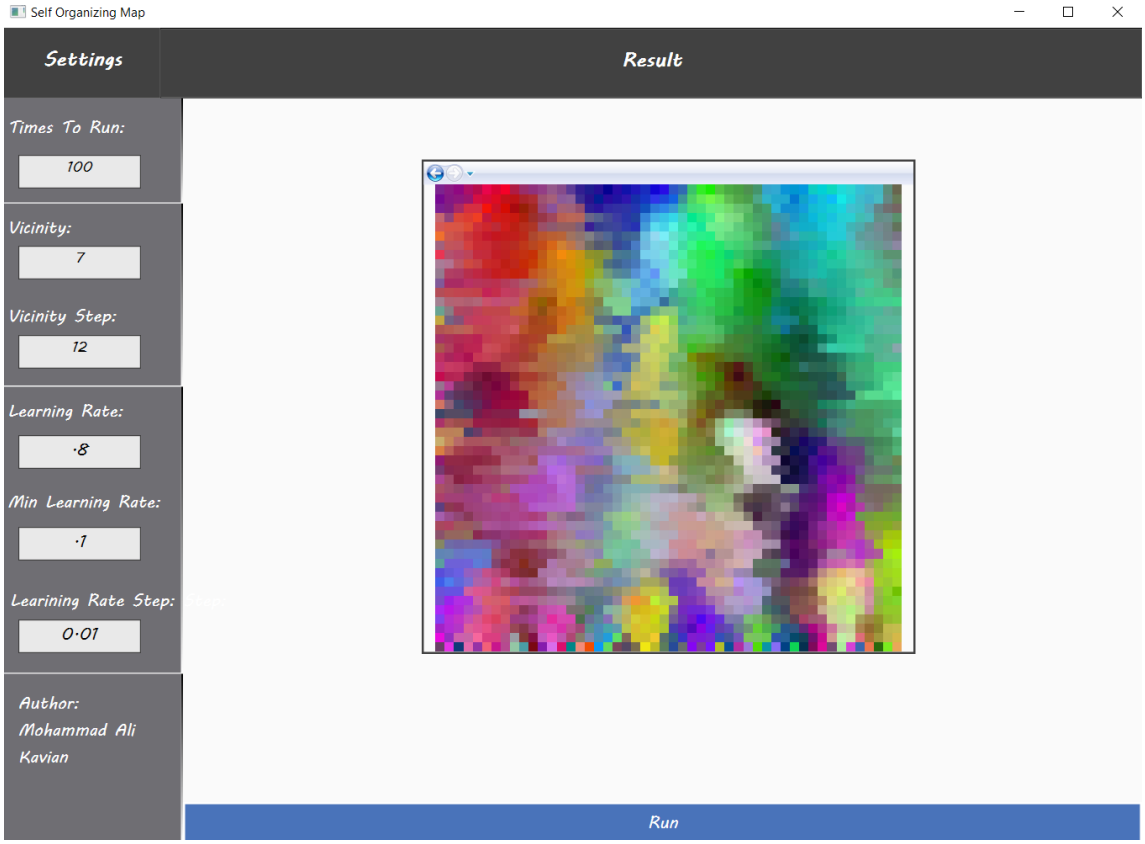
به نام خدا

تشریح پروژه

محمد علی کاویان

پروژه ی مربوطه برنامه نویسی یک شبکه ی عصبی SOM برای ساختن یک palette رنگ بوده است. یک palette رنگ 50 در 50 را قرار است به شبکه ی عصبی SOM بدهیم تا بتواند palette رنگ مورد نظر را بسازد. ابتدا به تصویر زیر توجه کنید که نمای اصلی برنامه را نشان می دهد. در ادامه راجع به نحوه ی مقدار دهی متغیرها (Textbox)و نحوه ی برخورد برنامه با آنها را شرح خواهیم داد.



تصویر بالا بهترین اجرایی بود که از این شبکه ی عصبی گرفته شده است. همچنین توجه کنید که برنامه ی مذکور از لحاظ Responsive ،UI می باشد. در برنامه تمامی متغیرها باید مقداردهی شوند و مقدار دهی آنها بایستی متناسب با نوع آنها باشد. در غیر این صورت بالا دکمه ی Run یک خطا نوشته خواهد شد.

- Times To Run

این Textbox مقادیر صحیح مثبت غیر صفر را می پذیرد. در واقع در این Textbox مشخص می کنید که چندبار می خواهیم داده های آموزش را به شبکه ی عصبی بدهیم. درواقع تعداد مراحل آموزش را مشخص می کند.

- Vicinity

این Textbox مقادیر صحیح مثبت را می پذیرد. در واقع این Textbox شعاع همسایگی را در ابتدای اجرا مشخص می کند. توجه کنید که همسایگی در این برنامه به صورت یک مربع در نظر گرفته شده است که نرون برنده در مرکز آن قرار دارد.

- Vicinity Step

این Textbox مقادیر صحیح مثبت را می پذیرد. در واقع این Textbox مشخص می کند که بعد از چند بار آموزش داده‌ها از مقدار شعاع همسایگی کم شود. به عنوان مثال در تصویر صفحه ی اول مقدار Vicinity، 7 داده شده است و مقدار Vicinity Step، 12 یعنی اینکه بعد از 12 بار اجرای مرحله ی آموزش یکی از مقدار شعاع همسایگی کم شود. یعنی بعد از 12 بار اجرای مرحله ی آموزش مقدار شعاع همسایگی 6 خواهد شد.

- Learning Rate

این Textbox مقادیر اعشاری مثبت، یک و کوچکتر از آن را می پذیرد. در واقع این مقدار، مقدار نرخ یادگیری را مشخص می کند.

- Min Learning Rate

این Textbox مقادیر اعشاری مثبت، یک و کوچکتر از آن را می پذیرد. در واقع این مقدار، مقدار کوچکترین نرخ یادگیری را مشخص می کند تا مقدار نرخ یادگیری در طول اجرا از آن کمتر نشود.

- Learning Rate Step
- این Textbox مقادیر اعشاری مثبت، یک و کوچکتر از آن را می پذیرد. در واقع این مقدار مشخص می کند که بعد از اجرای هر مرحله ی آموزش چقدر از مقدار نرخ یادگیری کم شود. توجه کنید که رابطه ی زیر برای دادن مقدار بایستی برقرار باشد در غیر این صورت با خطا مواجه خواهید شد.

$$Learning\ Rate - Learning\ Rate\ Step > Min\ Learning\ Rate$$

در ادامه به نحوه ی کار برنامه خواهیم پرداخت.

توجه کنید که تمام UI مربوط به برنامه با استفاده از XAML نوشته شده است.

در برنامه یک آرایه ی 3 بعدی paletteNumberArray برای تعداد خانه های 50 در 50 و سه رنگ R، G و B در نظر گرفته شده است. در ابتدای برنامه به این آرایه به صورت زیر مقدار Random برای مقدار دهی به نماینده های نرون ها داده شده است.

```
1 reference
private void initialPalette()
{
    for (int n = 0; n < 3; n++)
    {
        for (int i = 0; i < 50; i++)
        {
            for (int j = 0; j < 50; j++)
            {
                palleteNumberArray[n, i, j] = (byte)randomeNumber.Next(0, 256);
            }
        }
    }
}
```

یک آرایه ی دو بعدی نیز برای داده های آموزشی به تعداد 1000 می باشد نیز در نظر گرفته شده است و در ابتدای برنامه به صورت زیر مقدار Random به آنها داده شده است.

```
for (int n = 0; n < 3; n++)
{
    for (int i = 0; i < 1000; i++)
    {
        randomNumberArray[n,i] = (byte)randomeNumber.Next(0, 256);
    }

}
```

برای بررسی صحیح بودن مقادیر وارد شده در Textbox ها از تابع validation به صورت زیر استفاده شده است.

```
private void validation()
{
    errLine.Content = "";
    errorFlag = false;
    try
    {
        loopCounter=Convert.ToUInt32(loopCount.Text);
        vicinity= Convert.ToUInt32(D.Text);
        vicinityCounter=Convert.ToUInt32(dStep.Text);

        learningRate=Convert.ToDouble(nStartValue.Text);
        if (learningRate < 0 || learningRate > 1)
            errorFlag = true;

        minLearningRate=Convert.ToDouble(nMinValue.Text);
        if (minLearningRate < 0 || minLearningRate > 1)
            errorFlag = true;

        learningRateStep=Convert.ToDouble(nStep.Text);
        if (learningRateStep < 0 || learningRateStep > 1)
            errorFlag = true;
    }
    catch (Exception)
    {
        errorFlag = true;
    }
}
```

برای هر داده ی آموزشی که به شبکه ی عصبی داده می شود با استفاده از فاصله ی اقلیدسی شبیه ترن نرون شبکه ی عصبی به آن انتخاب می شود. و سپس با استفاده از تابع update، نرون برنده خود و همسایه هایش را به روز رسانی می کند. تابع update به صورت زیر می باشد.

```
private void update(uint Neuron, uint neighborSize)
{
    uint i;
    uint j;
    if (winnerNeuron[0] - neighborSize > 0)
        i = winnerNeuron[0] - neighborSize;
    else
    {
        i = 0;
    }
    if (winnerNeuron[1] - neighborSize > 0)
        j = winnerNeuron[1] - neighborSize;
    else
    {
        j = 0;
    }
    for (; i <= (winnerNeuron[0] + neighborSize) && i < 50; i++)
    {
        for (; j <= (winnerNeuron[1] + neighborSize) && j < 50; j++)
        {
            for (uint n = 0; n < 3; n++)
            {
                palleteNumberArray[n, i, j] = (byte)(palleteNumberArray[n, i, j] + learningRate * (randomNumberArray[n, Neuron] - palleteNumberArray[n, i, j]));
            }
        }
    }
}
```

توجه کنید که این تابع update دو ورودی می گیرد. یک شماره نرون مورد نظر در داده های آموزشی می باشد. و دومی شعاع همسایگی می باشد.
در این برنامه برای کار کردن با RGB از کلاس های WriteableBitmap و Image استفاده شده است. به صورت های زیر یک نمونه از کلاس WriteableBitmap و Image ایجاد می کنیم.

```
writeableBitmap = new WriteableBitmap(50 , 50 , 96 , 96, PixelFormats.Rgb24, null);
image = new Image();
RenderOptions.SetBitmapScalingMode(image, BitmapScalingMode.NearestNeighbor);
RenderOptions.SetEdgeMode(image, EdgeMode.Aliased);
image.Source = writeableBitmap;
```

توجه کنید که در تعریف یک نمونه از WriteableBitmap از Rgb24 استفاده شده است، یعنی اینکه برای هر رنگ Red،Green و Blue از 8 بیت استفاده می کند.
برای نوشتن پیکسل های به روز شده پس از آموزش آرایه paletteNumberArray نیز به صورت زیر عمل می کنیم.

```
for (int k = 0; k < 50; k++)
{
    for(int m = 0; m < 50; m++)
    {
        tempArray[0] = palleteNumberArray[0, k, m];
        tempArray[1] = palleteNumberArray[1, k, m];
        tempArray[2] = palleteNumberArray[2, k, m];
        writeableBitmap.WritePixels(new Int32Rect(m, k, 1, 1),tempArray,3,0);
    }
}
paletteHolder.Content = image;
image.Source = writeableBitmap;
```

توجه کنید در تصویر بالا k سطر و m ستون خواهد بود. توجه کنید که رنگ هر پیکسل را در tempArray به ترتیب RGB ریخته ایم و این رنگ هارا به صورت آرایه ای به WritePixels می دهیم. عدد 3،Stride را مشخص می کند. در واقع Stride مشخص می کند که چند بایت در آرایه وجود دارد. عدد 0 نیز offset را مشخص می کند. در واقع مشخص می کند که از چه اندیسی از آرایه داده ها خوانده شود.