
Epoch 5/100
70/70 - 29s - loss: 0.6490 - acc: 0.6571 - val_loss: 2.4660 - val_acc: 0.5000 - 29s/epoch - 420ms/step
Epoch 6/100
70/70 - 30s - loss: 0.6330 - acc: 0.6286 - val_loss: 1.9164 - val_acc: 0.5000 - 30s/epoch - 430ms/step
Epoch 7/100
70/70 - 29s - loss: 0.5965 - acc: 0.6929 - val_loss: 1.8847 - val_acc: 0.5000 - 29s/epoch - 419ms/step
Epoch 8/100
70/70 - 31s - loss: 0.6618 - acc: 0.5857 - val_loss: 0.9466 - val_acc: 0.5000 - 31s/epoch - 437ms/step
Epoch 9/100
70/70 - 30s - loss: 0.6102 - acc: 0.6714 - val_loss: 0.7172 - val_acc: 0.6000 - 30s/epoch - 432ms/step
Epoch 10/100
70/70 - 31s - loss: 0.5991 - acc: 0.6500 - val_loss: 0.6060 - val_acc: 0.6500 - 31s/epoch - 440ms/step
Epoch 11/100
70/70 - 30s - loss: 0.5790 - acc: 0.7000 - val_loss: 0.6044 - val_acc: 0.6500 - 30s/epoch - 429ms/step
Epoch 12/100
70/70 - 30s - loss: 0.5831 - acc: 0.7357 - val_loss: 0.9922 - val_acc: 0.5333 - 30s/epoch - 425ms/step
Epoch 13/100
70/70 - 29s - loss: 0.5783 - acc: 0.6929 - val_loss: 0.7319 - val_acc: 0.6500 - 29s/epoch - 415ms/step
Epoch 14/100
70/70 - 30s - loss: 0.6074 - acc: 0.6286 - val_loss: 1.4215 - val_acc: 0.5000 - 30s/epoch - 432ms/step
Epoch 15/100
70/70 - 29s - loss: 0.5655 - acc: 0.6714 - val_loss: 0.9379 - val_acc: 0.5167 - 29s/epoch - 415ms/step
Epoch 16/100
```

3D_image_classification

File Edit View Insert Runtime Tools Help
Cannot save changes

Share
Settings
Profile

+ Code + Text Copy to Drive

✓ T4 RAM 12.7GB Disk 100GB

```

epoch > 100
70/70 - 29s - loss: 0.2744 - acc: 0.8643 - val_loss: 1.0135 - val_acc: 0.6833 - 29s/epoch - 416ms/step
Epoch 52/100
70/70 - 29s - loss: 0.3454 - acc: 0.8500 - val_loss: 1.1675 - val_acc: 0.5500 - 29s/epoch - 414ms/step
<keras.src.callbacks.History at 0x7c1751f499c0>

[12] fig, ax = plt.subplots(1, 2, figsize=(20, 3))
ax = ax.ravel()


for i, metric in enumerate(["acc", "loss"]):
    ax[i].plot(model.history.history[metric])
    ax[i].plot(model.history.history["val_" + metric])
    ax[i].set_title("Model {}".format(metric))
    ax[i].set_xlabel("epochs")
    ax[i].set_ylabel(metric)
    ax[i].legend(["train", "val"])

```

Model acc

Model loss

✓ 2s completed at 1:12PM



The screenshot shows a Google Colab notebook titled "3D_image_classification". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with icons for file operations and sharing. The notebook content consists of a single code cell with the following Python code:

```
# load best weights.
model.load_weights("3d_image_classification.keras")
prediction = model.predict(np.expand_dims(x_val[0], axis=0))[0]
scores = [1 - prediction[0], prediction[0]]

class_names = ["normal", "abnormal"]
for score, name in zip(scores, class_names):
    print(
        "This model is %.2f percent confident that CT scan is %s"
        % ((100 * score), name)
    )

--NORMAL--

1/1 [=====] - 1s 694ms/step
This model is 36.31 percent confident that CT scan is normal
This model is 63.69 percent confident that CT scan is abnormal
```

The output of the code cell shows the model's confidence for a CT scan image. It prints "This model is 36.31 percent confident that CT scan is normal" and "This model is 63.69 percent confident that CT scan is abnormal".