DsaiUebung-009--*Train your own LLM* : Assignment (12.2.2025)

Visit The Github-page ! -- by *Jan Ritt*

Teachable Machines

Ascii-Box(Char)-Recognizer

***The Idea*:**

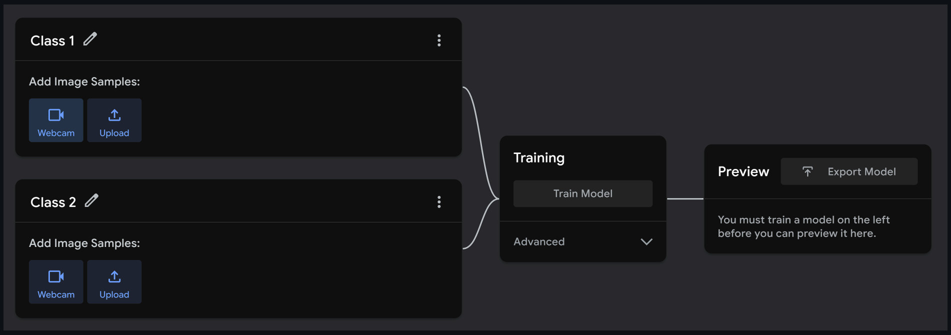
- *I wanted to make something maybe more useful than an LLM to differ between cat & dog pictures.*
- Because on Teachable Machines the LLM training can be done based on picture- & audio-data,
  > - *I choose to train it to recognize ASCII-Box-Characters*[1];
  >
  >   and then whole Boxes and their properties like 'closed-box', 'one-line-stroke-style', 'bold-line-style', 'irregular-box', ...[2]
- The first hurdle was to provide a good dataset for this - in size and variety - **DIY**.
  > - *I coded an application that can generate* **random boxchars** *in* **random RGB-colors**, *on random background-color:*
  >
  >   It prints each char in a consistent surrounding space, takes a screenshot with Skia from this sample and stores it as .png to feed them into the LLM afterwards.*
- The LLM should be able to tell me if a drawn Ascii-box is a valid (eg. `closed` and `coherent`) box, or if my box-creation-code needs some adjustments,
  > - *This would be very helpful, in conjunction to normal unit-testing of each combination of box-/line-attributes,*
  >
  >   *to test my codebases creation and to provide a more robust ASCII-Box-Drawing-Solution, for my usecase.*

As I started coding my Boxdrawer, i found this paragraph on wikipedia and could not belive it at first:

"... *However*, DOS line- and box-drawing characters are not ordered in any programmatic manner, *so calculating* a particular character shape needs a look-up table ..." source Wiki :Box-drawing-characters

1.) Getting familliar with Teachable-Machines

On teachable machines you can train a large language model either on audio, or images.
I chose to use the image-variant, which has this GUI on startup:



The LLM can learn the difference between uploaded classes.
In the first step, i wanted it to just to differ between alphanumeric-chars and the special ASCII-box-chars.

2.) Creating the Trainings-DataSets

To satisfy the "additional requirement" of the assignment - to implement the Model in our own project,
I generated the needed trainings-data instead of using the trained model afterwards.

> *(Because it wasn't specified in the assignment, i assume that either side of the LLM pipeline would be fine.)*

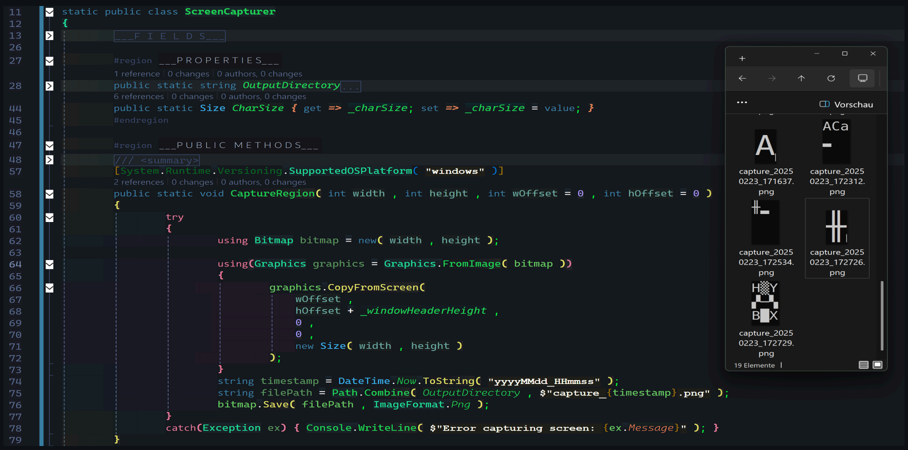
- ## 2.1.) 'The Charifier' -

  My C#-Application for Testdata Creation

  - Comprehensive documentation can be found here .

    > further notes can be found here

  - ***"Heart"* of the application - screencapturing with `SkiaSharp` :**

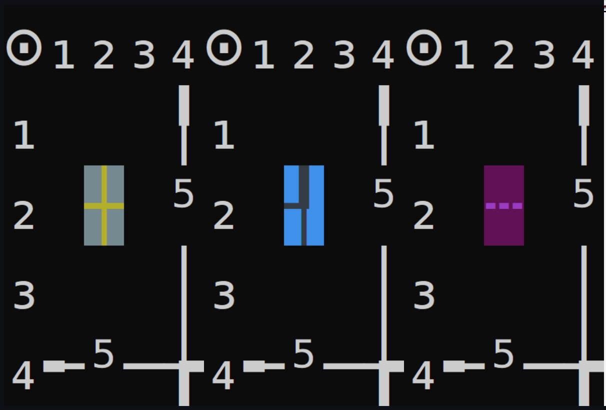


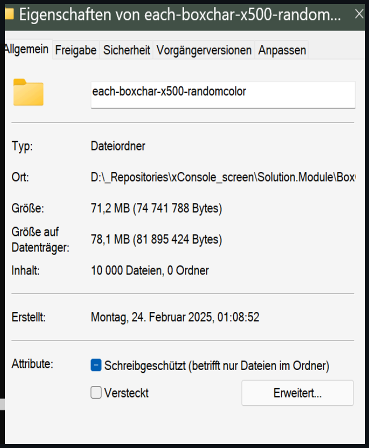
```
static public class ScreenCapturer
{
        ___F I E L D S___

        #region ___PROPERTIES___
        1 reference | 0 changes | 0 authors, 0 changes
        public static string OutputDirectory...
        6 references | 0 changes | 0 authors, 0 changes
        public static Size CharSize { get => _charSize; set => _charSize = value; }
        #endregion

        #region ___PUBLIC METHODS___
        /// <summary>
        [System.Runtime.Versioning.SupportedOSPlatform( "windows" )]
        2 references | 0 changes | 0 authors, 0 changes
        public static void CaptureRegion( int width , int height , int wOffset = 0 , int hOffset = 0 )
        {
                try
                {
                        using Bitmap bitmap = new( width , height );

                        using(Graphics graphics = Graphics.FromImage( bitmap ))
                        {
                                graphics.CopyFromScreen(
                                    wOffset ,
                                    hOffset + _windowHeaderHeight ,
                                    0 ,
                                    0 ,
                                    new Size( width , height )
                                );
                        }
                        string timestamp = DateTime.Now.ToString( "yyyyMMdd_HHmmss" );
                        string filePath = Path.Combine( OutputDirectory , $"capture_{timestamp}.png" );
                        bitmap.Save( filePath , ImageFormat.Png );
                }
                catch(Exception ex) { Console.WriteLine( $"Error capturing screen: {ex.Message}" ); }
        }
```

Vorschau
A
capture_20250223_171637.png
ACa
capture_20250223_172312.png
capture_20250223_172534.png
capture_20250223_172726.png
HY
BX
capture_20250223_172729.png
19 Elemente

- **Examples of the final custom generated trainings-data:**

> - *I generated `500 variants` of each of the `200 Bockchars` => `10.000 BoxChars`.*
> - *And `3400 alpha-numeric` chars (also in random colors) as second trainings-set.*





Eigenschaften von each-boxchar-x500-random... ×

each-boxchar-x500-randomcolor

Typ: Dateiordner

Ort: D:_Repositories\xConsole_screen\Solution.Module\Box

Größe: 71,2 MB (74 741 788 Bytes)

Größe auf Datenträger: 78,1 MB (81 895 424 Bytes)

Inhalt: 10 000 Dateien, 0 Ordner

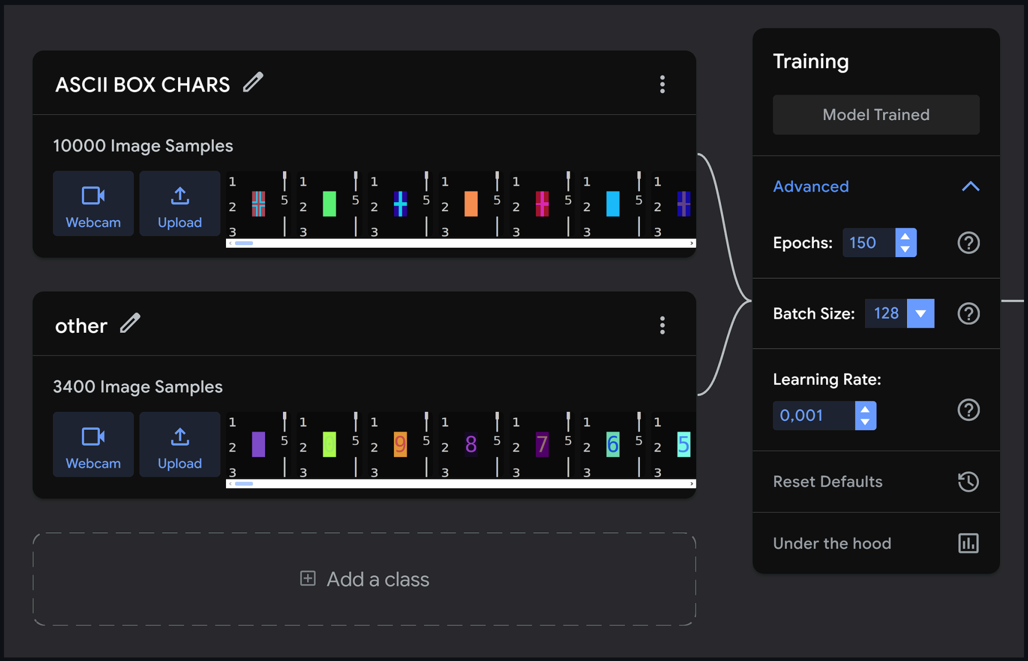
Erstellt: Montag, 24. Februar 2025, 01:08:52

Attribute: ☑ Schreibgeschützt (betrifft nur Dateien im Ordner)

☐ Versteckt

3.) Training the Large Language Model

> To train the model, i uploaded `13400` generated images in total, which took a while..



ASCII BOX CHARS

10000 Image Samples

other

3400 Image Samples

Training

Model Trained

Advanced

Epochs: 150

Batch Size: 128

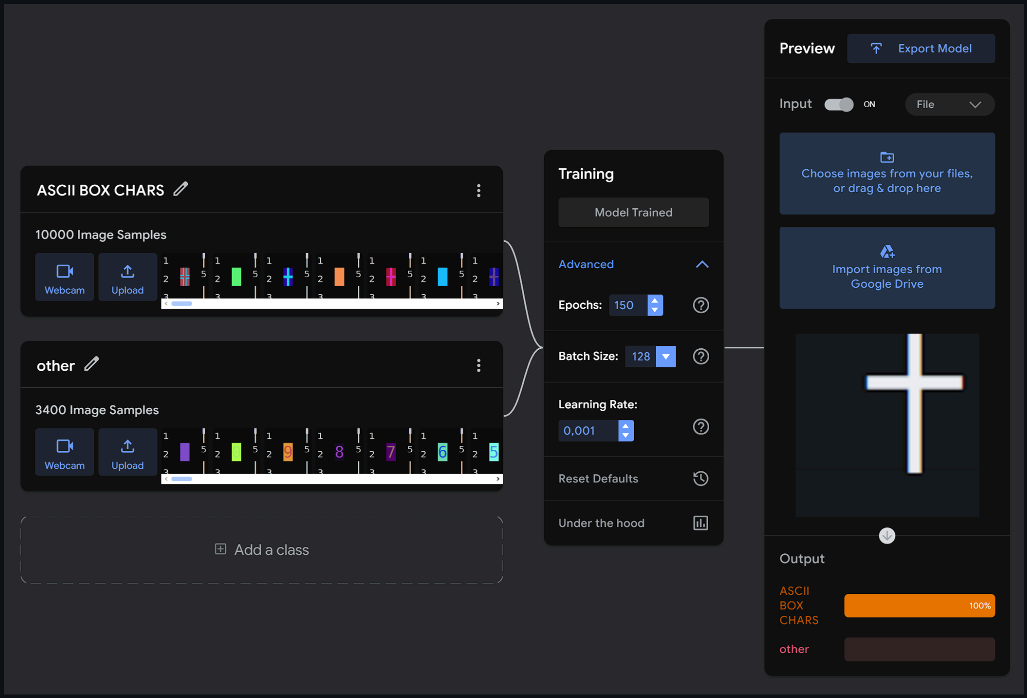
Learning Rate: 0,001

- ▶ Peak 'Under the Hood'

4.) Testing the Capabilities of the Model

- There are no `black-white` sample-chars in the trainings set.

> So I tried to take some screenshots from
> normal console outputs,
> varrying with x-y-offset.
>



ASCII BOX CHARS

10000 Image Samples

Webcam Upload

other

3400 Image Samples

Webcam Upload

Add a class

Training

Model Trained

Advanced

Epochs: 150

Batch Size: 128

Learning Rate: 0,001

Reset Defaults

Under the hood

Preview Export Model

Input ON File

Choose images from your files, or drag & drop here

Import images from Google Drive

Output

ASCII BOX CHARS 100%

other

> *Now it knows exactly how to differentiate the characters on screen at all times.* 🥳 🎉 🤝🏻 💻

You can download the trained model here !

---

1. **All ASCII - Box - Characters**:

```
╵ ╷ ╴╶ ╹ ╻ ╸╺ │ ┃ ║ ╽ ╿ ╎ ╏ ┆ ┇ ┊ ┋ ─━═╼╾╌╍┄┅┈┉
╭╮ ┌┐ ╰╯ └┘ ┍┑ ┎┒ ┕┙ ┖┚ ┏┓ ┗┛ ╒╕ ╓╖ ╘╛ ╙╜ ╔╗ ╚╝
├┤ ┝┥ ┞┦ ┟┧ ┠┨ ┡┩ ┢┪ ┣┫ ╞╡ ╟╢ ╠╣
┬┴┭┮┯┵┶┷┰┱┲┸┹┺┳┻╤╥╧╨╦╩
┼┽┾┿╀╁╂╪╫╃╄╅╆╇╈╉╊╋╬
```
↩

2. *The full implementation of the BoxDrawer and updated trainings-set is still in-development, yet.* ↩