```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
Mounted at /content/drive
```

```
!pip install -q libtiff
!pip install -q tqdm
```

```
# import the necessary packages
import matplotlib.pyplot as plt
import sys
import os
from pathlib import Path
from libtiff import TIFF
import numpy as np
from typing import List
from tqdm.notebook import tqdm
from time import sleep
from PIL import Image
import IPython.display
import tensorflow as tf
from sklearn.metrics import balanced_accuracy_score
from keras.optimizers import SGD, RMSprop, Adam
from keras import optimizers, Model, callbacks
from sklearn.metrics import mean_squared_error
from keras.utils import plot_model
from sklearn.preprocessing import StandardScaler, MinMaxScaler
```

Класс Dataset

Предназначен для работы с наборами данных, хранящихся на Google Drive, обеспечивает чтение изображений и соответствующих меток, а также формирование пакетов (батчей).

```
class Dataset:
    def __init__(self, name, gdrive_dir):
        self.name = name
        self.is_loaded = False
        p = Path("/content/drive/MyDrive/" + gdrive_dir + name + '.npz')
        if p.exists():
            print(f'[INFO] Loading dataset {self.name} from npz...')
            np_obj = np.load(str(p))
            self.images = np_obj['data']
            self.labels = np_obj['labels']
            self.n_files = self.images.shape[0]
```

```
            self.is_loaded = True
            print(f'[INFO] Done. Dataset {name} consists of {self.n_files}

    def image(self, i):
        # read i-th image in dataset and return it as numpy array
        if self.is_loaded:
            return self.images[i, :, :, :]

    def images_seq(self, n=None):
        # sequential access to images inside dataset (is needed for testing
        for i in range(self.n_files if not n else n):
            yield self.image(i)

    def random_image_with_label(self):
        # get random image with label from dataset
        i = np.random.randint(self.n_files)
        return self.image(i), self.labels[i]

    def random_batch_with_labels(self, n):
        # create random batch of images with labels (is needed for training
        indices = np.random.choice(self.n_files, n)
        imgs = []
        for i in indices:
            imgs.append(self.image(i))
        logits = np.array([self.labels[i] for i in indices])
        return np.array(imgs), logits

    def image_with_label(self, i: int):
        # return i-th image with label from dataset
        return self.image(i), self.labels[i]
```

Класс Model

Предназначен для работы с нейросетью. Обеспечивает загрузку, обучение, тестирование и сохранение нейронной сети, отображение дополнительной информации об алгоритме обучения.

```
class Model:

    def __init__(self):
        chanDim = -1
        self.model = tf.keras.models.Sequential(
            [tf.keras.layers.Conv2D(8, (3, 3), padding="same", activation='
          input_shape=(224, 224, 3)),
```

```
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(64, (3, 3), padding="same" , activation='r
        tf.keras.layers.MaxPooling2D(data_format='channels_last'),

        tf.keras.layers.Conv2D(128, (3, 3), padding="same" , activation='
        tf.keras.layers.MaxPooling2D(data_format='channels_last'),

        tf.keras.layers.Conv2D(256, (3, 3), padding="same", activation='r
        tf.keras.layers.MaxPooling2D(data_format='channels_last'),

        tf.keras.layers.Conv2D(512, (3, 3), padding="same", activation='r
        tf.keras.layers.MaxPooling2D(data_format='channels_last'),
        tf.keras.layers.Dropout(0.5),

        tf.keras.layers.Flatten(),

        tf.keras.layers.Dense(1000, kernel_regularizer=tf.keras.regulariz
        tf.keras.layers.Activation("relu"),
        tf.keras.layers.Dropout(0.3),

        tf.keras.layers.Dense(512, kernel_regularizer=tf.keras.regularize
        tf.keras.layers.Activation("relu"),
        tf.keras.layers.Dropout(0.3),

        tf.keras.layers.Dense(256, kernel_regularizer=tf.keras.regularize
        tf.keras.layers.Activation("relu"),
        tf.keras.layers.Dropout(0.3),

        tf.keras.layers.Dense(9),
        tf.keras.layers.Activation("softmax")
        ])

      self.save_dir = "/content/drive/MyDrive/nn_tasks/"


    def save_model(self, name: str):
      # save model to SAVE_DIR folder on gdrive with name 'name'
      #self.save_dir += name
      self.model.save(self.save_dir + "model-res/" + name + "_model.h5")
      self.model.save_weights(self.save_dir + "model-res/" + name + "_mod
      plot_model(self.model, to_file=self.save_dir + "model-res/" + name


    def load(self, name: str, is_checkpoint=False):
      p1 = Path(self.save_dir + "model-res/"  + name + ".h5")
      p2 = Path(self.save_dir + "model-res/best_checkpoint/")
```

```
        self.model = None
        if is_checkpoint and p2.exists():
          self.model = tf.keras.models.load_model(p2)
        elif p1.exists():
          self.model = tf.keras.models.load_model(p1)
        return model


    def train(self, dataset: Dataset, is_showing_history=True):
        # you can add some plots for better visualization,
        self.INIT_LR = 6e-4
        self.EPOCHS = 100
        self.BS = 64

        print(f"[INFO] split and shuffle the data...")
        d_train_images_1, d_train_labels_1 = dataset.random_batch_with_labe
        n_train = int(dataset.n_files * 0.92)

        d_train_images = d_train_images_1[0:n_train]
        d_train_labels = d_train_labels_1[0:n_train]
        #LBL1
        d_val_images = d_train_images_1[n_train:dataset.n_files]
        d_val_labels = d_train_labels_1[n_train:dataset.n_files]

        #scaler = MinMaxScaler()
        #X = scaler.fit_transform(d_train_images)
        #X_val = scaler.fit_transform(d_val_images)

        optA = Adam(learning_rate=self.INIT_LR)
        self.model.compile(loss=tf.keras.losses.SparseCategoricalCrossentro
          metrics=["accuracy"])

        #LBL2
        checkpoint = callbacks.ModelCheckpoint(self.save_dir + "model-res/b
    save_weights_only=False, save_freq='epoch') #mode='auto',
        callbacks_list = [checkpoint]
        #LBL3
        print(f"[INFO] training network...")
        self.History = self.model.fit(d_train_images, d_train_labels, valid
        print(f'[INFO] training done')
        #LBL4
        self.show_history(str(n_train))


    def test_on_dataset(self, dataset: Dataset, limit=None):
        # you can upgrade this code if you want to speed up testing using b
        predictions = []
```

```
        n = dataset.n_files if not limit else int(dataset.n_files * limit)
        for img in tqdm(dataset.images_seq(n), total=n):
            predictions.append(self.test_on_image(img))
        return predictions


    def test_on_image(self, img):
        # todo: replace this code
        prediction = self.model.predict(np.array(img))
        return np.argmax(prediction)


    def evaluate_model(self, dataset: Dataset):
        print("[INFO] evaluating network...")
        result = self.model.evaluate(dataset.images, dataset.labels, batch_
        print('test loss, test acc:', result)

        predictions = self.model.predict(dataset.images)
        pred = []
        print("MSE  (test_y, predictions)")
        for i in range(0, len(dataset.labels)):
          pred.append(predictions[i].max())
        print(mean_squared_error(dataset.labels, pred, squared=False))


    def show_history(self, size):
        N = np.arange(0, self.EPOCHS)
        plt.style.use("ggplot")
        plt.figure()
        plt.plot(N, self.History.history["loss"], label="train_loss")
        plt.plot(N, self.History.history["accuracy"], label="train_acc")
        plt.plot(N, self.History.history["val_loss"], label="test_loss")
        plt.plot(N, self.History.history["val_accuracy"], label="test_acc")
        plt.title("Training Los")
        plt.xlabel("Epoch #")
        plt.ylabel("Loss, accuracy")
        plt.legend()
        plt.savefig(self.save_dir + "model-res/history_training_on_" + size



PROJECT_DIR = "nn_tasks/"


EVALUATE_ONLY = False
TEST_ON_LARGE_DATASET = True
TISSUE_CLASSES = ('ADI', 'BACK', 'DEB', 'LYM', 'MUC', 'MUS', 'NORM', 'STR',
```

```
TISSUE_CLASSES = ('ADI', 'BACK', 'DEB', 'LYM', 'MUC', 'MUS', 'NORM', 'STR',
```

```
dataset_test = Dataset('test', PROJECT_DIR)
dataset_train = Dataset('train', PROJECT_DIR)
```

```
[INFO] Loading dataset test from npz...
[INFO] Done. Dataset test consists of 4500 images.
[INFO] Loading dataset train from npz...
[INFO] Done. Dataset train consists of 18000 images.
```

```
model = Model()
if not EVALUATE_ONLY:
    model.train(dataset_train, is_showing_history=True)
    model.save_model('best')
else:
    model.load('best')
```

```
[INFO] split and shuffle the data...
[INFO] training network...
Epoch 1/100
259/259 [==============================] - 25s 94ms/step - loss: 25.2997 - accura

Epoch 00001: val_accuracy improved from -inf to 0.42361, saving model to /conten
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 2/100
259/259 [==============================] - 25s 96ms/step - loss: 9.3074 - accura

Epoch 00002: val_accuracy improved from 0.42361 to 0.62153, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 3/100
259/259 [==============================] - 24s 94ms/step - loss: 5.1818 - accura

Epoch 00003: val_accuracy did not improve from 0.62153
Epoch 4/100
259/259 [==============================] - 24s 94ms/step - loss: 3.4037 - accura

Epoch 00004: val_accuracy improved from 0.62153 to 0.70139, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 5/100
259/259 [==============================] - 24s 94ms/step - loss: 2.4769 - accura

Epoch 00005: val_accuracy improved from 0.70139 to 0.77431, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 6/100
259/259 [==============================] - 24s 94ms/step - loss: 1.9708 - accura

Epoch 00006: val_accuracy improved from 0.77431 to 0.80625, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 7/100
259/259 [==============================] - 24s 94ms/step - loss: 1.6131 - accura

Epoch 00007: val_accuracy improved from 0.80625 to 0.81250, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 8/100
259/259 [==============================] - 24s 94ms/step - loss: 1.3739 - accura

Epoch 00008: val_accuracy improved from 0.81250 to 0.85556, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 9/100
259/259 [==============================] - 25s 95ms/step - loss: 1.1241 - accura

Epoch 00009: val_accuracy did not improve from 0.85556
Epoch 10/100
259/259 [==============================] - 24s 94ms/step - loss: 1.0445 - accura

Epoch 00010: val_accuracy improved from 0.85556 to 0.89444, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 11/100
259/259 [==============================] - 24s 94ms/step - loss: 0.9527 - accura

Epoch 00011: val_accuracy improved from 0.89444 to 0.91736, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 12/100
259/259 [==============================] - 25s 95ms/step - loss: 0.8349 - accura
```

```
Epoch 00012: val_accuracy did not improve from 0.91736
Epoch 13/100
259/259 [==============================] - 24s 94ms/step - loss: 0.7818 - accura

Epoch 00013: val_accuracy did not improve from 0.91736
Epoch 14/100
259/259 [==============================] - 24s 94ms/step - loss: 0.7302 - accura

Epoch 00014: val_accuracy improved from 0.91736 to 0.93681, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 15/100
259/259 [==============================] - 24s 94ms/step - loss: 0.6678 - accura

Epoch 00015: val_accuracy did not improve from 0.93681
Epoch 16/100
259/259 [==============================] - 24s 94ms/step - loss: 0.6698 - accura

Epoch 00016: val_accuracy did not improve from 0.93681
Epoch 17/100
259/259 [==============================] - 24s 95ms/step - loss: 0.6749 - accura

Epoch 00017: val_accuracy did not improve from 0.93681
Epoch 18/100
259/259 [==============================] - 24s 94ms/step - loss: 0.6094 - accura

Epoch 00018: val_accuracy did not improve from 0.93681
Epoch 19/100
259/259 [==============================] - 24s 94ms/step - loss: 0.6396 - accura

Epoch 00019: val_accuracy improved from 0.93681 to 0.94097, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 20/100
259/259 [==============================] - 24s 94ms/step - loss: 0.5821 - accura

Epoch 00020: val_accuracy did not improve from 0.94097
Epoch 21/100
259/259 [==============================] - 24s 94ms/step - loss: 0.5542 - accura

Epoch 00021: val_accuracy improved from 0.94097 to 0.94306, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 22/100
259/259 [==============================] - 24s 94ms/step - loss: 0.5652 - accura

Epoch 00022: val_accuracy improved from 0.94306 to 0.96111, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 23/100
259/259 [==============================] - 24s 94ms/step - loss: 0.5099 - accura

Epoch 00023: val_accuracy did not improve from 0.96111
Epoch 24/100
259/259 [==============================] - 24s 94ms/step - loss: 0.5439 - accura

Epoch 00024: val_accuracy did not improve from 0.96111
Epoch 25/100
259/259 [==============================] - 24s 94ms/step - loss: 0.5219 - accura

Epoch 00025: val_accuracy improved from 0.96111 to 0.96458, saving model to /con
```

```
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 26/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4968 - accura

Epoch 00026: val_accuracy did not improve from 0.96458
Epoch 27/100
259/259 [==============================] - 24s 94ms/step - loss: 0.5384 - accura

Epoch 00027: val_accuracy did not improve from 0.96458
Epoch 28/100
259/259 [==============================] - 24s 94ms/step - loss: 0.5284 - accura

Epoch 00028: val_accuracy did not improve from 0.96458
Epoch 29/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4781 - accura

Epoch 00029: val_accuracy did not improve from 0.96458
Epoch 30/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4540 - accura

Epoch 00030: val_accuracy improved from 0.96458 to 0.96875, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 31/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4485 - accura

Epoch 00031: val_accuracy did not improve from 0.96875
Epoch 32/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4749 - accura

Epoch 00032: val_accuracy did not improve from 0.96875
Epoch 33/100
259/259 [==============================] - 24s 94ms/step - loss: 0.5126 - accura

Epoch 00033: val_accuracy did not improve from 0.96875
Epoch 34/100
259/259 [==============================] - 24s 93ms/step - loss: 0.4417 - accura

Epoch 00034: val_accuracy did not improve from 0.96875
Epoch 35/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4410 - accura

Epoch 00035: val_accuracy did not improve from 0.96875
Epoch 36/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4482 - accura

Epoch 00036: val_accuracy did not improve from 0.96875
Epoch 37/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4558 - accura

Epoch 00037: val_accuracy did not improve from 0.96875
Epoch 38/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4198 - accura

Epoch 00038: val_accuracy did not improve from 0.96875
Epoch 39/100
259/259 [==============================] - 24s 93ms/step - loss: 0.4046 - accura

Epoch 00039: val_accuracy did not improve from 0.96875
```

```
Epoch 00039: val_accuracy did not improve from 0.96875
Epoch 40/100
259/259 [==============================] - 24s 93ms/step - loss: 0.4267 - accura

Epoch 00040: val_accuracy did not improve from 0.96875
Epoch 41/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3782 - accura

Epoch 00041: val_accuracy did not improve from 0.96875
Epoch 42/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4626 - accura

Epoch 00042: val_accuracy did not improve from 0.96875
Epoch 43/100
259/259 [==============================] - 24s 93ms/step - loss: 0.4053 - accura

Epoch 00043: val_accuracy improved from 0.96875 to 0.97500, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 44/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4063 - accura

Epoch 00044: val_accuracy did not improve from 0.97500
Epoch 45/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3764 - accura

Epoch 00045: val_accuracy did not improve from 0.97500
Epoch 46/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3950 - accura

Epoch 00046: val_accuracy did not improve from 0.97500
Epoch 47/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3843 - accura

Epoch 00047: val_accuracy did not improve from 0.97500
Epoch 48/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3664 - accura

Epoch 00048: val_accuracy improved from 0.97500 to 0.97917, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 49/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3900 - accura

Epoch 00049: val_accuracy did not improve from 0.97917
Epoch 50/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3849 - accura

Epoch 00050: val_accuracy did not improve from 0.97917
Epoch 51/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3485 - accura

Epoch 00051: val_accuracy did not improve from 0.97917
Epoch 52/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3859 - accura

Epoch 00052: val_accuracy did not improve from 0.97917
Epoch 53/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3480 - accura

Epoch 00053: val_accuracy did not improve from 0.97917
```

```
Epoch 00053: val_accuracy did not improve from 0.97917
Epoch 54/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3639 - accura

Epoch 00054: val_accuracy did not improve from 0.97917
Epoch 55/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3472 - accura

Epoch 00055: val_accuracy did not improve from 0.97917
Epoch 56/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3575 - accura

Epoch 00056: val_accuracy did not improve from 0.97917
Epoch 57/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3480 - accura

Epoch 00057: val_accuracy improved from 0.97917 to 0.98472, saving model to /con
INFO:tensorflow:Assets written to: /content/drive/MyDrive/nn_tasks/model-res/bes
Epoch 58/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3300 - accura

Epoch 00058: val_accuracy did not improve from 0.98472
Epoch 59/100
259/259 [==============================] - 24s 94ms/step - loss: 0.4200 - accura

Epoch 00059: val_accuracy did not improve from 0.98472
Epoch 60/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3410 - accura

Epoch 00060: val_accuracy did not improve from 0.98472
Epoch 61/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3187 - accura

Epoch 00061: val_accuracy did not improve from 0.98472
Epoch 62/100
259/259 [==============================] - 25s 95ms/step - loss: 0.3049 - accura

Epoch 00062: val_accuracy did not improve from 0.98472
Epoch 63/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3869 - accura

Epoch 00063: val_accuracy did not improve from 0.98472
Epoch 64/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3561 - accura

Epoch 00064: val_accuracy did not improve from 0.98472
Epoch 65/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3555 - accura

Epoch 00065: val_accuracy did not improve from 0.98472
Epoch 66/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3504 - accura

Epoch 00066: val_accuracy did not improve from 0.98472
Epoch 67/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3264 - accura

Epoch 00067: val_accuracy did not improve from 0.98472
```

```
Epoch 68/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3738 - accura

Epoch 00068: val_accuracy did not improve from 0.98472
Epoch 69/100
259/259 [==============================] - 24s 93ms/step - loss: 0.2958 - accura

Epoch 00069: val_accuracy did not improve from 0.98472
Epoch 70/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3527 - accura

Epoch 00070: val_accuracy did not improve from 0.98472
Epoch 71/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3519 - accura

Epoch 00071: val_accuracy did not improve from 0.98472
Epoch 72/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3603 - accura

Epoch 00072: val_accuracy did not improve from 0.98472
Epoch 73/100
259/259 [==============================] - 24s 93ms/step - loss: 0.2979 - accura

Epoch 00073: val_accuracy did not improve from 0.98472
Epoch 74/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3588 - accura

Epoch 00074: val_accuracy did not improve from 0.98472
Epoch 75/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3222 - accura

Epoch 00075: val_accuracy did not improve from 0.98472
Epoch 76/100
259/259 [==============================] - 24s 93ms/step - loss: 0.2922 - accura

Epoch 00076: val_accuracy did not improve from 0.98472
Epoch 77/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3195 - accura

Epoch 00077: val_accuracy did not improve from 0.98472
Epoch 78/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3337 - accura

Epoch 00078: val_accuracy did not improve from 0.98472
Epoch 79/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3145 - accura

Epoch 00079: val_accuracy did not improve from 0.98472
Epoch 80/100
259/259 [==============================] - 24s 93ms/step - loss: 0.2642 - accura

Epoch 00080: val_accuracy did not improve from 0.98472
Epoch 81/100
259/259 [==============================] - 24s 93ms/step - loss: 0.2776 - accura

Epoch 00081: val_accuracy did not improve from 0.98472
Epoch 82/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3986 - accura
```

```
Epoch 00082: val_accuracy did not improve from 0.98472
Epoch 83/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3309 - accura

Epoch 00083: val_accuracy did not improve from 0.98472
Epoch 84/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3282 - accura

Epoch 00084: val_accuracy did not improve from 0.98472
Epoch 85/100
259/259 [==============================] - 24s 94ms/step - loss: 0.3506 - accura

Epoch 00085: val_accuracy did not improve from 0.98472
Epoch 86/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3179 - accura

Epoch 00086: val_accuracy did not improve from 0.98472
Epoch 87/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3464 - accura

Epoch 00087: val_accuracy did not improve from 0.98472
Epoch 88/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3419 - accura

Epoch 00088: val_accuracy did not improve from 0.98472
Epoch 89/100
259/259 [==============================] - 24s 93ms/step - loss: 0.2952 - accura

Epoch 00089: val_accuracy did not improve from 0.98472
Epoch 90/100
259/259 [==============================] - 24s 93ms/step - loss: 0.2985 - accura

Epoch 00090: val_accuracy did not improve from 0.98472
Epoch 91/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3108 - accura

Epoch 00091: val_accuracy did not improve from 0.98472
Epoch 92/100
259/259 [==============================] - 24s 93ms/step - loss: 0.2658 - accura

Epoch 00092: val_accuracy did not improve from 0.98472
Epoch 93/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3227 - accura

Epoch 00093: val_accuracy did not improve from 0.98472
Epoch 94/100
259/259 [==============================] - 24s 93ms/step - loss: 0.2858 - accura

Epoch 00094: val_accuracy did not improve from 0.98472
Epoch 95/100
259/259 [==============================] - 24s 93ms/step - loss: 0.3642 - accura

Epoch 00095: val_accuracy did not improve from 0.98472
Epoch 96/100
```

```
model.evaluate_model(dataset_test)
```

```
[INFO] evaluating network...
71/71 [==============================] - 3s 38ms/step - loss: 0.4414 - accuracy:
test loss, test acc: [0.4413629472255707, 0.9511111378669739]
MSE  (test_y, predictions)
3.9812546
```

```
img, lbl = dataset_test.random_image_with_label()
pred = model.test_on_image([img])
```

```
TISSUE_CLASSES[pred], pred, TISSUE_CLASSES[lbl], lbl
```

```
('DEB', 2, 'DEB', 2)
```

```
Epoch 00100: val accuracy did not improve from 0.98472
```

```
#LOADING MODEL FROM BEST CHECKPOINT
mm = model.load("best_model", is_checkpoint=True)
```

```
mm.evaluate_model(dataset_test)
```

```
[INFO] evaluating network...
71/71 [==============================] - 2s 24ms/step - loss: 0.4487 - accuracy:
test loss, test acc: [0.44866058230400085, 0.9546666741371155]
MSE  (test_y, predictions)
3.9859054
```

```
img, lbl = dataset_test.random_image_with_label()
pred = model.test_on_image([img])
```

```
TISSUE_CLASSES[pred], pred, TISSUE_CLASSES[lbl], lbl
```

```
('STR', 7, 'STR', 7)
```

```
img, lbl = dataset_train.random_image_with_label()
print()
print(f'Got numpy array of shape {img.shape}, and label with code {lbl}.')
print(f'Label code corresponds to {TISSUE_CLASSES[lbl]} class.')

pil_img = Image.fromarray(img)
IPython.display.display(pil_img)
```

```
Got numpy array of shape (224, 224, 3), and label with code 2.
Label code corresponds to DEB class.
```