def load\_image\_into\_numpy\_array(path):

    """Load an image from file into a numpy array.

    Puts image into numpy array to feed into tensorflow graph.

    Note that by convention we put it into a numpy array with shape

    (height, width, channels), where channels=3 for RGB.

    Args:

    path: a file path.

    Returns:

    uint8 numpy array with shape (img\_height, img\_width, 3)

    """

    img\_data = tf.io.gfile.GFile(path, 'rb').read()

    image = Image.open(BytesIO(img\_data))

    (im\_width, im\_height) = image.size

    return np.array(image.getdata()).reshape(

        (im\_height, im\_width, 3)).astype(np.uint8)

def plot\_detections(image\_np,

                    boxes,

                    classes,

                    scores,

                    category\_index,

                    figsize=(12, 16),

                    image\_name=None):

    """Wrapper function to visualize detections.

    Args:

    image\_np: uint8 numpy array with shape (img\_height, img\_width, 3)

    boxes: a numpy array of shape [N, 4]

    classes: a numpy array of shape [N]. Note that class indices are 1-based,

          and match the keys in the label map.

    scores: a numpy array of shape [N] or None.  If scores=None, then

          this function assumes that the boxes to be plotted are groundtruth

          boxes and plot all boxes as black with no classes or scores.

    category\_index: a dict containing category dictionaries (each holding

          category index `id` and category name `name`) keyed by category indices.

    figsize: size for the figure.

    image\_name: a name for the image file.

    """

    image\_np\_with\_annotations = image\_np.copy()

    viz\_utils.visualize\_boxes\_and\_labels\_on\_image\_array(

        image\_np\_with\_annotations,

        boxes,

        classes,

        scores,

        category\_index,

        use\_normalized\_coordinates=True,

        min\_score\_thresh=0.8)

    if image\_name:

        plt.imsave(image\_name, image\_np\_with\_annotations)

    else:

        plt.imshow(image\_np\_with\_annotations)