MACHINE LEARNING

Project -
Report 01

COURSE INSTRUCTOR
DR
WAJAHAT HUSSAIN
MS-EE (AI&AS)

TEAM MEMBERS

Rana Muhammad Saad (400363)

Abdullah Khalid (401168)

AsadUllah Zafar (402646)

TITLE: How close Fake Depth is to Real Depth

Problem Statement:

- What's the difference between virtual depth and real-world depth?
- How can we automatically detect if the image is real or synthetic based on its depth map
- How can we make fake depth maps close to real depth maps

Abstract:

Fake images becoming very common nowadays with the advent of deepfakes along with the fact that simulator engines, including game engines, are becoming more and more powerful, creating more realistic images. This can create problems for people as these can use their misleading nature. We propose a model through the depth maps to catch fake images.

Introduction:

Recent advances in artificial intelligence enable the generation of fake images and videos with a incredible level of realism. our study analyzes the alteration introduced by the manipulation on RGB and depth features. Monocular depth estimation (MDE) is a fundamental task in computer vision where a per-pixel distance map is reconstructed from a single RGB input image. Unlike other passive depth perception methodologies such as the binocular stereo vision and the multi-view systems, this depth perception approach does not impose any geometric constraints due to the use of a single camera, thus leading to a multitude of possible applications such as scene understanding and robotics [1][3]. These spatial features contain semantic information that has the advantage of being more easily interpret able and robust to strong compression operations. With these strengths, semantic features can help solve the lack of explainable detectors, which do not limit themselves to classifying the contents as true or false but allow us to understand what information led to a certain decision [2].

Proposed methodology:

To analyze semantic features, we will extract the depth of the depth map with a monocular-based estimation method that is concatenated to the RGB image. We than train a network to classify each frame as real or fake. These two modalities enable the analysis of semantic inconsistencies in each frame by investigating colour and spatial irregularities.

We analyze the importance of depth features and show that they consistently improve the detection rate. To the best of our knowledge, this is the first work that analyses the depth inconsistencies left by the deepfake creation process [4].

We investigate the contribution of the RGB data and show that a simple RGB-to grayscale conversion can still lead to acceptable or even higher results in some experiments. We hypothesize that there are semantic features in this conversion that still allow good detection despite the reduction of input channels.

To automate the detection process, we will train a Convolutional Neural Network (CNN) on the data we have pre-processed such that the features that differentiate between real and fake depth maps are enhanced, making it easy for the model to generalize. If the same scenery between the real and fake image is used, we may not need the pre-process it as the deepfakes are analyzed in [6]. We can see the differences between the real and fake depth maps also by seeing what features the CNN has extracted as well. This information will help us in guiding simulators on making more realistic depth maps.

The depth image is the most used representation of range data, and it is a mere prequantization of the D distance matrix. A depth image ID is encoded as a one-channel grey-scale image, in which the intensity of each pixel represents the quantized version of dij. This representation is usually referred as depth image, as well as range or 2.5D image. Spatial resolution, depth precision, and data format strictly depend on the acquisition device. Frequently, 8-bit grey-scale image formats are used to increase compatibility and facilitate viewing. Consequently, the computed depth image loses the full 3D content of the original depth map, in exchange for a 2D representation, which is easier to manage.

Revised Work division:

Rana Muhammad Saad & AsadUllah Zafar:

1. Gather the usable dataset, analyze the CNN model result for differences in real and fake depth maps.
2. Tunning the accuracy of trained model by using Tricks underfit, overfit, lymda.

Muhammad Abdullah Khalid:

1. CNN model training for selected dataset.

Progress Update:

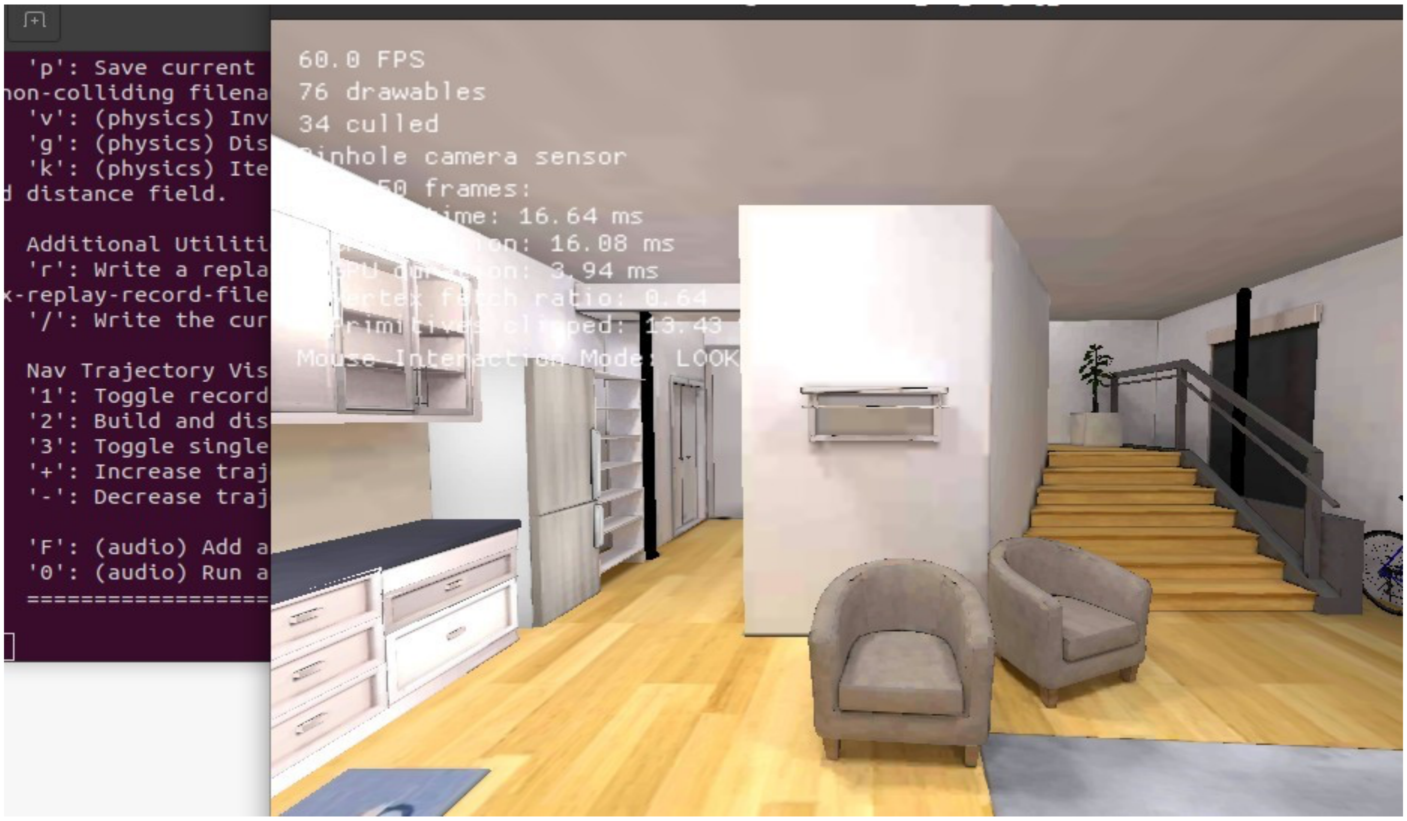
1. DATASET COLLECTION:

we installed Habitat to access the virtual maps but there are some synthetic datasets like Replica_CAD and Replica, Matterport3D and SUN_cg.

Problems encountered:

The problem was with selection of synthetic dataset as some datasets are very large and are not available on sites and we have to mail them to access them. After getting access we come to know that some of them are not Synthetic. We couldn't find prefect virtual depth maps from them. Below are some of the datasets.

- Materport3D was not Synthetic, and it was about 15GB large.
- SUN_cg was also not available on site.
- Replica And Replica_CAD was synthetic dataset but only works on Habitat Simulator



```
  'p': Save current
non-colliding filena
  'v': (physics) Inv
  'g': (physics) Dis
  'k': (physics) Ite
d distance field.

  Additional Utiliti
  'r': Write a repla
x-replay-record-file
  '/': Write the cur

  Nav Trajectory Vis
  '1': Toggle record
  '2': Build and dis
  '3': Toggle single
  '+': Increase traj
  '-': Decrease traj

  'F': (audio) Add a
  '0': (audio) Run a
  ==================
```

```
60.0 FPS
76 drawables
34 culled
Pinhole camera sensor
  50 frames:
     ime: 16.64 ms
        on: 16.08 ms
  GPU duration: 3.94 ms
  Vertex fetch ratio: 0.64
  Primitives clipped: 13.43
Mouse-Interaction Mode: LOOK
```

Fig.1 Replica_CAD runs on Habitat Simulator

Final Selected dataset:

- **Real dataset:** NYU dataset (4GB) – Office_003
- **Synthetic dataset:** Augmented ICL-NUIM dataset (almost 2GB) – Office1_depth_clean

2. Problems encountered while Coding:

i **Uploading Data:**

For now, we have uploaded 8gb dataset on Drive so we can Access it from Colab to train model using GPU's. It took time as it was large large dataset. Only part left is to train our model on CNN.

ii **Training error:**

Since the intensity levels of both datasets does not match, we can't generalize if depth of something in real map is same as depth of anything in virtual map that is at the same distance.

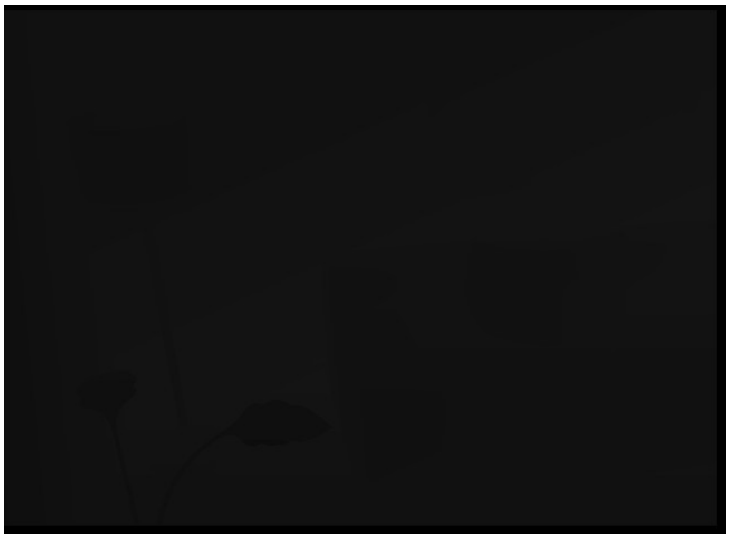


Fig.2 Fake Depth

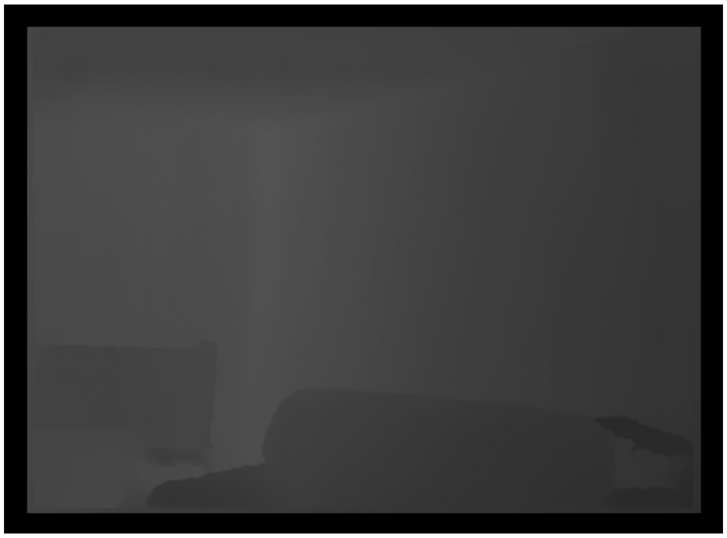
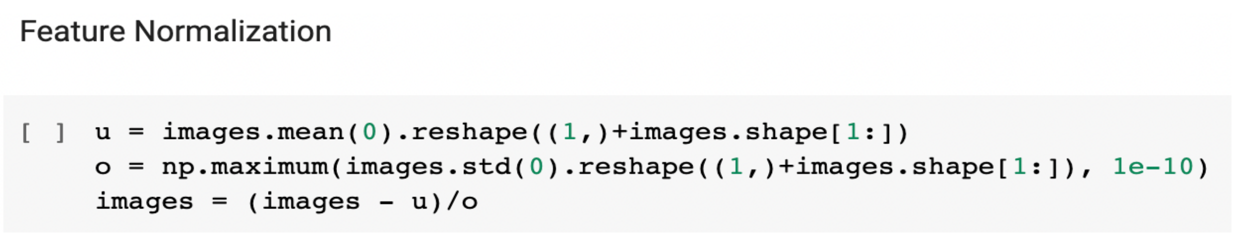


Fig.3 Real Depth

Solution:

We have normalized the features of both images. For reference, sample images are attached below.

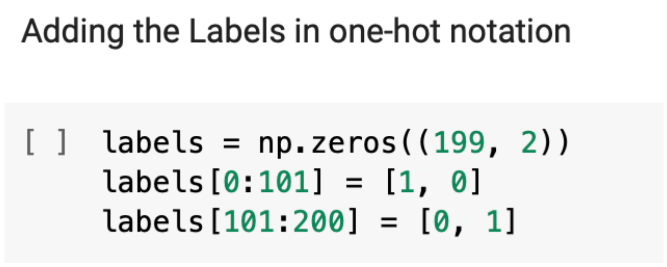


Feature Normalization

```
[ ] u = images.mean(0).reshape((1,)+images.shape[1:])
    o = np.maximum(images.std(0).reshape((1,)+images.shape[1:]), 1e-10)
    images = (images - u)/o
```

iii. **Labelling the data:**

Out of 199 images in total, 100 belongs to virtual dataset & 99 belongs to nyu. We have used one hot notation to create vectors for both for classification such that,

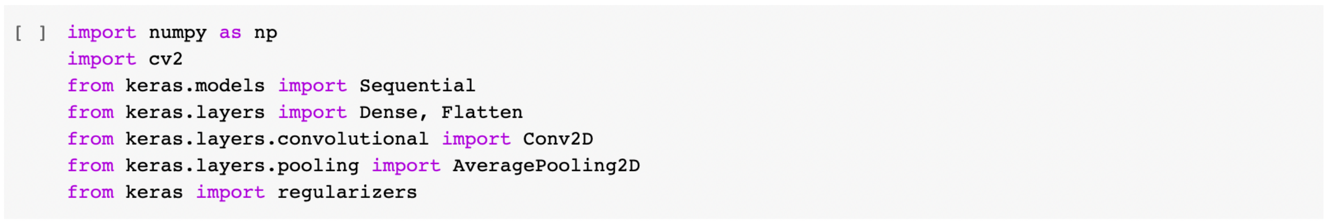


Adding the Labels in one-hot notation

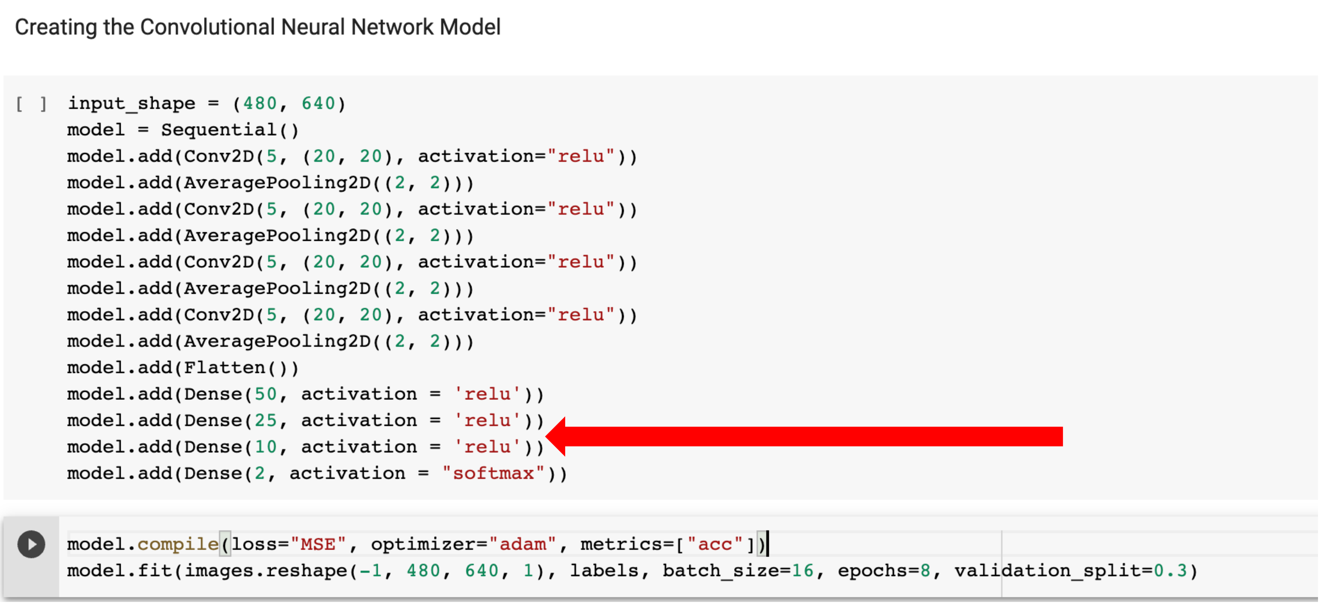
```
[ ] labels = np.zeros((199, 2))
    labels[0:101] = [1, 0]
    labels[101:200] = [0, 1]
```

The data was shuffled afterwards.

iv. **Model Training Issue:**



```
[ ] import numpy as np
    import cv2
    from keras.models import Sequential
    from keras.layers import Dense, Flatten
    from keras.layers.convolutional import Conv2D
    from keras.layers.pooling import AveragePooling2D
    from keras import regularizers
```



Creating the Convolutional Neural Network Model

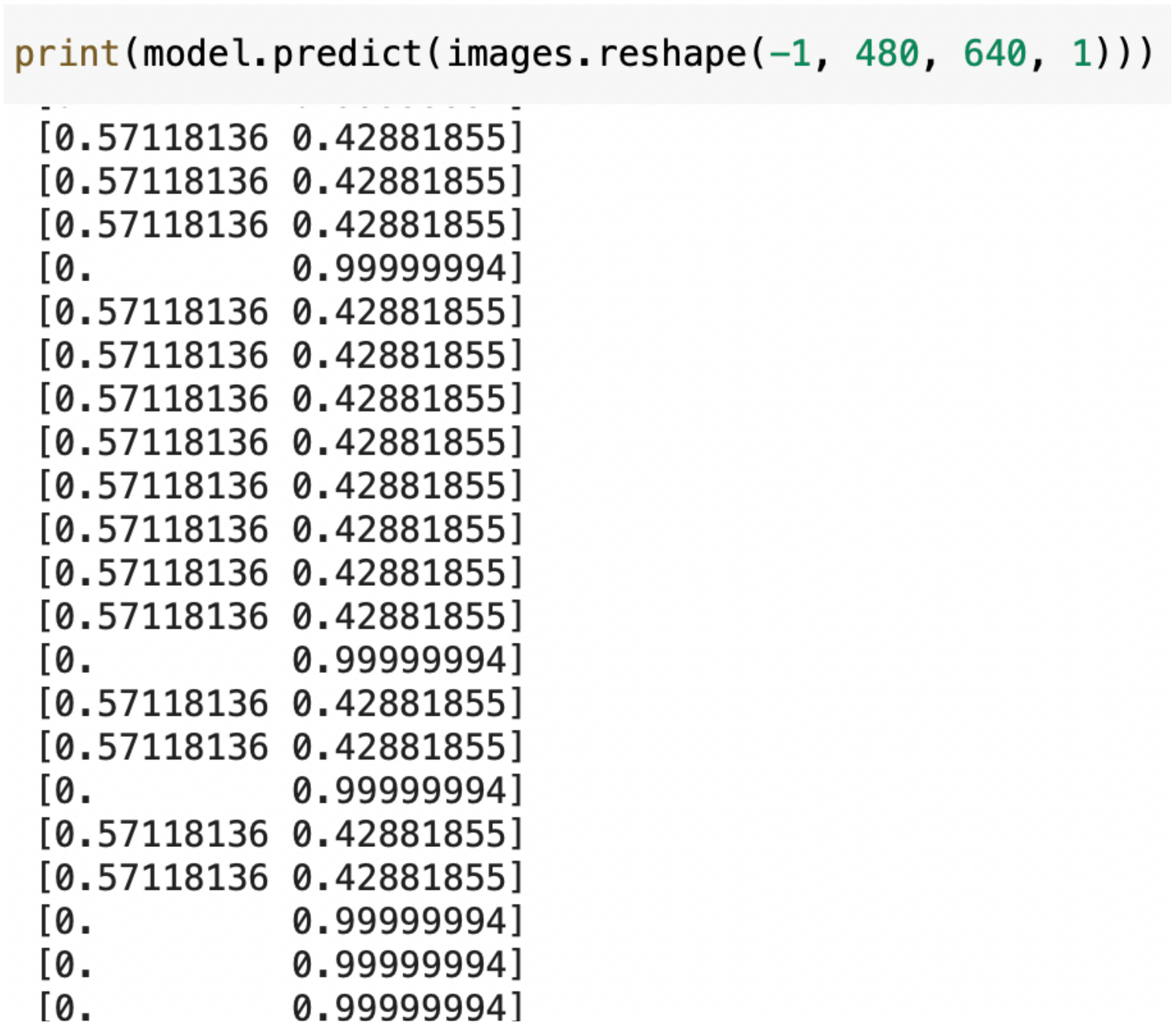
```
[ ] input_shape = (480, 640)
    model = Sequential()
    model.add(Conv2D(5, (20, 20), activation="relu"))
    model.add(AveragePooling2D((2, 2)))
    model.add(Conv2D(5, (20, 20), activation="relu"))
    model.add(AveragePooling2D((2, 2)))
    model.add(Conv2D(5, (20, 20), activation="relu"))
    model.add(AveragePooling2D((2, 2)))
    model.add(Conv2D(5, (20, 20), activation="relu"))
    model.add(AveragePooling2D((2, 2)))
    model.add(Flatten())
    model.add(Dense(50, activation = 'relu'))
    model.add(Dense(25, activation = 'relu'))
    model.add(Dense(10, activation = 'relu'))
    model.add(Dense(2, activation = "softmax"))
```

```
model.compile(loss="MSE", optimizer="adam", metrics=["acc"])
model.fit(images.reshape(-1, 480, 640, 1), labels, batch_size=16, epochs=8, validation_split=0.3)
```

We were using the keras model built in functions for training our model. Layers was selected randomly by hit & trail. However, before adding the last two layers (pointed by red arrow) training accuracy wasn't good.

v. **Predicting the output:**

We have generated the array based on their probabilities such that if the map is real is probability of 2nd column will be greater and if it's virtual, it's probability of 1st column will be greater. In total, sum of each row will be 1 as its binary classification problem.



```
print(model.predict(images.reshape(-1, 480, 640, 1)))
```

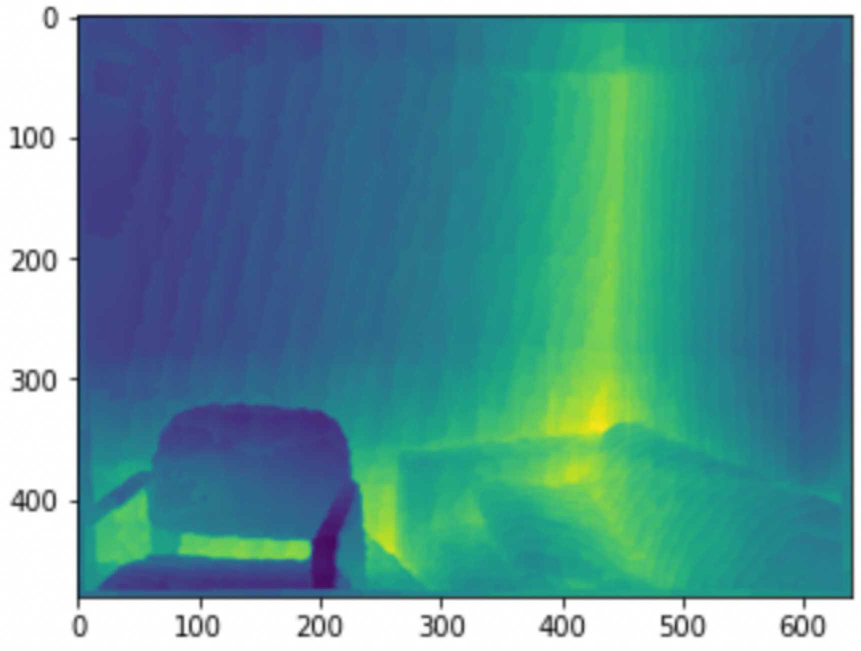
```
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.         0.99999994]
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.         0.99999994]
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.         0.99999994]
 [0.57118136 0.42881855]
 [0.57118136 0.42881855]
 [0.         0.99999994]
 [0.         0.99999994]
 [0.         0.99999994]
```

Once we have these probabilities, we have created a function for Real or fake(virtual) depth map, i.e.

```
def RealOrFake(image):
  j = model.predict(image.reshape(1, 480, 640, 1))
  if(j[0,0] > j[0,1]):
    print('Fake Depth')
  else:
    print('Real Depth')
```

```
import matplotlib.pyplot as plt
RealOrFake(images[6])
plt.imshow(images[6])
```

```
1/1 [==============================] - 0s 454ms/step
Real Depth
<matplotlib.image.AxesImage at 0x7fd59b839af0>
```



Reference:

[1]. S. Zhu, H. Xu and L. Yan, "An Improved Depth Image Based Virtual View Synthesis Method for Interactive 3D Video," in *IEEE Access*, vol. 7, pp. 115171-115180, 2019, doi: 10.1109/ACCESS.2019.2935021.

[2]. R. Nandhini Abirami, P. M. Durai Raj Vincent, Kathiravan Srinivasan, Usman Tariq, Chuan-Yu Chang, "Deep CNN and Deep GAN in Computational Visual Perception-Driven Image Analysis", Complexity, vol. 2021, Article ID 5541134, 30 pages, 2021. https://doi.org/10.1155/2021/5541134

[3]. Zhao, C., Sun, Q., Zhang, C. et al. Monocular depth estimation based on deep learning: An overview. Sci. China Technol. Sci. 63, 1612–1627 (2020). https://doi.org/10.1007/s11431-020-1582-8

[4]. Zhao, Z., & Yang, M. (2021, August). RGB-Based No-Reference Depth Map Quality Assessment. In *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)* (pp. 1-6). IEEE.

[5]. Li, Yuezun & Lyu, Siwei. (2018). Exposing DeepFake Videos By Detecting Face Warping Artifacts.

[6]. Trisha Mittal, Uttaran Bhattacharya, Rohan Chandra, Aniket Bera, and Dinesh Manocha. 2020. Emotions Don't Lie: An Audio-Visual Deepfake Detection Method using Affective Cues. In Proceedings of the 28th ACM International Conference on Multimedia (MM '20). Association for Computing Machinery, New York, NY, USA, 2823–2832. https://doi.org/10.1145/3394171.3413570

[7]. L. Maiano, L. Papa, K. Vocaj, and I. Amerini, “Depthfake: A depth-based strategy for detecting deepfake videos,” arXiv.org, 23-Aug-2022. [Online]. Available: https://arxiv.org/abs/2208.11074v1.