BURSA TEKNİK ÜNİVERSİTESİ

LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ (T) BÖLÜMÜ

BİLGİSAYAR OYUNLARDA YAPAY ZEKA

Ödev-5 Raporu

Github: https://github.com/MoussaBane/BOYZ-Perceptron-Train-Test

MOUSSA BANE

24435004029

Giriş:

Bu rapor, **Perceptron**'un temel mantığını anlamak, ve, veya, ve xor işlemlerini öğrenmesi için nasıl eğitilebileceğini göstermek amacıyla hazırlanmıştır.

- Perceptron, bir sinir ağının işleyişinin ardındaki temel bir algoritmadır.
- İnsan beyninin temel yapı taşlarını (nöron) taklit eden bir programdır.
- Bir nöron, yalnızca beyinde değil, sinir sistemi boyunca vücudun başka yerlerinde de bulunan bir sinir hücresidir.
- Nöronlar, vücuda elektriksel uyarılar gönderen bir sinir ağı içerisinde birbirine bağlanır.
- Bir nöron, gelen sinyaller bir giden sinyale neden olduğunda tetiklenir.

- Gelen sinyaller her zaman bir nöronun tetiklenmesine neden olmaz.
- Hücredeki enerji birikimi ancak bir eşik değer aşılırsa hücre onu (sinyali) serbest bırakır.

1. **Eğitim Seti Tanımlama (TrainingSet Sınıfı)**

   TrainingSet sınıfı, Perceptron'a verilecek giriş-çıkış çiftlerini tanımlamak için kullanılır. Her bir input dizisi ve output (beklenen sonuç) bu sınıfta tutulur.

```
// Represents a single training set with inputs and the desired output.
[System.Serializable]
1 reference
public class TrainingSet
{
    2 references
    public double[] input; // Input values for the perceptron
    1 reference
    public double output;  // Expected output for the given input
}
```

   Unity'de bu sınıfı tanımladıktan sonra, bir *public TrainingSet[] ts;* değişkeni oluşturuyoruz. Bu değişken sayesinde giriş-çıkış verilerini Unity arayüzünden kolayca düzenleyebiliriz.

2. **Unity Arayüzünden Eğitim Verilerini Tanımlama**

   Unity içinde bir GameObject oluşturun ve kodu bu objeye bağlayın.

   TrainingSet öğesini ekleyerek ve veya veya işlemleri için giriş-çıkış değerlerini düzenleyin.

   Ve işlemi için:

Giriş (input[0], input[1])	Çıkış (output)
0, 0	0
0, 1	0
1, 0	0
1, 1	1

   Veya işlemi için:

Giriş (input[0], input[1])	Çıkış (output)
0, 0	0

0, 1	1
1, 0	1
1, 1	1

3. Perceptron Eğitimi İçin Fonksiyonlar

➢ Ağırlık ve Bias Tanımları

Perceptron'da iki ağırlık (weights) ve bir bias kullanılır. Ağırlıklar ve bias, eğitimin başlangıcında rastgele atanır:

```
4 references
public TrainingSet[] ts; // Array of training sets to train the perceptron
8 references
double[] weights = { 0, 0 }; // Weights associated with each input
4 references
double bias = 0; // Bias term for the perceptron
3 references
double totalError = 0; // Accumulated error during training
```

➢ DotProductBias Fonksiyonu

Bu fonksiyon, giriş vektörü ile ağırlık vektörünün çarpımını ve bias'ı hesaplar:

```
// Computes the dot product of two vectors and adds the bias term.
2 references
double DotProductBias(double[] v1, double[] v2)
{
    if (v1 == null || v2 == null)
        return -1; // Return -1 if inputs are null (error condition)
    if (v1.Length != v2.Length)
        return -1; // Return -1 if input vectors have different sizes (error condition)

    double d = 0; // Accumulator for the dot product
    for (int x = 0; x < v1.Length; x++)
    {
        d += v1[x] * v2[x]; // Multiply corresponding elements and sum them
    }
    d += bias; // Add bias to the result
    return d;
}
```

➢ Aktivasyon Fonksiyonu

Perceptron'un çıktı üretmesi için bir aktivasyon fonksiyonu kullanılır. Basit bir eşik fonksiyonu, sonucu ya 1 ya da 0 olarak verir:

```
// Calculates the perceptron output for the training set at index i.
1 reference
double Calcoutput(int i)
{
    double dp = DotProductBias(weights, ts[i].input); // Compute weighted sum + bias
    return dp > 0 ? 1 : 0; // Return 1 if positive, otherwise return 0
}
```

➢ **Ağırlık Güncelleme**

Eğitim sırasında, Perceptron'un ağırlıkları ve bias'ı, hataya göre güncellenir:

```
// Updates the weights and bias based on the error for a given training set.
1 reference
void UpdateWeights(int j)
{
    double error = ts[j].output - Calcoutput(j); // Calculate error (desired - actual)
    totalError += Mathf.Abs((float)error); // Accumulate absolute error

    // Adjust weights based on error and input
    for (int i = 0; i < weights.Length; i++)
    {
        weights[i] += error * ts[j].input[i];
    }
    bias += error; // Adjust bias
}
```

➢ **Eğitim (Train) Fonksiyonu**

Perceptron'u birden fazla **epoch** boyunca eğitmek için Train fonksiyonu kullanılır.

Her epoch sonunda toplam hata kontrol edilir.

```
// Trains the perceptron using the training set over a specified number of epochs.
1 reference
void Train(int epochs)
{
    InitialiseWeights(); // Initialize weights and bias
    for (int e = 0; e < epochs; e++)
    {
        totalError = 0; // Reset total error at the start of each epoch
        for (int t = 0; t < ts.Length; t++)
        {
            UpdateWeights(t); // Update weights for each training example
            Debug.Log("W1: " + (weights[0]) + "W2: " + (weights[1]) + "B: " + bias);
        }
        Debug.Log("TOTAL ERROR: " + totalError); // Log total error for the epoch
    }
}
```

4. Unity'de Eğitimi Test Etme

➢ Unity sahnesinde bir GameObject oluşturun ve bu objeye Perceptron kodunu ekleyin.

➢ Eğitim setini doldurun (örneğin, **ve** işlemi için giriş-çıkışları yukarıda belirtildiği gibi girin).

➢ Train fonksiyonunu çağırarak eğitimi başlatın.

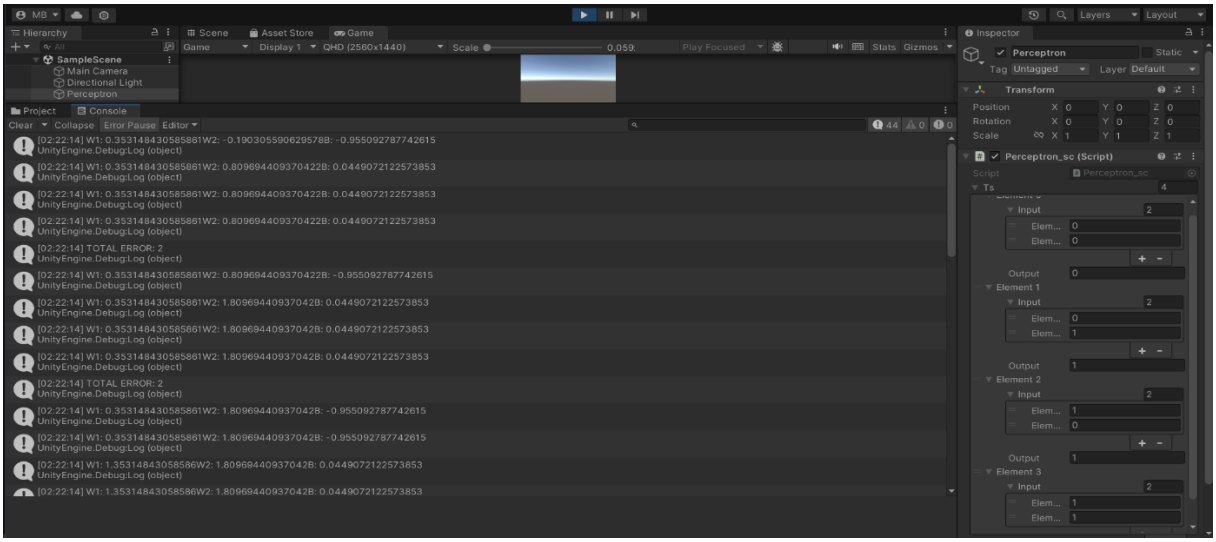
XOR işlemi için:

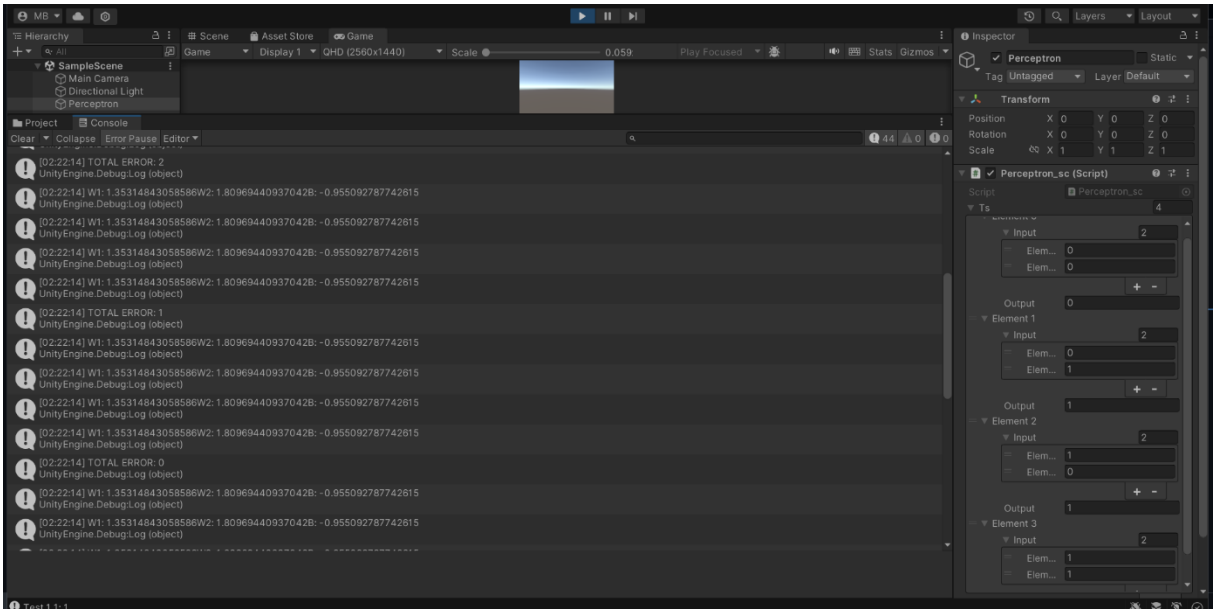
Giriş (input[0], input[1])	Çıkış (output)
0, 0	0
0, 1	1
1, 0	1
1, 1	0

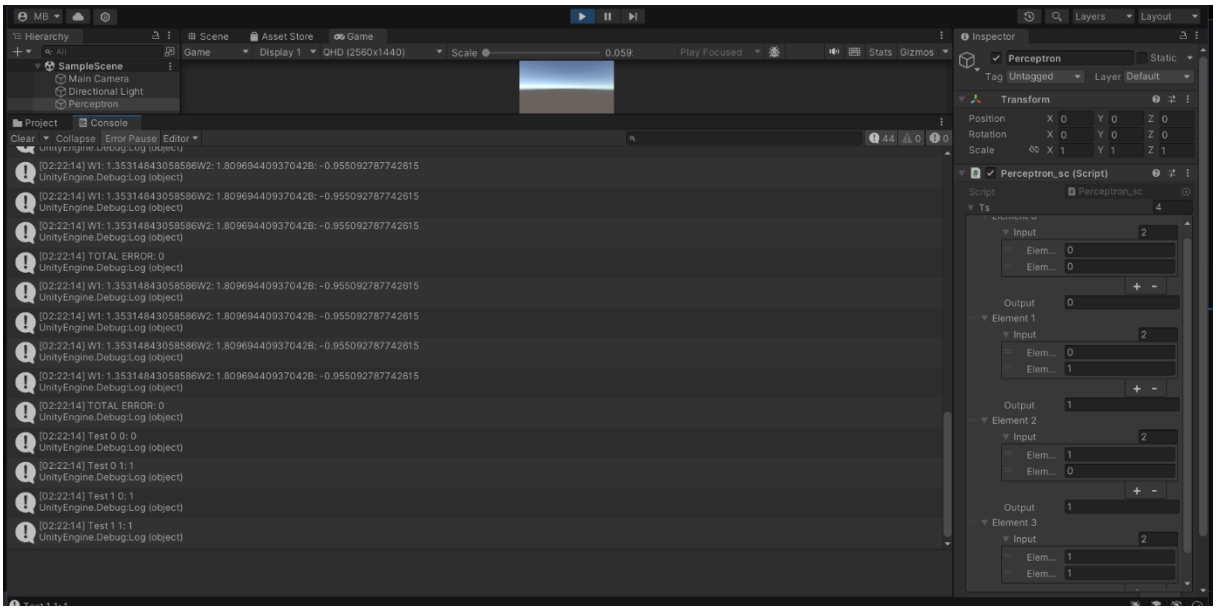
Perceptron'un **XOR** işlemini öğrenmesi için giriş-çıkış değerlerini değiştirin.

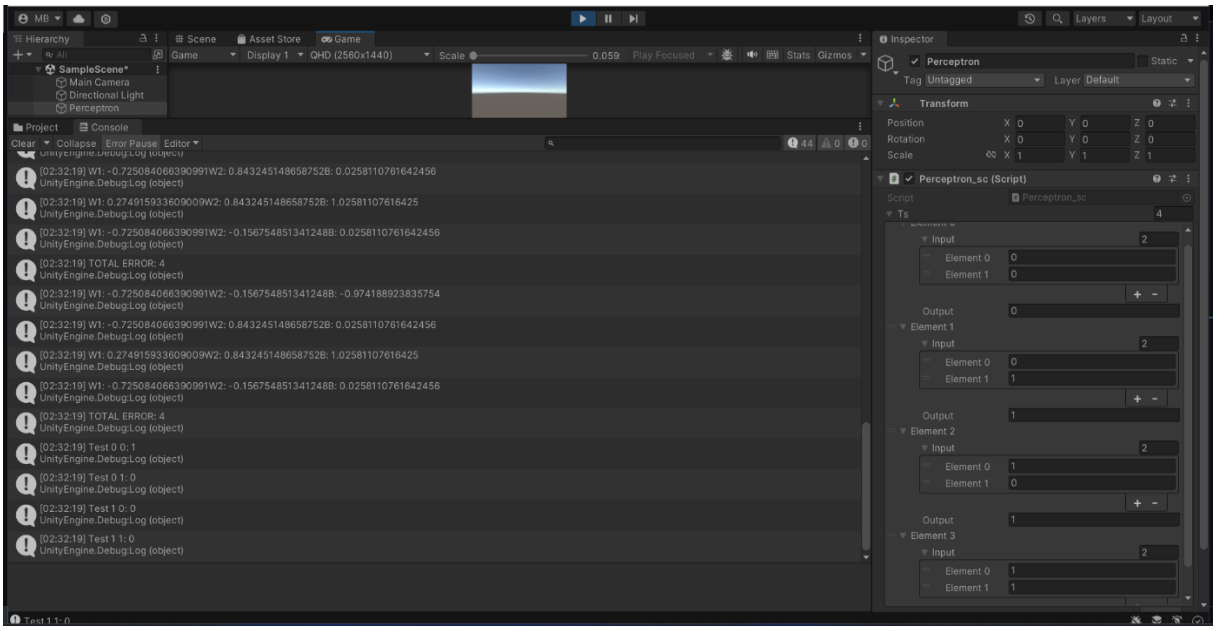
XOR işlemi için eğitimi deneyin. Sonuçta Perceptron başarısız olacaktır çünkü bir Perceptron doğrusal olarak ayrılabilir problemleri çözebilir. Ancak epoch sayısını artırarak değişimleri gözlemleyebilirsiniz.

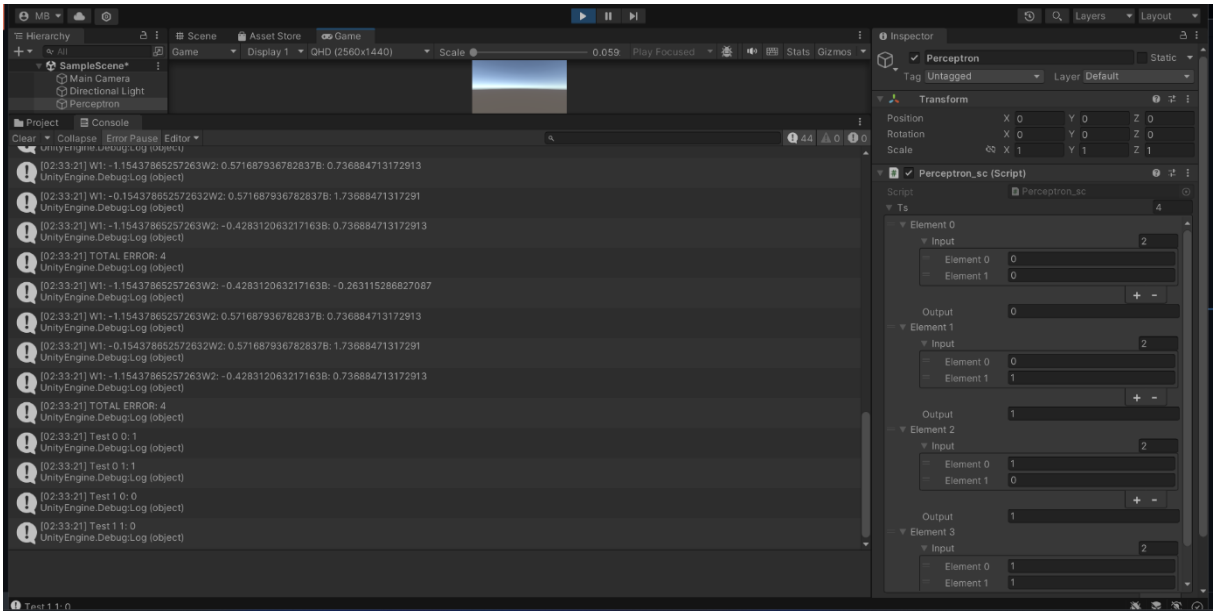
5. Sonuçların Bazı Ekran Görüntüleri

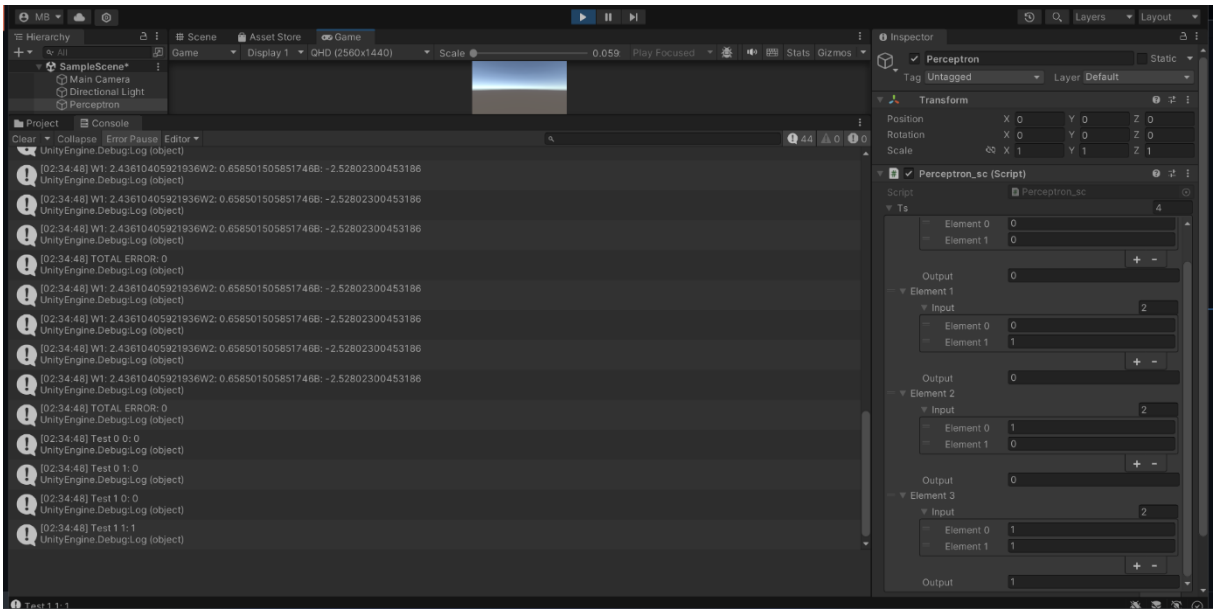


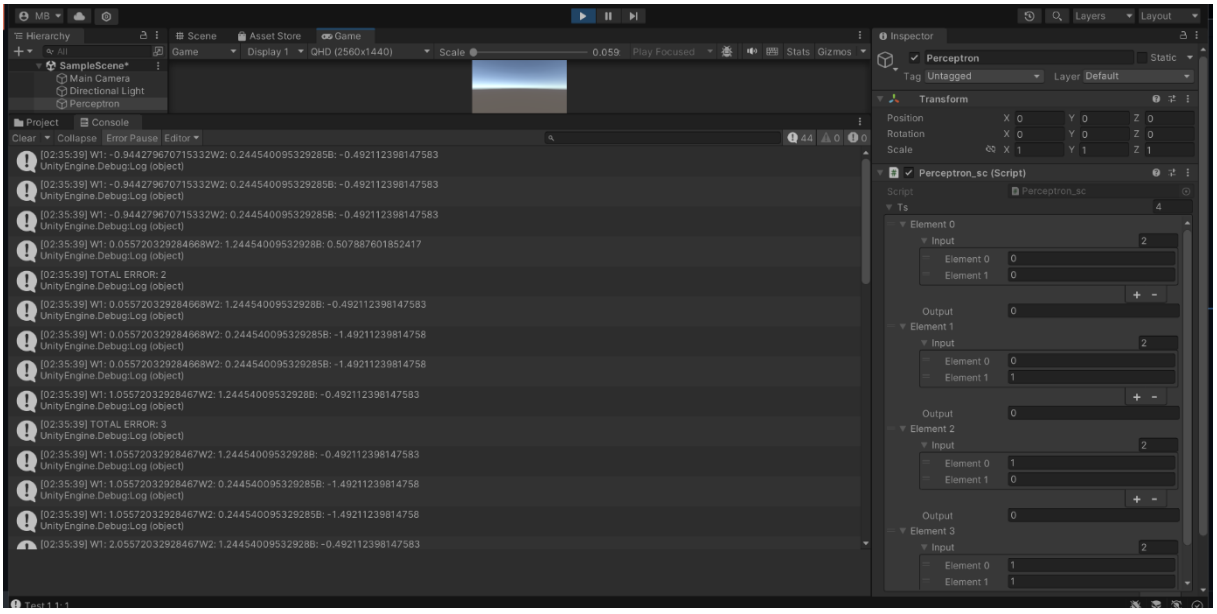


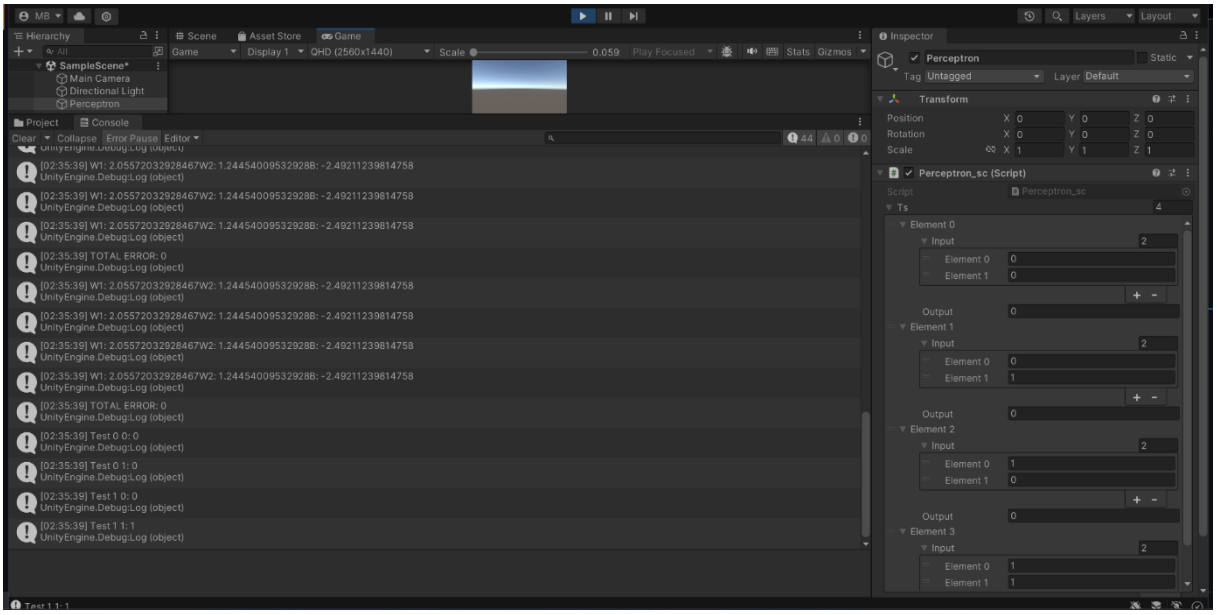

Hierarchy
Scene
Asset Store
Game
SampleScene
Main Camera
Directional Light
Perceptron
Project
Console
Clear
Collapse
Error Pause
Editor
Inspector
Perceptron
Static
Tag Untagged
Layer Default
Transform
Position
Rotation
Scale
Perceptron_sc (Script)
Script
Perceptron_sc
Ts
Input
Output
Element 1
Element 2
Element 3
[02:22:14] W1: 1.353148430585862W2: 1.809694409370428: -0.955092787742615
UnityEngine.Debug:Log (object)
[02:22:14] TOTAL ERROR: 0
[02:22:14] Test 0 0: 0
[02:22:14] Test 0 1: 1
[02:22:14] Test 1 0: 1
[02:22:14] Test 1 1: 1


Hierarchy
Scene
Asset Store
Game
SampleScene*
Main Camera
Directional Light
Perceptron
Project
Console
Clear
Collapse
Error Pause
Editor
Inspector
Perceptron
Static
Tag Untagged
Layer Default
Transform
Position
Rotation
Scale
Perceptron_sc (Script)
Script
Perceptron_sc
Ts
Input
Output
Element 1
Element 2
Element 3
[02:32:19] W1: -0.7250840663909912W2: 0.8432451486587528: 0.025811076164245
[02:32:19] W1: 0.2749159336090088W2: 0.8432451486587528: 1.02581107616425
[02:32:19] W1: -0.7250840663909912W2: -0.1567548513412488: 0.025811076164245
[02:32:19] TOTAL ERROR: 4
[02:32:19] W1: -0.7250840663909912W2: -0.1567548513412488: -0.974188923835754
[02:32:19] Test 0 0: 1
[02:32:19] Test 0 1: 0
[02:32:19] Test 1 0: 0
[02:32:19] Test 1 1: 0


Hierarchy
Scene
Asset Store
Game
SampleScene*
Main Camera
Directional Light
Perceptron
Project
Console
Clear
Collapse
Error Pause
Editor
Inspector
Perceptron
Static
Tag Untagged
Layer Default
Transform
Position
Rotation
Scale
Perceptron_sc (Script)
Script
Perceptron_sc
Ts
Input
Output
Element 1
Element 2
Element 3
[02:33:21] W1: -1.154378652572632W2: 0.5716879367828378: 0.736884713172913
[02:33:21] W1: -0.154378652572632W2: 0.5716879367828378: 1.73688471317291
[02:33:21] W1: -1.154378652572632W2: -0.428312063217163B: 0.736884713172913
[02:33:21] TOTAL ERROR: 4
[02:33:21] W1: -1.154378652572632W2: -0.428312063217163B: -0.263115286827087
[02:33:21] Test 0 0: 1
[02:33:21] Test 0 1: 1
[02:33:21] Test 1 0: 0
[02:33:21] Test 1 1: 0


Hierarchy
Scene
Asset Store
Game
Inspector
SampleScene*
Main Camera
Directional Light
Perceptron
Project
Console
Clear
Collapse
Error Pause
Editor
[02:34:48] W1: 2.436104059219366W2: 0.6585015058517468: -2.528023004531866
UnityEngine.Debug:Log (object)
[02:34:48] TOTAL ERROR: 0
[02:34:48] Test 0 0: 0
[02:34:48] Test 0 1: 0
[02:34:48] Test 1 0: 0
[02:34:48] Test 1 1: 1
Perceptron
Transform
Position
Rotation
Scale
Perceptron_sc (Script)
Ts
Output
Input
Element 0
Element 1
Element 2
Element 3


Hierarchy
Scene
Asset Store
Game
Inspector
SampleScene*
Main Camera
Directional Light
Perceptron
Project
Console
Clear
Collapse
Error Pause
Editor
[02:35:39] W1: -0.9442796707153328W2: 0.2445400953292858B: -0.4921123981475838
[02:35:39] W1: 0.05572032928466W2: 1.2445400953292858B: 0.5078876018524178
[02:35:39] TOTAL ERROR: 2
[02:35:39] TOTAL ERROR: 3
Perceptron
Transform
Position
Rotation
Scale
Perceptron_sc (Script)
Ts
Output
Input
Element 0
Element 1
Element 2
Element 3


Hierarchy
Scene
Asset Store
Game
Inspector
SampleScene*
Main Camera
Directional Light
Perceptron
Project
Console
Clear
Collapse
Error Pause
Editor
[02:35:39] W1: 2.055720329284678W2: 1.2445400953292858B: -2.492112398147588
[02:35:39] TOTAL ERROR: 0
[02:35:39] Test 0 0: 0
[02:35:39] Test 0 1: 0
[02:35:39] Test 1 0: 0
[02:35:39] Test 1 1: 1
Perceptron
Transform
Position
Rotation
Scale
Perceptron_sc (Script)
Ts
Output
Input
Element 0
Element 1
Element 2
Element 3