CS838: Neural Networks

Progress Report

Optimal Coding Functions for Continuous Wave Time of Flight Imaging

Zhicheng Gu, Nikhil Nakhate, Zhenyu Zhang, Felipe Gutierrez Barragan

1. Project Overview

Time of flight (ToF) refers to the time that a light pulse takes to travel from a source to a target scene and back to the detector. This technique is often used to recover scene depths. CW-ToF systems are one type of ToF cameras where the light source intensity and sensor exposure are temporally modulated by a modulation (M(t)) and demodulation (D(t)) function, respectively. As described in the project proposal, K ($K >= 3$) brightness measurements (b) of the incident light on the sensor are made to recover the scene depth.

The brightness measurements are susceptible to multiple signal-dependent (photon noise) and sensor-dependent (read noise, ADC noise) noise sources. Recent work has shown that certain modulation and demodulation functions are more robust to noise than others, leading to lower depth errors. *The goal of this project is to leverage neural networks to find sets of optimal sets of coding (modulation/demodulation) functions.* To this end, we developed two neural network architectures that learn the depth recovery process and output a depth estimate. The next section describes the network architectures.

2. Network Architectures (Implemented)

In this section we describe the neural networks that we have implemented and tested. See Section 5 for the neural networks that we will implement using these ones as building blocks.

a. ***Single Pixel Black Box Network:*** The single-pixel black box network is used to predict the depth based on K brightness measurements. The input to this network is the measured brightness, which is calculated based on the modulation, demodulation, and scene and sensor parameters. Currently we obtain the input brightness from a CW-ToF simulator, but in the following architecture the input brightness measurements will be an intermediate layer input.

The output is the predicted depth. The structure of this network that we are using now is shown in Figure 1.

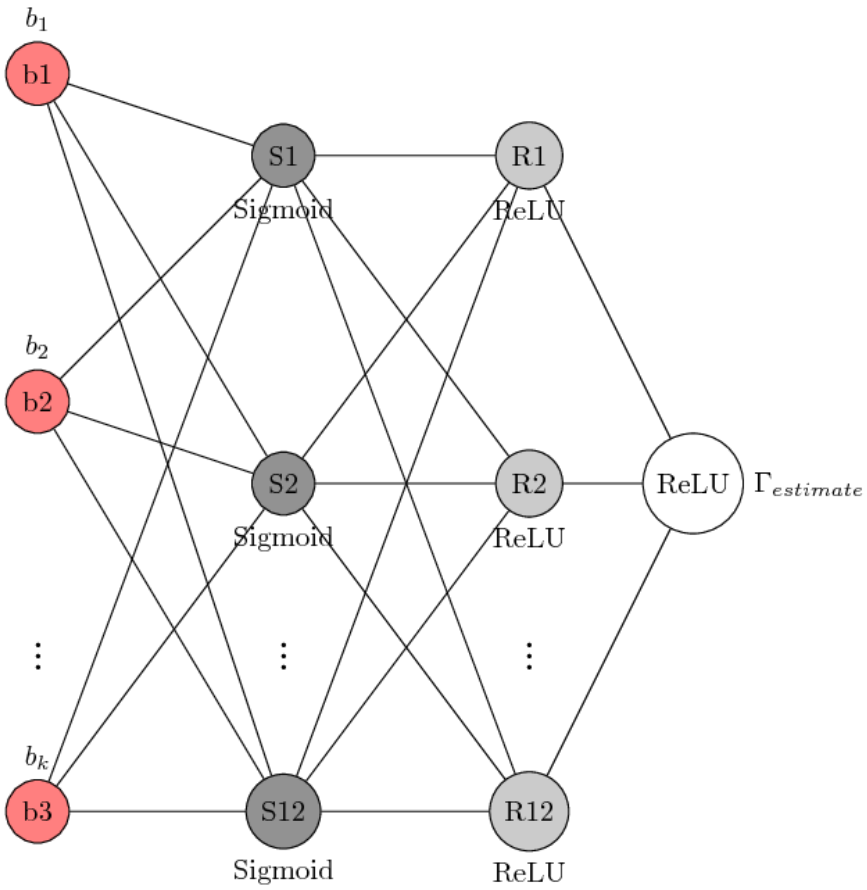

Figure 1: Black box neural network. This network takes as input K brightness measurements and outputs a depth prediction.

The input dimensions are equal to the K measurements (set to three in our evaluations). We use a fully-connected neural network with two hidden layers, with 12 hidden units in each layer. The first layer uses the Sigmoid activation function. This will add non-linear transformation to the input data. Then we use ReLu activation function in the second hidden layer to scale up the output of Sigmoid function. In the last layer there is only one output which is the depth that the network predict.

The loss function we use is mean absolute error. The other options for loss function are mean squared error and mean absolute percentage error. We choose mean absolute error as the loss function because this value makes more sense when comparing the estimated depth to the

true depth. The drop out or early stopping has not been added to the network because there is no over-fitting issue right now. This also means the result can be better if we train for a longer time. We also tried using Leaky ReLU to replace the ReLU function and the result becomes better.

b. Single Pixel Neural Network: Figure 2 illustrates the network architecture.

Input: The input to this neural network is the discretized modulation functions with a phase shift associated to a true depth.

Demodulation Layer: The first layer weights, d_i, correspond to the discrete demodulation function. During training the neural network is updating the demodulation function weights that will minimize the depth error. Consequently, the demodulation weights of the trained network will represent an optimal demodulation function.

Image Formation Activations: The *K* b_i neurons are the noiseless brightness measurements obtained from the dot product of the modulation and the demodulation functions. The activation function applied to the b_i inputs will represent the various transformations that a brightness measurement undergoes during the image formation process in the physical world. These correspond to signal dependent and sensor dependent noise sources. Thus, we apply the following transformations to b_i:

1. *Photon noise (signal dependent)*: We draw a normal random variable ~ N(0, variance=b_i) and add it to b_i.
2. *Read noise (sensor dependent)*: We draw a normal random variable ~ N(0, variance) and add it to b_i. The read noise variance is a parameter specified at the beginning or that can be chosen randomly from a range of typical variances.
3. *Quantization noise:* This is the noise associated with the loss of accuracy due to analog to digital conversion.

Then we use the black box previously defined to produce a depth estimate.

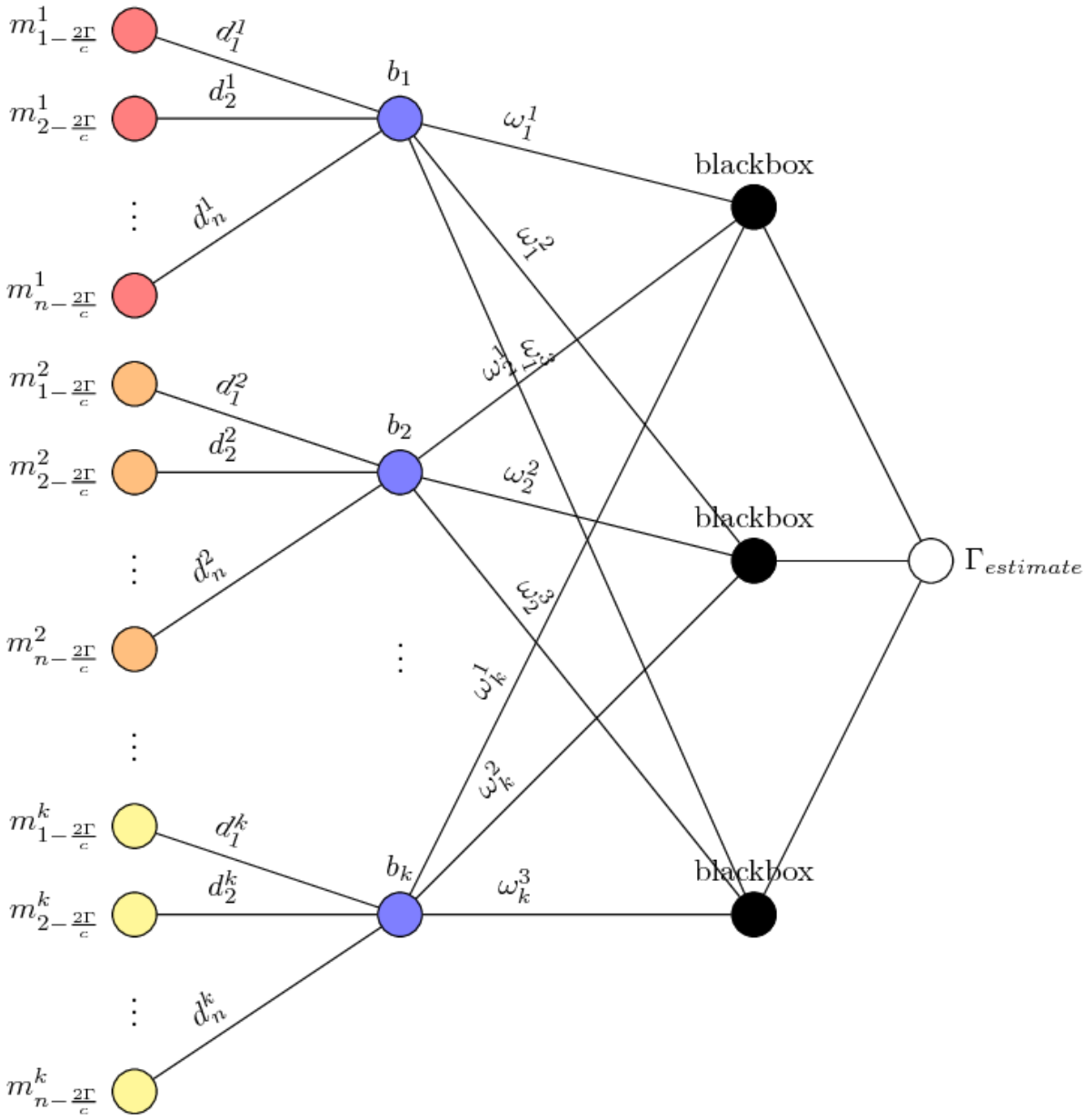

Figure 2: Single-pixel Neural Network Architecture. The inputs are *K* phase shifted modulation functions. The first layer weight (d) represent the demodulation function. The brightness measurements are obtained from the vector product of the modulation and demodulation functions and adding the multiple noise sources. The rest of the network referred to as the black box takes the brightness measurements and outputs a depth estimate.

3. Simulated Datasets

Each network architecture requires a different dataset. In this section we describe the datasets we have prepared for each network.

a. ***Single Pixel Black Box Network Data:*** This network takes as input K brightness measurements for a single-pixel. Hence, these measurements need to be obtained from somewhere. To this end we developed a physics-based CW-ToF simulator (see Appendix for further details) that simulates the image formation process. The simulator outputs K brightness measurements corresponding to the following inputs:
   - *Coding functions:* A set of K modulation and demodulation functions.
   - *Scene parameters:* Ambient illumination, reflectance factor, normals, and true depths.
   - *Light Source and Sensor parameters:* Read noise, number of bits (determines quantization noise), source/sensor coordinates, light source strength.

Given the flexibility of the simulator we can choose to generate brightness measurement for a set of parameters. We can then generate data for the network by setting the parameters and running the simulator. After running the simulator R times we end up with a brightness measurement dataset. We generated two types of datasets. In the first type we fixed all the parameters, draw the true depth of the pixel from a uniform random distribution between 0 and max depth, and ran the simulator R times. The second dataset type we generated we followed the same steps as in the first one, except that at each simulation run we vary a subset of the parameters. Following these two strategies we generate a total of 7 datasets varying the parameters that impact the amount of noise the brightness measurement will have.

Table 1. Type I Dataset.

Type I Datasets						
Parameters\Dataset ID	**18_1**	**18_10**	**18_50**	**19_1**	**19_10**	**19_50**
R (Number of runs)	100,000	100,000	100,000	100,000	100,000	100,000
True Depth Range (uniform random varible)	[0,5000]	[0,5000]	[0,5000]	[0,5000]	[0,5000]	[0,5000]
Ambient Illumination	10^18	10^18	10^18	10^19	10^19	10^19
Read Noise Variance	1	10	50	1	10	50
Demodulation Function	Sinusoid	Sinusoid	Sinusoid	Sinusoid	Sinusoid	Sinusoid
Modulation Function	Sinusoid	Sinusoid	Sinusoid	Sinusoid	Sinusoid	Sinusoid

Table 2. Type II Dataset.

Type II Dataset	
Parameters\Dataset ID	**Rand_Rand**
R (Number of runs)	100,000
True Depth Range (uniform random varible)	Uniform Random Variable ~ Unif(0,5000)
Ambient Illumination	Uniform Random Variable ~ 10 ^ Unif(18, 19)
Read Noise Variance	Uniform Random Variable ~ Unif(1, 50)
Demodulation Function	Sinusoid
Modulation Function	Sinusoid

The type I datasets intend to emulate *fixed* real world scenarios, hence they should result in a network that performs well when the given brightness measurements are obtained from scenarios with the same fixed parameters. The type II dataset intends to emulate *all* possible real-world scenarios (any ambient illumination and read noise). Hence, a network trained on this dataset should perform well no matter which parameter combination the brightness measurements are obtained from.

b. Single Pixel Neural Network Data: This network as described in the previous section takes as input K vectors associated to the K discretized modulation functions. The input vectors will be phase shifted versions of their modulation functions, and hence will have a true depth associated to them. Therefore, to generate a dataset for this network we:

- Fix the K modulation functions.
- Draw a random true depth from a uniform distribution between 0 and a fixed maximum depth.
- Shift the K modulation functions by a phase shift corresponding to the random depth.

4. Results + Implementation Details

Seven dataset were created using the parameters described in Tables 3 and 4. The numbers 18 and 19 in Dataset ID indicate ambient illumination level. The numbers 1, 10, 50 in Dataset ID indicate the variance of Gaussian read noise. Test set 7 is generated by randomly selecting illumination level between 18 and 19 and randomly setting the read noise variance to a value between 1 and 50. These parameters are given to the simulator. The Analytical Depth Error is calculated as the baseline or reference column. Neural Net Trained on Dataset X column gives the testset error of the same dataset file. In this case, each file is splitted into train tune and test set. Neural network is trained, tuned, tested inside the same file. Neural Net trained on Dataset 7 column gives the test set error of each data set with the neural net model trained only using Dataset 7. In this case, a neural net was first trained using dataset 7, then this structure is used from Test Set 1 to Test Set 6 to calculate the test error.

We use Keras with TensorFlow backend to implement the structure. Stochastic gradient descent with batch size 128 is used to train the network. 50 is chosen as the epoch number. Table 3 is obtained by using ReLu as the activation function in the second hidden layer. Table 4 is generated by using leaky ReLu as the activation function of the second layer in the neural network. Table 3 shows a good performance among the last four datasets but a relatively high error from test set 1 to test set 3. Table 4 decrease the mean absolute depth error noticeably. It achieved results closer to the analytical depth error among the last four datasets. It also increase the accuracy in the first three test set.

Table 3. Mean Absolute Depth Error on Testing Set using ReLu

Mean Absolute Depth Error on Testing Set			
Dataset ID	**Analytical Depth**	**Neural Net Trained on Dataset X**	**Neural Net Trained on Dataset 7**
Test Set 1 18_1	45.953	71.521	105.154
Test Set 2 18_10	45.878	80.878	107.041
Test Set 3 18_50	50.656	67.947	107.192

Test Set 4 19_1	122.434	139.314	**130.387**
Test Set 5 19_10	122.854	133.350	**131.820**
Test Set 6 19_50	124.160	157.204	**132.983**
Test Set 7 UnifRand[18-19]_UnifRand[1-50]	67.281	75.082	75.082

Table 4. Mean Absolute Depth Error on Testing Set using Leaky ReLu

Mean Absolute Depth Error on Testing Set		
Dataset ID	**Analytical Depth**	**Neural Net Trained on Dataset 7**
Test Set 1 18_1	45.953	74.303
Test Set 2 18_10	45.878	75.701
Test Set 3 18_50	50.656	77.477
Test Set 4 19_1	122.434	**126.465**
Test Set 5 19_10	122.854	**128.227**
Test Set 6 19_50	124.160	**128.864**
Test Set 7 UnifRand[18-19]_UnifRand[1-50]	67.281	**71.328**

5. Next Steps

The next steps would be to implement the Single Pixel Neural Network described in section 4.b. to work as desired (with error close to or better than the analytical error measure) by conducting experiments with different architectures of the black box neural networks and also with different activation functions. The following are the specifications of the Convolutional Neural Network that we would implement:

1. **Full-Image Black Box CNN:**
    a. **Input:** K=3 brightness measurements per pixel (nRows x nCols x 3).
    b. **Output:** Depth Map (nRows x nCols).
2. **Full-Image Depth Estimation Network**
    a. **Input:** K=3, nx1 modulation function vectors (nRows x nCols x n x 3)
    b. **Output:** Depth Map (nRows x nCols).

6. Appendix: Physics-based CW-ToF Simulator

The CW-ToF simulator we developed follows the steps outlined in Figure 3. Given a scene represented by a set of pixels with parameters associated to them (true depth, albedos, normals, ambient illumination), the simulator simulates the image formation process when light is modulated onto the scene and gives *K* brightness measurements for each pixel. To generate the single-pixel datasets we simply set the scene to be a single-pixel.

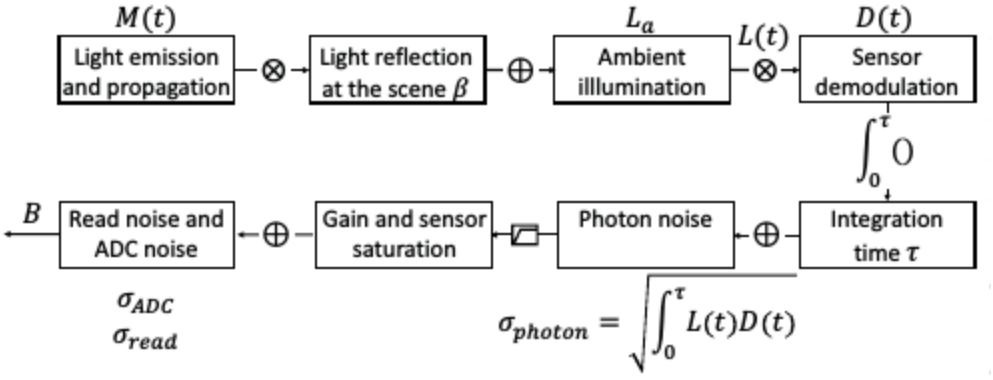

Figure 3: CW-ToF Simulation Pipeline.